ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΕΚΜΑΘΗΣΗΣ ΤΕΧΝΙΚΩΝ ΚΑΙ ΕΝΝΟΙΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΕ ΤΗΝ ΡΥΤΗΟΝ

Διπλωματική Εργασία

του

Γιαγκούδη Ιωάννη

Θεσσαλονίκη, Ιούνιος 2019

ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΕΚΜΑΘΗΣΗΣ ΤΕΧΝΙΚΩΝ ΚΑΙ ΕΝΝΟΙΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΕ ΤΗΝ ΡΥΤΗΟΝ

Γιαγκούδης Ιωάννης

Πτυχίο Τμήματος Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, 2016

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων/ουσα Καθηγητής/τρια
Ξυνόγαλος Στυλιανός

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 25/6/2019

Ξυνόγαλος Στέλιος	Σατρατζέμη Μαρία	Κολωνιάρη Γεωργία
.....	.....	.....

Γιαγκούδης Ιωάννης

.....

Περίληψη

Ο σκοπός της εργασίας ήταν η δημιουργία εκπαιδευτικού υλικού και η ανάπτυξη συστήματος μέσω του οποίου μπορεί να αξιοποιηθεί αυτό για την εκμάθηση βασικών τεχνικών και εννοιών προγραμματισμού με τη γλώσσα Python.

Στα πλαίσια της εργασίας αναπτύχθηκε εφαρμογή ιστού με τη χρήση του framework της Python Django που προσφέρει πρόσβαση σε θεωρητικό εκπαιδευτικό υλικό, δυνατότητα επίλυσης ερωτήσεων πολλαπλής επιλογής και προγραμματιστικών προβλημάτων.

Η εφαρμογή ενσωματώνει επίσης στοιχεία παιχνιδιοποίησης (gamification) όπως τη συλογή πόντων και επιτευγμάτων (achievements) με στόχο να κεντρίσει το ενδιαφέρον του χρήστη.

Λέξεις Κλειδιά: παιχνιδοποίηση, εφαρμογή ιστού, Python, Django

Abstract

The purpose of this thesis is to create educational material and to develop a system through which somebody can learn the basics of programming with Python.

In the context of this thesis a web application was developed, using Python and the framework Django, that gives access to educational material, multiple choice tests and programming exercises.

The application incorporates gamification elements such as the ability to earn achievements in an attempt to stimulate user interest.

Keywords: gamification, web app, Python, Django

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον Αναπληρωτή Καθηγητή κ. Ξυνόγαλο Στυλιανό για την καθοδήγηση και την υπομονή του και την οικογένεια μου για τη στήριξη τους όλα τα χρόνια των σπουδών μου.

Περιεχόμενα

1 Εισαγωγή 1

1.1 Πρόβλημα – Σημαντικότητα του θέματος 1

1.2 Σκοπός – Στόχοι 1

1.3 Συνεισφορά 2

1.4 Διάρθρωση της μελέτης 2

2 Βιβλιογραφική Επισκόπηση – Θεωρητικό Υπόβαθρο 4

2.1 Παιχνιδοποίηση (Gamification) 4

2.1.1 Ορισμός 4

2.1.2 Στοιχεία Παιχνιδοποίησης 5

2.1.3 Παραδείγματα Παιχνιδοποίησης 6

2.1.4 Πλεονεκτήματα Παιχνιδοποίησης 7

2.2 Εφαρμογές Ιστού 7

2.2.1 Ορισμός 7

2.2.2 Τεχνολογίες 7

2.2.3 Παράγοντες επιλογής Τεχνολογίας 10

3 Σχεδίαση και Ανάλυση 11

3.1 Εκπαιδευτικό Υλικό 11

3.2 Τεχνολογίες που χρησιμοποιήθηκαν 12

3.2.1 Python 12

3.2.2 Django 12

3.2.3 SQLite (development) / PostgreSQL (production) 12

3.2.4 HTML 13

3.2.5 CSS 13

3.2.6 Javascript 13

3.2.7 Σύνοψη 13

3.3 Ανάπτυξη Εφαρμογής Ιστού Με Django 14

3.3.1 Δημιουργία Project 14

3.3.2 Δημιουργία Ενός App 15

3.3.3 Περιγραφή των Υποεφαρμογών (Apps) 23

4 Παρουσίαση Εφαρμογής 33

4.1 Λειτουργία για Χρήστες 33

4.2 Λειτουργία για Διαχειριστή 45
5 Επίλογος 49
5.1 Σύνοψη και συμπεράσματα 49
5.2 Όρια και περιορισμοί της έρευνας 50
5.3 Μελλοντικές Επεκτάσεις 50
Παράρτημα Α - Κώδικας 53
Παράρτημα Β - Εκπαιδευτικό Υλικό 89
Παράρτημα C - Προγραμματιστικές Ασκήσεις 103
Παράρτημα D - Τεστ Πολλαπλής Επιλογής 121

Κατάλογος Εικόνων

Εικόνα 3-1: Δομή μιας Υποεφαρμογής 15
Εικόνα 4-1: Αρχική Οθόνη 33
Εικόνα 4-2: Φόρμα Εγγραφής 34
Εικόνα 4-3: Φόρμα Σύνδεσης 35
Εικόνα 4-4: Λίστα με Παραδείγματα Κώδικα 36
Εικόνα 4-5: Παράδειγμα Κώδικα 37
Εικόνα 4-6: Λίστα με Προγραμματιστικές Ασκήσεις 37
Εικόνα 4-7: Παράδειγμα Άσκησης 38
Εικόνα 4-8: Παράδειγμα Αποτελέσματος Σωστής Λύσης 39
Εικόνα 4-9: Παράδειγμα Αποτελέσματος Λάθος Λύσης 40
Εικόνα 4-10: Παράδειγμα Χρήσης Λάθος Ονόματος Μεταβλητής 41
Εικόνα 4-11: Παράδειγμα Αποθηκευμένης Λύσης 42
Εικόνα 4-12: Λίστα με Κουιζ 42
Εικόνα 4-13: Παράδειγμα Κουιζ 43
Εικόνα 4-14: Παράδειγμα Αποτελεσμάτων Κουιζ 43
Εικόνα 4-15: Πίνακας Κατάταξης 44
Εικόνα 4-16: Παράδειγμα Προφίλ Χρήστη 44
Εικόνα 4-17: Παράδειγμα Σελίδας Προόδου 45
Εικόνα 4-18: Κεντρική σελίδα διαχείρισης 46
Εικόνα 4-19: Φόρμα Προσθήκης Άσκησης 47
Εικόνα 4-20: Φόρμα Αλλαγής ή Διαγραφής Άσκησης 47
Εικόνα 4-21: Φόρμα Δημιουργίας Κουίζ 48
Εικόνα 4-22: Φόρμα Δημιουργίας Ερώτησης 48

Κατάλογος Κώδικα

Κώδικας 3-1: Προβολή Βασισμένη σε Μέθοδο 16
Κώδικας 3- 2: Προβολή Βασισμένη σε Κλάση 17
Κώδικας 3- 3: Οι Διευθύνσεις URL μιας Υποεφαρμογής 18
Κώδικας 3-4: Το Κεντρικό Αρχείο urls.py της Εφαρμογής 19
Κώδικας 3-5: Τα μοντέλα του codingExcercise 20
Κώδικας 3-6: Το Πρότυπο του CodingExcerciseListView 21
Κώδικας 3-7: Ρυθμίσεις για Χρήση sqlite 22
Κώδικας 3-8: Ρυθμίσεις για Χρήση PostgreSQL 22
Κώδικας 3-9: Λίστα με Εγκατεστημένες Εφαρμογές 23
Κώδικας 3-10: Κώδικας για την Εκτέλεση Τεστ 25
Κώδικας 3-11: Παράδειγμα Τεστ 26
Κώδικας 3-12: Παράδειγμα Επιτεύγματός 32

Κατάλογος Πινάκων

Πίνακας 1: Γλώσσες προγραμματισμού και back-end frameworks..... 9

1 Εισαγωγή

1.1 Πρόβλημα – Σημαντικότητα του θέματος

Η τεχνολογία βρίσκεται παντού τα τελευταία χρόνια και η γνώση προγραμματισμού είναι μια από τις βασικές δεξιότητες του 21ου αιώνα. Για τον λόγο αυτό ηγέτες του χώρου συμφωνούν πως οι δεξιότητες τεχνογνωσίας είναι απαραίτητες για τους εργαζομένους. Ο Bill Gates έχει δηλώσει πως η εκμάθηση προγραμματισμού διευρύνει το μυαλό και βοηθά στην καλύτερη σκέψη, δημιουργεί έναν ευρύτερο τρόπο σκέψης που είναι χρήσιμος σε όλους τους τομείς της ζωής [1].

Η Python θεωρείται μια από τις κορυφαίες γλώσσες προγραμματισμού και χρησιμοποιείται από κορυφαίες εταιρίες όπως η Google καθώς και από σημαντικά πανεπιστημιακά ιδρύματα. Η Python είναι μια γλώσσα γενικού σκοπού αφού κανείς μπορεί να την χρησιμοποιήσει για να δημιουργήσει από ένα παιχνίδι εως μέχρι και την λειτουργικότητα μιας ιστοσελίδας. Το απλό συντακτικό της την κάνει μια καλή επιλογή για αρχάριους στον χώρο του προγραμματισμού.

Τα τελευταία χρόνια το μερίδιο αγοράς των σοβαρών παιχνιδιών και των εφαρμογών με στοιχεία παιχνιδοποίησης έχει αυξηθεί σημαντικά. Σύμφωνα με έρευνες έχει παρατηρηθεί πως η χρήση σοβαρών παιχνιδιών και παιχνιδοποίησης έχει πλεονεκτήματα σε σχέση με παραδοσιακές μεθόδους εκμάθησης προγραμματισμού (Conolly et al. (2012)).

1.2 Σκοπός – Στόχοι

Ο λόγος που με οδήγησε στην επιλογή του συγκεκριμένου θέματος ήταν η επιθυμία να δημιουργήσω μια εφαρμογή με πρακτική αξία.

Στόχος της εργασίας είναι όταν ο χρήστης ολοκληρώσει όλες τις δραστηριότητες που του προσφέρονται να έχει μια καλή κατανόηση των βασικών αρχών προγραμματισμού με Python, να γνωρίζει το συντακτικό της και την ικανότητα να δημιουργεί προγράμματα που μπορούν να επιλύσουν απλά προβλήματα.

[1] https://code.org/quotes

1.3 Συνεισφορά

Μέσω της εφαρμογής κάποιος θα μπορεί να εξοικειωθεί με τις βασικές έννοιες προγραμματισμού με την Python χωρίς να χρειάζεται να εγκαταστήσει τον διερμηνευτή ή κάποιο περιβάλλον ανάπτυξης στον υπολογιστή του. Συγκεκριμένα:

- Ο χρήστης εισάγεται σε βασικές έννοιες προγραμματισμού μέσω απλών παραδειγμάτων κώδικα.
- Ελέγχει την κατανόηση των σχετικών εννοιών μέσω κουιζ με ερωτήσεις κλειστού τύπου. Μετά την ολοκλήρωση ενός κουίζ παρουσιάζονται στον χρήστη τόσο οι απαντήσεις που έδωσε όσο και οι σωστές απαντήσεις.
- Αναπτύσσει προγράμματα για απλά προβλήματα, η ορθότητα των οποίων ελέγχεται μέσω προκαθορισμένων περιπτώσεων ελέγχου. Ο χρήστης έχει τη δυνατότητα να αποθηκεύσει κάθε σωστή απάντηση.

Αφού ο χρήστης εξοικειωθεί με τις βασικές έννοιες προγραμματισμού σε Python και έχοντας γνώση των λειτουργιών της συγκεκριμένης εφαρμογής μπορεί να χρησιμοποιήσει τον κώδικά της ως μια μελέτη περίπτωσης μιας διαδικτυακής εφαρμογής με την Pyhton και το Django. Ο κώδικας της εφαρμογής διατίθεται στο GitHub: https://github.com/JohnGiag/PyLearnWebApp και η εφαρμογή στο Heroku: https://pylearn.herokuapp.com/.

1.4 Διάρθρωση της μελέτης

Στο κεφαλαίο 2 θα συζητηθεί ο ορισμός του όρου της παιχνιδοποίησης και στη συνέχεια θα γίνει ανασκόπηση εφαρμογών ιστού με στοιχεία παιχνιδοποίησης. Επίσης, θα γίνει αναφορά στις τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη εφαρμογών ιστού.

Το κεφάλαιο 3 παρουσιάζει το εκπαιδευτικό υλικό, τη διαδικασία ανάπτυξης της εφαρμογής και την αρχιτεκτονική της.

Στο κεφάλαιο 4 παρουσιάζεται η εφαρμογή και οι διαφορετικές λειτουργίες που προσφέρονται στους χρήστες και στον διαχειριστή.

Στο κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα που προέκυψαν και προτάσεις για μελλοντική έρευνα.

2 Βιβλιογραφική Επισκόπηση – Θεωρητικό Υπόβαθρο

2.1 Παιχνιδοποίηση (Gamification)

2.1.1 Ορισμός

Ο όρος παιχνιδοποίηση εμφανίστηκε αρχικά το 2008 αλλά δεν κέρδισε δημοτικότητα μέχρι το 2010 (Deterding (2011)) . Ο όρος έχει αμφισβητηθεί από τα μέλη της αγοράς ανάπτυξης παιχνιδιών και συχνά χρησιμοποιείται με αρνητική χροιά εξαιτίας της δυσαρέσκειας τους ως προς τη χρήση απλοποιημένων στοιχείων ανάπτυξης παιχνιδιών που χρησιμοποιούνται και της εκμεταλλευτικής φύσης πολλών εφαρμογών που έχουν ως αποκλειστικό στόχο την παραγωγή κέρδους και όχι την επιμόρφωση , την διασκέδαση ή ανταμοιβή του κοινού τους.

Η παιχνιδοποίηση αναφέρεται στη χρήση στοιχείων ανάπτυξης παιχνιδιών σε καταστάσεις που δεν έχουν παραδοσιακά σχέση με παιχνίδια με στόχο την αύξηση της δραστηριότητας του χρήστη. Στοιχεία παιχνιδοποίησης μπορούν να χρησιμοποιηθούν σε πολλές εφαρμογές όπως κοινωνικής δικτύωσης, ψυχαγωγίας, ενημέρωσης, εκπαιδευτικές. Άλλοι όροι που συνυπάρχουν και αναφέρονται σε ίδιες ή παρόμοιες τεχνικές με την παιχνιδοποίηση είναι παραγωγικά παιχνίδια, παιχνίδια συμπεριφοράς και “funware”.

Οι Robinson και Bellotti (2013) έχουν χωρίσει τα δομικά συστατικά παιχνιδοποίησης σε έξι υποκατηγορίες:

- Γενικό πλαίσιο (General Framing): παροχή πληροφοριών σχετικά με τον στόχο της εφαρμογής με σκοπό την παρότρυνση του χρήστη.
- Γενικοί κανόνες και πλαίσιο αξιολόγησης (General Rules and Performance Framing): επεξήγηση των κανόνων του παιχνιδιού ώστε να γνωρίζει ο χρήστης τι καθιστά «σωστή» συμπεριφορά στο πλαίσιο του παιχνιδιού
- Κοινωνικά Στοιχεία (Social Features): να δίνεται η δυνατότητα στον χρήστη να αλληλεπιδράσει με άλλους μέσα στο παιχνίδι
- Κίνητρα (Incentives): τα κίνητρα μπορεί να είναι είτε εσωτερικά π.χ. περιέργεια, διασκέδαση είτε εξωτερικά π.χ. οικονομική ανταμοιβή.

- ❖ Πόροι και περιορισμοί (Resources and Constraints): τα όρια του παιχνιδιού μέσα στα οποία πρέπει να λειτουργήσει ο παίκτης.
- ❖ Ανατροφοδότηση (Feedback and Status Information): παροχή πληροφοριών στον χρήστη ως προς το επίπεδο της απόδοσης του.

2.1.2 Στοιχεία Παιχνιδοποίησης

Στοιχεία παιχνιδοποίησης που χρησιμοποιούνται συχνά είναι (Glover (2013)):

- ❖ Πόντοι: η εφαρμογή ανταμείβει τον χρήστη με πόντους όταν ολοκληρώνει κάποια δραστηριότητα τους οποίους είτε μπορεί να τους χρησιμοποιήσει ενεργά ώστε να αγοράσει κάτι που προσφέρεται μέσα στην εφαρμογή π.χ. αντικείμενα για τη διακόσμηση του χαρακτήρα του ή απλά για να γίνει μια κατάταξη όλων των χρηστών.

- ❖ Πίνακες Κατάταξης: πίνακες με τους πόντους όλων των χρηστών όπου ο καθένας μπορεί να συγκρίνει την απόδοση του.

- ❖ Επιτεύγματα: επιβράβευση των χρηστών όταν επιτυγχάνουν κάποιο μεγάλο στόχο με απονομή εμβλημάτων που φαίνονται στο προφίλ τους.

- ❖ Εικονικός κόσμος: δημιουργία περιβάλλοντος με γραφικά δύο ή τριών διαστάσεων μέσα στο οποίο εξελίσσεται το παιχνίδι για τη διασκέδαση του χρήστη.

- ❖ Ιστορία: χρήση σεναρίου και παροχή πληροφοριών σχετικά με την ιστορία του εικονικού κόσμου

2.1.3 Παραδείγματα Παιχνιδοποίησης

Όπως ήδη αναφέρθηκε η παιχνιδοποίηση μπορεί να χρησιμοποιηθεί για τον εμπλουτισμό πολλών διαφορετικών ειδών εφαρμογών. Το gamification έχει χρησιμοποιηθεί σε διάφορους χώρους όπως αυτός της υγείας (King (2013)), κοινωνικής προσφοράς, επιχειρηματικότητας (Mitchell et al. (2018)), εκπαίδευσης (Borges et al.(2014)), αύξησης παραγωγικότητας, κοινωνικής δικτύωσης, διασκέδασης και πολλούς άλλους.

Το FoldIt ένα διαδικτυακό παιχνίδι παζλ που δημιουργήθηκε από το πανεπιστήμιο της Washington οδήγησε σε ανακάλυψη της δομής ενός ενζύμου που σχετίζετε με το AIDS που η επιστημονική κοινότητα αναζητούσε για μια δεκαετία.

Το FreeRice είναι ένα παιχνίδι με κουίζ για πληθώρα γνωστικών αντικειμένων όπου ο χρήστης κερδίζει γνώση απαντώντας στις διάφορες ερωτήσεις και για κάθε σωστή απάντηση του δωρίζεται ρύζι σε άτομα που το έχουν ανάγκη.

Το Sellfee είναι ένα παιχνίδι για κινητές συσκευές όπου ο παίκτης ανοίγει το δικό του εικονικό online κατάστημα και επιλέγει την ονομασία και τι προϊόντα θέλει να πουλήσει με στόχο να δημιουργήσει την περιουσία του. Το κατάστημα όμως δεν είναι απλά εικονικό και μέρος του παιχνιδιού, αλλά υπάρχει στην πραγματικότητα και τα προϊόντα είναι διαθέσιμα για αγορά, ενώ παίκτης μπορεί να διαχειριστεί την επιχείρηση του μέσα από την απλή διεπαφή της εφαρμογής.

Το CodinGame είναι μια πλατφόρμα που δίνει τη δυνατότητα να ανταγωνιστείς έναντι άλλων χρηστών σε ασκήσεις προγραμματισμού και υποστηρίζει πάνω από εικοσιπέντε γλώσσες προγραμματισμού , ανάμεσα τους και την Python.

Μία ακόμα πλατφόρμα που δίνει τη δυνατότητα στους χρήστες να ανταγωνιστούν είναι το SoloLearn. Ο νικητής του διαγωνισμού κερδίζει πόντους εμπειρίας και σκαρφαλώνει τον πίνακα με τα μεγαλύτερα σκορ. Οι χρήστες έχουν επίσης τη δυνατότητα να δημιουργήσουν τις δικές τους πρωτότυπες εφαρμογές και να τις υποβάλουν ώστε να πάρουν feedback από την κοινότητα.

2.1.4 Πλεονεκτήματα Παιχνιδοποίησης

Η παιχνιδοποίηση είναι μια τεχνική που χρησιμοποιείται στην εκμάθηση επειδή προσφέρει μια πληθώρα πλεονεκτημάτων, μιας και μπορεί να απλοποιεί σύνθετα θέματα και να τα παρουσιάζει με τέτοιο τρόπο που κεντρίζει το ενδιαφέρον των μαθητών (Kiryakova et al.(2014)).

Όπως φαίνεται και από πολλά από τα παραδείγματα που αναφέρθηκαν παραπάνω η παιχνιδοποίηση συχνά ενθαρρύνει ή ακόμα και επιβάλει την αλληλεπίδραση ανάμεσα στους μαθητές δημιουργώντας έτσι την αίσθηση μιας κοινότητας.

Οι μαθητές αισθάνονται υπερήφανοι επειδή οδηγούνται να βρουν λύσεις στα προβλήματα που τους παρουσιάζονται μόνοι τους αντίθετα με παραδοσιακές μεθόδους διδασκαλίας όπου ο καθηγητής έχει το βασικό ρόλο.

Τέλος, δίνει τη δυνατότητα στους μαθητές να ασχοληθούν με τη διδακτική ύλη εκτός του χώρου διδασκαλίας και την ελευθερία να δοκιμάσουν διαφορετικές προσεγγίσεις σε ένα πρόβλημα χωρίς να υπάρχει επίκριση ή αρνητικές συνέπειες σε περίπτωση αποτυχίας.

2.2 Εφαρμογές Ιστού

2.2.1 Ορισμός

Εφαρμογή ιστού ονομάζεται μια εφαρμογή που παρέχεται στους χρήστες μέσω του διαδικτύου. Ο χρήστης χρειάζεται μόνο να είναι συνδεδεμένος στο διαδίκτυο και να έχει εγκατεστημένο κάποιο πρόγραμμα περιήγησης όπως το Firefox ή το Chrome ώστε να έχει πρόσβαση στην εφαρμογή.

2.2.2 Τεχνολογίες

Υπάρχουν διάφορες τεχνολογίες που μπορούν να χρησιμοποιηθούν για την ανάπτυξη εφαρμογών ιστού ανάλογα με τις ιδιαιτερότητες και τις απαιτήσεις του εκάστοτε έργου. Η στοίβα τεχνολογίας (technology stack) αποτελείται από εργαλεία και framework που χρησιμοποιούνται για την ανάπτυξη μιας εφαρμογής και χωρίζεται σε front-end και back-end.

2.2.2.1 Front-End / Client Side

Τα front-end στοιχεία επιτρέπουν την αλληλεπίδραση του χρήστη με μια εφαρμογή ιστού. Τα τρία βασικά εργαλεία που χρησιμοποιούνται για την δημιουργία του front-end μιας εφαρμογής ιστού είναι:

- HTML
- CSS
- Javascript

Η HTML ορίζει τη δομή μιας ιστοσελίδας, η CSS χρησιμοποιείται για να δημιουργήσει το στυλ της και η Javascript για να της δώσει περεταίρω λειτουργικότητα.

Τέλος υπάρχουν έτοιμες λύσεις και γνωστά framework για HTML και CSS όπως το Bootsrap που κάνει ευκολότερη την ανάπτυξη ιστοσελίδων παρέχνοντας έναν εύκολο και γρήγορο τρόπο για να κάνει κάποιος μια ιστοσελίδα να μοιάζει επαγγελματική και για Javascript όπως τα Angular , React, Ember και Vue.js.

2.2.2.2 Back-End / Server Side

Σε αντίθεση με το front-end που έχουμε περιορισμένες επιλογές στο back-end υπάρχουν πολλές επιλογές ανάλογα με τις ανάγκες της εφαρμογής (Πίνακας 1). Τα βασικά συστατικά του back-end είναι:

- Το λειτουργικό σύστημα
- ο server
- η βάση δεδομένων
- η γλώσσα ανάπτυξης της εφαρμογής
- το framework που θα επιλεχθεί για την ανάπτυξη της εφαρμογής

Πίνακας 1: Γλώσσες προγραμματισμού και back-end frameworks

Γλώσσα προγραμματισμού	Framework
Javascript	Node.js
Ruby	Ruby on Rails
Python	Django
PHP	Laravel
C#	.NET
Java	Spring, Struts

Η επιλογή των στοιχείων αυτών δεν είναι απαραίτητα ανεξάρτητη αφού υπάρχουν ήδη διάφοροι αποδοτικοί συνδυασμοί που ενδείκνυνται. Κάποιοι από τους πιο χρησιμοποιημένους είναι:

- LAMP: είναι τα αρχικά για Linux (λειτουργικό σύστημα), Apache (server), MySQL (βάση δεδομένων), PHP (γλώσσα προγραμματσμού). Αυτή η στοίβα τεχνολογίας συναντάται συχνά λόγω της επεκτασιμότητας, ασφάλειας και τις δυνατότητες προσαρμογής που προσφέρει (Lee,Ware (2003)).
- Python-Django: βασισμένη στην γλώσσα προγραμματισμού Python αυτή η στοίβα χρησιμοποιεί Apache, MySQL και το framework Django της Python και βοηθάει στην γρήγορη δημιουργία εφαρμογών (Django).
- MEAN: είναι τα αρχικά για MongoDB (βάση δεδομένων), Express.js (framework ανάπτυξης), Angular (front-end framework), Node.js (server). Η στοίβα αυτή χρησιμοποιεί Javascript στο front-end και στο back-end και φημίζεται για την απόδοση και την ευκολία χρήσης της (Dickey (2014)).
- MERN: είναι τα αρχικά για MongoDB (βάση δεδομένων), Express.js (framework ανάπτυξης), React (front-end framework), Node.js (server). Είναι μια εναλλακτική της MEAN όπου γίνεται χρήση του framework React αντί για Angular (Subramanian (2017)).

2.2.3 Παράγοντες επιλογής Τεχνολογίας

Ανάλογα με τις ανάγκες μιας εφαρμογής κάποια στοίβα τεχνολογίας είναι πιο κατάλληλη από τις υπόλοιπες για την υλοποίηση της.

Ένας από τους παράγοντες είναι το μέγεθος της εφαρμογής. Για μικρά project τόσο η στοίβα Python-Django όσο και η Node.js-React είναι καλές επιλογές. Για project μέσου μεγέθους όπως για παράδειγμα ένα e-shop οι στοίβες MEAN, MERN και LAMP είναι καλές επιλογές λόγω απόδοσης και ασφάλειας. Τέλος, για μεγάλα project όπως social networks χρησιμοποιούνται διάφορες τεχνολογίες με τον κορμό τους να χρησιμοποιεί στοίβες όπως Angular-Node.js, Python-Django και Ruby on Rails.

Άλλοι παράγοντες είναι η ταχύτητα με την οποία πρέπει να ολοκληρωθεί το έργο. Η στοίβα MEAN είναι μια καλή επιλογή πρέπει όμως να λάβουμε υπόψιν τις γνώσεις και την οικειότητα του developer με τις τεχνολογίες.

Άλλος παράγοντας είναι η δυνατότητα επεκτασιμότητας της εφαρμογής , η προσθήκη καινούργιας λειτουργικότητας καθώς και η δυνατότητα να χειριστεί η εφαρμογή μεγάλο αριθμό αιτημάτων. Η χρήση Node.js και Angular δίνει καλά αποτελέσματα.

3 Σχεδίαση και Ανάλυση

Σε αυτό το κεφάλαιο θα συζητηθούν το εκπαιδευτικό υλικό, οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής και ο λόγος που επιλέχτηκαν και θα παρουσιαστεί η διαδικασία ανάπτυξης της εφαρμογής με Python και Django και η λειτουργικότητα της.

3.1 Εκπαιδευτικό Υλικό

Η εφαρμογή απευθύνεται σε αρχάριους με ελάχιστη ή καθόλου εμπειρία προγραμματισμού.

Το εκπαιδευτικό υλικό έχει χωριστεί σε επτά ενότητες:

- Βασικά (Basics): Σε αυτό το κεφάλαιο περιλαμβάνονται τα πρώτα πράγματα που χρειάζεται να μάθει κάποιος για να προγραμματίσει όπως η δυνατότητα εκτύπωσης στην κονσόλα, τι είναι οι μεταβλητές και τους τύπους μεταβλητών.
- Δομές Δεδομένων (Data Structures): Σε αυτό το κεφάλαιο ο χρήστης μαθαίνει για τις βασικές δομές δεδομένων που χρησιμοποιούνται στον προγραμματισμό με Python, λίστες (lists), πλειάδες (tuples), λεξικά (dictionaries).
- Μεθόδους (Functionς): Σε αυτό το κεφάλαιο ο χρήστης μαθαίνει να δημιουργεί μεθόδους.
- Δομές Επιλογής (Control Flow): Αυτό το κεφάλαιο αναφέρεται στην χρήση δομών επιλογής όπως το «if/else».
- Δομές Επανάληψης (Loops): Αυτό το κεφάλαιο αναφέρεται στην χρήση δομών επανάληψης, “for loops”, “while loops”.
- Αλφαριθμητικά (Strings): Σε αυτό το κεφάλαιο ο χρήστης μαθαίνει κάποιες χρήσιμες μεθόδους για τη διαχείριση αλφαριθμιτικών.
- Αντικειμενοστρεφής Προγραμματισμός (Object-Oriented Programming): Αυτό το κεφάλαιο αφορά τα βασικά στοιχεία του αντικειμενοστρεφούς προγραμματισμού όπως τη δημιουργία κλάσεων και την κληρονομικότητα μεταξύ τους.

Για κάθε θεματική ενότητα υπάρχουν παραδείγματα κώδικα και θεωρία, προγραμματιστικές ασκήσεις και τεστ ερωτήσεων πολλαπλής επιλογής.

3.2 Τεχνολογίες που χρησιμοποιήθηκαν

3.2.1 Python

Η Python είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού υψηλού επιπέδου. Έχει απλό συντακτικό κάνοντας την αρκετά εύκολη να την μάθει κάποιος σε μικρό χρονικό διάστημα αλλά αρκετά ισχυρή ώστε να υπάρχει η δυνατότητα να δημιουργήσει κανείς περίπλοκες εφαρμογές με αυτή. Μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών web και desktop και είναι open-source κάνοντας τη χρήση της δωρεάν (Python). Ο λόγος που επιλέχθηκε η Python είναι το framework Django.

3.2.2 Django

Το Django είναι ένα από τα πιο διαδεδομένα framework για την ανάπτυξη εφαρμογών ιστού με Python. Ένα framework σαν το Django παρέχει λειτουργίες όπως η διαχείριση βάσεων δεδομένων. Το Django επιλέχθηκε επειδή κάνει εύκολη και γρήγορη την ανάπτυξη ενός MVP (minimum value product), δίνει τη δυνατότητα επεκτασιμότητας, έχει άριστο documentation και μεγάλη κοινότητα χρηστών. Η δομή μιας εφαρμογής που έχει δημιουργηθεί με Django θα αναλυθεί σε βάθος σε επόμενο κεφάλαιο.

3.2.3 SQLite (development) / PostgreSQL (production)

Οι τεχνολογίες βάσεων δεδομένων που χρησιμοποιήθηκαν είναι η SQlite κατά την ανάπτυξη και η PostgreSQL για το τελικό προϊόν . Η SQlite είναι προεπιλεγμένη βάση δεδομένων από το Django και η PostgreSQL επιλέχθηκε επειδή είναι open source.

3.2.4 HTML

Η HTML είναι γλώσσα σήμανσης για ιστοσελίδες και ένα από τα τρία στοιχεία του front-end κάθε εφαρμογής ιστού.

3.2.5 CSS

Η CSS (Cascading Style Sheets) χρησιμοποιείται για να δοθεί στυλ στα στοιχεία της HTML. Για την εφαρμογή μου χρησιμοποίησα το front-end framework Bootstrap που προσφέρει έτοιμα templates.

3.2.6 Javascript

Η Javascript είναι μια γλώσσα που υποστηρίζεται από όλους τους σύγχρονους browser και είναι αυτή που κάνει δυνατή την πιο περίπλοκη αλληλεπίδραση του χρήστη με την ιστοσελίδα.

3.2.6.1 Brython.js

Η Brython.js είναι μια βιβλιοθήκη που κάνει δυνατή τη χρήση της Python αντί για Javascript (Brython) για την λειτουργικότητα του front-end. Ο λόγος που την χρησιμοποίησα είναι ώστε να εκτελέσω τον κώδικα των χρηστών μέσα στο πρόγραμμα περιήγησης τους. Η εκτέλεση μέσα στο πρόγραμμα περιήγησης αντί για την εκτέλεση στον server επιλέχθηκε ώστε οι χρήστες να έχουν τη δυνατότητα να γράψουν ότι κώδικα θέλουν χωρίς όμως να κινδυνεύει ο server από κακόβουλους χρήστες.

3.2.7 Σύνοψη

Συνοπτικά το framework Django χρησιμοποιήθηκε για το back-end της εφαρμογής, οι PostgreSQL για την αποθήκευση στοιχείων απαραίτητα για την εφαρμογή

και Bootstrap, Javascript και πιο συγκριμένα η βιβλιοθήκη Brython για την ανάπτυξη του front-end.

3.3 Ανάπτυξη Εφαρμογής Ιστού Με Django

Σε αυτό το κεφάλαιο θα γίνει περιγραφή της ανάπτυξης μιας εφαρμογής με Django. Όλες οι οδηγίες υποθέτουν ότι η ανάπτυξη γίνεται σε Windows και ότι έχουν εγκατασταθεί η Python και το Django. Επίσης οι οδηγίες ισχύουν για την έκδοση 3.6 της Python και 1.11.4 για το Django.

Ένα project φτιαγμένο με Django αποτελείται από εφαρμογές (apps), οπότε στη συνέχεια ο όρος project θα αναφέρεται στην εφαρμογή ιστού ως σύνολο και ο όρος app στις ύποεφαρμογες από τις οποίες αποτελείται.

3.3.1 Δημιουργία Project

Ένα Django project ξεκινάει με την εντολή:

- `django-admin startproject pyGame`

Το αποτέλεσμα είναι το εξής:

```
pyGame/
    manage.py
    pyGame/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Το pyGame/ είναι το root του project, το manage.py είναι ένα αρχείο που αποτελεί εντολή κονσόλας και μας επιτρέπει να ξεκινήσουμε τον test server, να δημιουργήσουμε πίνακες στη βάση δεδομένων και άλλους τρόπους αλληλεπίδρασης με το back-end του project.

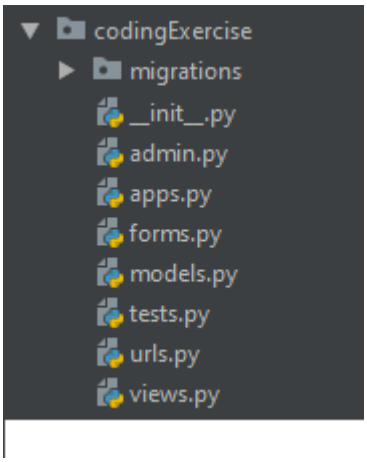
Το __init__.py είναι ένα κενό αρχείο που λέει ότι ο φάκελος είναι ένα Python package, το settings.py περιέχει τις ρυθμίσεις για το site μας, το wsgi.py αποτελεί ένα σημείο εισόδου για τους Web servers οι οποίοι είναι συμβατοί με το WSGI specification.

Τέλος το urls.py θα περιέχει όλα τα url που θα χρησιμοποιεί το site μας.

3.3.2 Δημιουργία Ενός App

Ένα Django project ξεκινά με την εντολή "startapp" που δημιουργεί τα απαραίτητα αρχεία (Κώδικας 3-1):

- `python manage.py startapp codingExcercise`



Εικόνα 3-1: Δομή μιας Υποεφαρμογής

Στις επόμενες ενότητες παρουσιάζονται τα βασικά στοιχεία μιας υποεφαρμογής.

3.3.2.1 Προβολές (Views)

Το Django χρησιμοποιεί το επίπεδο των προβολών για την επεξεργασία των αιτημάτων των χρηστών και την επιστροφή απαντήσεων. Οι προβολές δέχονται τα αιτήματα των χρηστών (HTTP requests) και στέλνουν πίσω την κατάλληλη απάντηση σύμφωνα με την λογική που εμπεριέχουν. Η απάντηση μπορεί να είναι μια σελίδα HTML, κάποιος κωδικός σφάλματος, μια εικόνα ή ένα αρχείο JSON ή XML.

Το πρώτο βήμα είναι να δημιουργήσουμε τις προβολές (views) μέσα στο αρχείο views.py. Οι προβολές καλούνται όταν κάποιος επισκέπτεται μια σελίδα του ιστοτόπου μας και στέλνουν πίσω τον HTML κώδικα με τον οποίο εμφανίζεται η σελίδα. Υπάρχουν

οι προβολές βασισμένες σε μεθόδους (function-based views) (Κώδικας 3-1) και οι προβολές βασισμένες σε κλάσεις (class-based views) (Κώδικας 3-2), τα πρώτα είναι εύκολα στην κατανόηση αλλά είναι δύσκολο να επεκταθούν ή να ξαναχρησιμοποιηθούν ενώ το αντίθετο ισχύει για τα δεύτερα.

Όπως μπορεί να καταλάβει κανείς από το όνομα τους οι προβολές βασισμένες σε μεθόδους περιέχουν τον κώδικα που ορίζει τη λειτουργικότητα μιας σελίδας μέσα σε μία μέθοδο, αυτό μπορεί να κάνει μια τέτοια προβολή μεγάλη σε μέγεθος και να οδηγήσει σε επανάληψη κώδικα. Το θετικό όμως στις προβολές βασισμένες σε μεθόδους είναι ότι δεν υπάρχει κρυμμένη λειτουργικότητα που θα πρέπει να ψάχνει κανείς αν κληθεί να δουλέψει πάνω σε κάποια έτοιμη προβολή.

Αντίθετα οι προβολές βασισμένες σε κλάσεις επιτρέπουν την οργάνωση του κώδικα μέσα σε κλάσεις και έχουν το πλεονέκτημα της κληρονομικότητας. Το αρνητικό είναι ότι ο προγραμματιστής πρέπει να έχει γνώση των κλάσεων από τις οποίες κληρονομούν οι προβολές του αλλά η χρήση όψεων βασισμένων σε κλάσεις οδηγεί σε πιο καλά οργανωμένο κώδικα χωρίς επαναλήψεις.

Στον Κώδικα 3-1 βλέπουμε την πιο απλή μορφή που μπορεί να έχει μια προβολή, η οποία δέχεται ένα αίτημα και απαντάει στέλνοντας μια απλή σελίδα HTML στην οποία γράφει μόνο “Hello, World!”.

```
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, world!")
```

Κώδικας 3-1: Προβολή Βασισμένη σε Μέθοδο

Το Django παρέχει “GenericViews” τα οποία μπορούν να κληρονομηθούν και παρέχουν έτοιμη λειτουργικότητα για καταστάσεις που εμφανίζονται συχνά όπως για παράδειγμα η προβολή μιας λίστας. Οι προβολές της υποεφαρμογής codingExcercise όπως και των υπολοίπων υποεφαρμογών είναι βασισμένες σε κλάσεις.

```python
class CodingExerciseListView(LoginRequiredMixin, ListView):
    model = Chapter
    template_name = 'codingExercise/exercise_list.html'

    def get_context_data(self, **kwargs):
        context = super(CodingExerciseListView, self).get_context_data(**kwargs)
        profileObject = Profile.objects.get(user__username=self.request.user)
        completed_exercises = profileObject.getCompletedExercises(True)
        context['completed_exercises'] = completed_exercises
        return context
```

Κώδικας 3- 2: Προβολή Βασισμένη σε Κλάση

Η προβολή CodingExcerciseListView (Κώδικας 3-2) είναι υπεύθυνη για να δείχνει τη λίστα με όλες τις προγραμματιστικές ασκήσεις (Εικόνα 4-6), άρα ανήκει στις περιπτώσεις που μπορούμε να χρησιμοποιήσουμε το Generic View "ListView" που μας παρέχει το Django. Το μόνο που χρειάζεται να κάνουμε εμείς είναι να συσχετίσουμε την προβολή με το μοντέλο που θέλουμε. Στην περίπτωση του CodingExcerciseListView αυτό είναι το Chapter, το μοντέλο που ορίζει τι πληροφορίες χρειάζονται για τα αντικείμενα που αντιπροσωπεύουν τα κεφάλαια στα οποία είναι χωρισμένη η διδακτική ύλη, και το πρότυπο (template), ένα HTML αρχείο που ορίζει τι παρουσιάζεται στον χρήστη (Κώδικας 3-6). Τα μοντέλα και τα πρότυπα αναλύονται σε παρακάτω ενότητα. Τέλος, υποσκελίζουμε τη μέθοδο get_context_data ώστε να προσθέσουμε όποιες πληροφορίες θέλουμε στο context, ένα λεξικό μέσω του οποίου μπορούμε να έχουμε πρόσβαση σε μεταβλητές του μοντέλου μέσα στο πρότυπο. Πρώτα καλούμε την super για να πάρουμε το αρχικό context αφού στόχος είναι να προστεθεί πληροφορία και όχι να διαγραφεί ότι ήδη περιέχει το context, ύστερα ψάχνουμε στη βάση δεδομένων το προφίλ του χρήστη που έχει κάνει το αίτημα και παίρνουμε τη λίστα με τις ασκήσεις που έχει ολοκληρώσει και την προσθέτουμε στο context. Στη περίπτωση του CodingExcerciseListView το κάνουμε ώστε να μπορεί να βλέπει την κατάλληλη ένδειξη δίπλα σε κάθε άσκηση ο χρήστης και να γνωρίζει τι του έχει απομείνει να ολοκληρώσει.

3.3.2.2 Διευθύνσεις URL

Έπειτα οι προβολές μιας υποεφαρμογής πρέπει να συσχετιστούν με μια διεύθυνση URL ώστε ο χρήστης να μπορεί να επισκεφτεί τις διαφορετικές σελίδες της

εφαρμογής. Όταν ο χρήστης επισκέπτεται μια από αυτές τις διευθύνσεις στέλνει στην εφαρμογή ένα αίτημα που όπως αναφέρθηκε στην προηγούμενη ενότητα η αντίστοιχη προβολή θα διαχειριστεί και θα επιστρέψει την κατάλληλη απάντηση.

Η συσχέτιση γίνεται μέσα στο αρχείο urls.py (Κώδικας 3-3) της υποεφαρμογής και ύστερα πρέπει να εισάγουμε αυτό το αρχείο στο αρχείο urls.py που δημιουργήθηκε κατά την αρχικοποίηση του project (Κώδικας 3-4). Όλες οι διευθύνσεις URL καταγράφονται σε μια λίστα με το όνομα "urlpatterns". Αν υποθέσουμε ότι η διαδρομή της βασικής σελίδας είναι "www.domain.com" τότε συμφώνα με τις κανονικές εκφράσεις που παρουσιάζονται στους Κώδικες 3-3 και 3-4 για να δει κάποιος τη λίστα με τις ασκήσεις πρέπει να επισπευτεί τη διαδρομή "www.domain.com/ce/".

```
from django.conf.urls import url


from .views import CodingExerciseListView, CodingExerciseDetailView, Finished


app_name = 'codingExercise'
urlpatterns = [
    url(r'^finished$', Finished.as_view(), name='finished'),
    url(r'^$', CodingExerciseListView.as_view(), name='exercise_list'),
    url(r'^(?P<pk>[0-9]+)$', CodingExerciseDetailView.as_view(), name='exercise'),

]
```

Κώδικας 3- 3: Οι Διευθύνσεις URL μιας Υποεφαρμογής

```
from django.conf.urls import url, include
from django.contrib import admin
from django.views.generic import TemplateView

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^ce/', include('codingExercise.urls')),
    url(r'^account/', include('account.urls')),
    url(r'^quiz/', include('quiz.urls')),
    url(r'^example/', include('examples.urls')),
    url(r'^leaderboard/', include('leaderboard.urls')),
    url(r'^', TemplateView.as_view(template_name='home.html'), name='home'),
    url(r'^badges/', include('badges.urls')),

]
```

Κώδικας 3-4: Το Κεντρικό Αρχείο urls.py της Εφαρμογής

3.3.2.3 Μοντέλα (Models)

Τα μοντέλα είναι ο μοναδικός τρόπος με τον οποίο μπορούμε να ανακτήσουμε δεδομένα σε μια εφαρμογή Django. Τα μοντέλα ορίζουν τη δομή της βάσης δεδομένων μας, με κάθε κλάση να αντιστοιχεί σε έναν πίνακα και κάθε ιδιότητα της στις στήλες του πίνακα. Επίσης περιέχουν τη συμπεριφορά των δεδομένων.

Το app codingExcercise έχει δύο μοντέλα το Chapter και το Exercise (Κώδικας 3-5).

```python
from django.db import models


# Create your models here.

class Chapter(models.Model):
    name = models.TextField(max_length=100)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['id']


class Exercise(models.Model):
    chapter = models.ForeignKey(Chapter, on_delete=models.CASCADE, blank=True, null=True)
    name = models.CharField(max_length=100, null=False, blank=False, )
    instructions = models.TextField(null=False, blank=False)
    test_imports = models.TextField(blank=True)
    test = models.TextField(null=False, blank=False)
    text = models.TextField(null=True, blank=True)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['id']
```

Κώδικας 3-5: Τα μοντέλα του codingExcercise

Ο πίνακας Chapter θα έχει μόνο μία στήλη το όνομα του κεφαλαίου, ενώ ο πίνακας Exercise περιέχει το κεφάλαιο στο οποίο ανήκει η άσκηση ως ξένο κλειδί και το όνομα, την περιγραφή, τις οδηγίες της ασκήσης καθώς και τον κώδικα που χρησιμοποιείται για τον έλεγχο σωστής λύσης.

3.3.2.4 Πρότυπα (Templates)

Τα πρότυπα είναι υπεύθυνα για την προβολή της εφαρμογής στον χρήστη. Αποτελούνται από κώδικα HTML, CSS και Javascript και είναι αρχεία html. Είναι ιδιαίτερα χρήσιμα επειδή μπορούμε να τους περάσουμε πληροφορίες μέσω του context από τις προβολές και είναι επεκτάσιμα, μπορούμε δηλαδή να χρησιμοποιήσουμε ένα βασικό πρότυπο που ο κώδικας του είναι απαραίτητος σε όλες τις σελίδες του ιστοτόπου, π.χ. ένα μενού, χωρίς να επαναλαμβάνουμε τον ίδιο κώδικα HTML σε κάθε αρχείο. Τέλος, μας δίνουν τη δυνατότητα χρήσης δομών επιλογής (if) και επανάληψης (for) κάνοντας έτσι δυνατή τη δυναμική καταχώρηση του περιεχομένου μιας σελίδας. Στο

παράδειγμα του Κώδικα 3-6 βλέπουμε το πρότυπο για τη σελίδα με τη λίστα όλων ασκήσεων (Εικόνα 4-6). Αρχικά βλέπουμε ότι επεκτείνει (extends) το περιεχόμενο ενός άλλου προτύπου, "base.html", που περιέχει κώδικα απαραίτητο σε όλες τις σελίδες του ιστοτόπου. Το περιεχόμενο του συγκεκριμένου προτύπου θα μπει στη θέση που ορίζεται από το "block" με το όνομα "content" στο "base.html" ώστε να γίνει η σωστή παρουσίαση της σελίδας. Για κάθε κεφάλαιο εμφανίζεται ο τίτλος του και ο τίτλος κάθε άσκησης που ανήκει σε αυτό αν υπάρχουν ασκήσεις, διαφορετικά εμφανίζεται ένδειξη ότι δεν υπάρχουν ασκήσεις. Δίπλα σε κάθε άσκηση υπάρχει ένδειξη αν έχει λυθεί ή όχι.

```
{% extends 'base.html' %}

{% block content %}


    <div class="container">
        {% for chapter in object_list %}
            <div class="chapter-container">
                <h1 class="card-header bg-dark text-white text-center "> {{ chapter }}</h1>
                <ul class="list-group">

                    {% for exercise in chapter.exercise_set.values %}
                        <li class="list-group-item">
                            {% if exercise.name in completed_exercises %}
                                <i class="fa fa-check-circle-o fa-2x" aria-hidden="true" style="..."></i>
                            {% else %}

                                <i class="fa fa-question-circle-o fa-2x" aria-hidden="true" style="..."></i>
                            {% endif %}

                            <a href="{% url 'codingExercise:exercise' pk=exercise.id %}">{{ exercise.name }} </a>

                        </li>




                    {% empty %}
                        <li>No exercises yet.</li>
                    {% endfor %}

                </ul>
            </div>
        {% endfor %}

    </div>

{% endblock %}
```

Κώδικας 3-6: Το Πρότυπο του CodingExcerciseListView

3.3.2.5 Βάση Δεδομένων (Database)

Το Django ώς framework είναι αγνωστικό όσον αφορά τις βάσεις δεδομένων στον πυρήνα του και ενθαρρύνει τη δημιουργία αγνωστικού κώδικα ώστε να είναι

επαναχρησιμοποιήσιμος χωρίς να έχει σημασία η επιλογή βάσης δεδομένων. Αυτό δεν σημαίνει βέβαια ότι δεν προσφέρει τη δυνατότητα χρήσης δυνατοτήτων που υπάρχουν μόνο σε μια τεχνολογία βάσεων δεδομένων και όχι σε κάποια άλλη αφού αυτές μπορεί να είναι κρίσιμες για κάποια έργα.

Το Django υποστηρίζει επίσημα τη χρήση τεχνολογιών βάσεων δεδομένων sqlite, PostgreSQL, MySQL, Oracle.

Για να χρησιμοποιήσει κάποιος μια από αυτές τις τεχνολογίες πρέπει να αλλάξει τις ρυθμίσεις σχετικά με ποιο "back-end" χρησιμοποιεί η εφαρμογή του στο αρχείο "settings.py" που φαίνεται στην Εικόνα 3-1. Αν για παράδειγμα θέλει να χρησιμοποιήσει sqlite πρέπει να αλλάξει τις ρυθμίσεις όπως φαίνεται στον Κώδικα 3-7. Αντίθετα αν θέλει να κάνει χρήση PostgreSQL απαιτούνται οι ρυθμίσεις του Κώδικα 3-8.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, '../db3.sqlite3'),
    }
}
```

Κώδικας 3-7: Ρυθμίσεις για Χρήση sqlite

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'myproject',
        'USER': 'myprojectuser',
        'PASSWORD': 'password',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

Κώδικας 3-8: Ρυθμίσεις για Χρήση PostgreSQL

Όπως αναφέρεται παραπάνω η δομή των βάσεων δεδομένων ορίζεται από τα μοντέλα. Αφού λοιπόν δημιουργηθούν τα μοντέλα και η βάση δεδομένων μας μπορούμε να δημιουργήσουμε και να εφαρμόσουμε τα "migrations" ώστε να δημιουργηθούν οι πίνακες. Πρέπει να σημειωθεί εδώ ότι για να γίνουν τα "migrations" πρέπει η υποεφαρμογή να είναι στην λίστα "INSTALLED_APPS" που βρίσκεται στο αρχείο "settings.py" (Κώδικας 3-9).

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'codingExercise',
    'django_ace',
    'account',
    'quiz',
    'examples',
    'leaderboard',
    'badges',
]
```

Κώδικας 3-9: Λίστα με Εγκατεστημένες Εφαρμογές

3.3.3 Περιγραφή των Υποεφαρμογών (Apps)

Σε αυτή την ενότητα αναφέρονται οι προβολές και τα μοντέλα όλων των υποεφαρμογών που απαρτίζουν το project.

3.3.3.1 CodingExercise App

Το CodingExcercise app περιέχει τη λειτουργικότητα που χρειάζεται για τη διαχείριση των προγραμματιστικών ασκήσεων. Σε αυτό ανήκουν όλες οι προβολές, πρότυπα και url που έχουν να κάνουν με τις προγραμματιστικές ασκήσεις της εφαρμογής. Είναι υπεύθυνο για την προβολή της λίστας με όλες τις ασκήσεις και κάθε μεμονωμένης άσκησης. Επίσης περιλαμβάνει τη λειτουργικότητα για την ενημέρωση της βάσης δεδομένων σε περίπτωση ολοκλήρωσης κάποιας άσκησης από έναν χρήστη.

Οι προβολές του codingExercise είναι:

- CodingExerciseListView: το μοντέλο της είναι το Chapter και η μόνη μέθοδος που έχει είναι η get_context_data μέσω της οποία προστίθεται στο context η λίστα με τις ασκήσεις που έχει ολοκληρώσει ο χρήστης. Είναι υπεύθυνη για την προβολή της σελίδας με τη λίστα όλων των ασκήσεων.

- CodingExerciseDetailView: το μοντέλο του είναι το Exercise και περιέχει τις μεθόδους get και post ώστε να απαντά στα αιτήματα των χρηστών. Σε περίπτωση get request επιστρέφει τη σελίδα με την άσκηση εφόσον αυτή υπάρχει, διαφορετικά επιστρέφει μήνυμα ότι η σελίδα που προσπαθεί να επισκευτεί ο χρήστης δεν υπάρχει. Σε περίπτωση post request κάτι που συμβαίνει μόνο στην περίπτωση που ο χρήστης δώσει σωστή λύση στην άσκηση γίνεται έλεγχος αν πρόκειται για νέα μοναδική λύση, αν ο χρήστης δεν έχει υποβάλει την ίδια λύση αυτολεξεί στο παρελθόν, και την προσθέτει στο προφίλ του χρήστη και αν είναι η πρώτη του λύση του προσθέτει πόντους εμπειρίας και τον επιστρέφει στην λίστα με τις ασκήσεις. Τέλος η μέθοδος get_context_data προσθέτει απαραίτητες πληροφορίες στο Template.

  Ο επεξεργαστής κειμένου που ενσωματώθηκε είναι ο "Django-Ace-Editor", μια υλοποίηση του "Ace Editor" ως "widget" που μπορεί να εισαχθεί σε μια φόρμα. Ο κώδικας που υποβάλει ο χρήστης ελέγχεται με τεστ που βρίσκονται αποθηκευμένα στη βάση δεδομένων. Όπως φαίνεται στον Κώδικα 3-10 που είναι το "script" που τρέχει όταν ο χρήστης υποβάλει τη λύση του παίρνουμε το τεστ από το "context" και το «καθαρίζουμε» επειδή στη βάση δεδομένων τα εισαγωγικά απεικονίζονται ως "&quot" και στη συνέχεια μέσα σε μια μέθοδο ενώνουμε τον κώδικα του τεστ και τον κώδικα του χρήστη και τον εκτελούμε όταν ο χρήστης πατήσει το κουμπί. Αν η λύση είναι σωστή τυπώνεται στην κονσόλα το μήνυμα "Correct" και εμφανίζεται η επιλογή να προχωρήσει παρακάτω.

  Τα τεστ αυτά δοκιμάζουν σενάρια με διαφορετικές εισόδους και αν τα αποτελέσματα που παράγει ο κώδικας του χρήστη συμπίπτουν με τα αναμενόμενα αποτελέσματα η λύση γίνεται δεκτή. Στον Κώδικα 3-11 φαίνεται ο κώδικας που χρησιμοποιείται για τον έλεγχο μιας άσκησης στην οποία ο χρήστης καλείται να δημιουργήσει μια μέθοδο που δέχεται δύο λίστες και επιστρέφει μια λίστα με τα κοινά τους στοιχεία. Πρώτα ελέγχεται αν υπάρχει η μέθοδος με το όνομα που ορίζεται στην εκφώνηση και στην συνέχεια ελέγχονται τρία σενάρια και αν η απάντηση είναι σωστή και για τα τρία τυπώνεται στην κονσόλα το μήνυμα "Correct" ενώ αν αποτύχει σε κάποιο από αυτά τυπώνεται το αναμενόμενο αποτέλεσμα και το αποτέλεσμα που παράγει η λύση του χρήστη για καθένα από αυτά. Κάθε σωστή λύση που υλοποιεί ο

χρήστης αποθηκεύεται στη βάση δεδομένων και παρουσιάζεται στην λίστα με τις υπόλοιπες σωστές λύσεις του χρήστη στη σελίδα της εκάστοτε άσκησης. Κάθε χρήστης μπορεί να δει μόνο τις δικές του λύσεις.

Ο κώδικας του χρήστη και το τεστ εκτελούνται στο πρόγραμμα περιήγησης του χρήστη και όχι στον διακομιστή. Αυτό επιτυγχάνεται με τη χρήση της βιβλιοθήκης Brython.js που κάνει δυνατή τη χρήση Python αντί Javascript ως τη γλώσσα προγραμματισμού για Web. Είναι μια υλοποίηση της Python προσαρμοσμένη στο περιβάλλον HTML5 και μας δίνει τη δυνατότητα να τροποποιήσουμε αντικείμενα μέσα σε μια σελίδα HTML αλλά είναι ικανή να τρέξει τα απλά προγράμματα που ζητείται να υλοποιήσουν οι χρήστες καθώς και τα τεστ που έχουν δημιουργηθεί για τον έλεγχο τους.

- Finished: ένα απλό πρότυπο βασισμένο στο TemplateView που ενημερώνει τον χρήστη ότι έχει ολοκληρώσει όλες τις ασκήσεις.

```
<!--Run the test-->
<script type="text/python">
    from browser import document, html

    #get test from context
    test="""{{ ex.test }}"""
    test_imports="""{{ ex.test_imports }}"""

    #fix the test string
    test= test.replace("&quot;","'")
    test_imports= test_imports.replace("&quot;","'")

    def run(ev):
        document['console'].clear()
        #get user code and test code and append them
        user_code = document["id_text"].value
        myCode = test_imports
        myCode += "\n"
        myCode += user_code
        myCode += "\n"
        myCode += test
        #run the code
        exec(myCode)
        #if the test is passed allow the user to progress
        if "Correct" in  document['console'].text and "Correct" not in user_code:
            document['success-bar'].style.display='block'

    #bind the run function the the button runCode
    document['runCode'].bind('click',run)
</script>
```

Κώδικας 3-10: Κώδικας για την Εκτέλεση Τεστ

```
def test_list_overlap():
    try:
        list_overlap
        if list_overlap([],[1,2]) == [] and list_overlap([1,3,1],[1,2]) == [1] and list_overlap([1,3,2,1],[1,2]) == [1,2]:
            print("Correct")
        else:
            print("Your funtion doesn't appear to return the correct result")
            print("Expected result for input [],[1,2]  is []  and " + str(list_overlap([],[1,2])) + " was returned")
            print("Expected result for input [1,3,1],[1,2]  is [1]  and " + str(list_overlap([1,3,1],[1,2])) + " was returned")
            print("Expected result for input([1,3,2,1],[1,2]  is [1,2]  and " + str(list_overlap([1,3,2,1],[1,2])) + " was returned")
    except NameError:
        print("You haven't declared a function named swap")


test_list_overlap()
```

Κώδικας 3-11: Παράδειγμα Τεστ

Τα μοντέλα του codingExercise είναι:

- Chapter που αντιπροσωπεύει τα κεφάλαια στα οποία είναι χωρισμένη η διδακτική ύλη με μοναδική ιδιότητα name (TextField).
- Exercise που αντιπροσωπεύει τις προγραμματιστικές ασκήσεις με ιδιότητες:
    - chapter (Chapter) που είναι foreign key
    - order (IntegerField)
    - name (CharField)
    - instructions (TextField)
    - test_imports (TextField)
    - test (TextField)
    - text (TextField)

3.3.3.2 Examples App

Το Examples app περιέχει τη λειτουργικότητα που χρειάζεται για τη διαχείριση των παραδειγμάτων κώδικα. Σε αυτό ανήκουν όλες οι προβολές, πρότυπα και url που έχουν να κάνουν με τα παραδείγματα κώδικα της εφαρμογής. Είναι υπεύθυνο για την προβολή της λίστας με τα παραδείγματα κώδικα και κάθε μεμονωμένου παραδείγματος.

Οι προβολές του examples είναι:

- ExampleListView: το μοντέλο της είναι το Chapter και δεν έχει μεθόδους. Είναι υπεύθυνη για την προβολή της σελίδας με τη λίστα όλων των παραδειγμάτων κώδικα.
- ExampleDetailView: το μοντέλο της είναι το Example και περιέχει τη μέθοδο post ώστε να απαντά στα αιτήματα των χρηστών. Σε περίπτωση post request κάνει προώθηση στην επόμενη σελίδα εφόσον αυτή υπάρχει, διαφορετικά ενημερώνει τον χρήστη ότι έχει ολοκληρώσει όλη τη θεωρία, δεν υπάρχει κάποια προβολή σε αυτή την υποεφαρμογή για την λειτουργία αυτή αφού μπορεί να επαναχρησιμοποιηθεί η προβολή Finished από το CodingExercise.

Τα μοντέλα του examples είναι:

- Example που αντιπροσωπεύει τα παραδείγματα θεωρίας με ιδιότητες:
  - chapter (Chapter) που είναι foreign key
  - title (CharField)
  - content (TextField)

3.3.3.3 Quiz App

Το Quiz app περιέχει τη λειτουργικότητα που χρειάζεται για τη διαχείριση των κουιζ. Σε αυτό ανήκουν όλες οι προβολές, πρότυπα και url που έχουν να κάνουν με τα παραδείγματα κώδικα της εφαρμογής. Είναι υπεύθυνο για την προβολή της λίστας με όλα τα κουιζ και κάθε μεμονωμένου κουιζ. Επίσης περιλαμβάνει τη λειτουργικότητα για την ενημέρωση της βάσης δεδομένων σε περίπτωση ολοκλήρωσης κάποιου κουίζ από έναν χρήστη και για την παρουσίαση των αποτελεσμάτων.

Οι προβολές του Quiz είναι:

- QuizListView: το μοντέλο της είναι το Chapter και η μόνη μέθοδος που έχει είναι η get_context_data μέσω της οποία προστίθεται στο context η λίστα με τα κουίζ που έχει ολοκληρώσει ο χρήστης. Είναι υπεύθυνη για την προβολή της σελίδας με τη λίστα όλων των κουίζ.

- QuizDetailView: το μοντέλο του είναι το Quiz και περιέχει τις μεθόδους get και post ώστε να απαντά στα αιτήματα των χρηστών. Σε περίπτωση get request επιστρέφει τη σελίδα με το κουίζ εφόσον αυτή υπάρχει, διαφορετικά επιστρέφει μήνυμα ότι η σελίδα που προσπαθεί να επισκευτεί ο χρήστης δεν υπάρχει. Σε περίπτωση post request γίνεται έλεγχος των απαντήσεων του χρήστη και αν είναι η πρώτη του λύση του προσθέτει πόντους εμπειρίας με βάση το ποσοστό σωστών απαντήσεων και κάνει προώθηση στην σελίδα με τα αποτελέσματα. Είναι υπεύθυνη για την προβολή της σελίδας με το κουίζ.

Τα μοντέλα του quiz είναι:

- Quiz που αντιπροσωπεύει ένα τεστ ερωτήσεων πολλαπλής επιλογής με ιδιότητες:
  - chapter (Chapter) που είναι foreign key
  - name (CharField)
- Question που αντιπροσωπεύει μια ερώτηση πολλαπλής επιλογής και τις απαντήσεις της με ιδιότητες:
  - quiz (Quiz) που είναι foreign key
  - question_text (CharField)
  - answer1 (CharField)
  - answer2 (CharField)
  - answer3 (CharField)
  - correct_answer (CharField)

3.3.3.4 Leaderboard App

Η υποεφαρμογή Leaderboard περιέχει τη λειτουργικότητα που χρειάζεται για τη διαχείριση του πίνακα κατάταξης. Σε αυτό ανήκουν όλες οι προβολές, πρότυπα και url που έχουν να κάνουν με τον πίνακα κατάταξης. Είναι υπεύθυνο για την προβολή του πίνακα κατάταξης.

Οι προβολές του leaderboard είναι:

- LeaderBoardListView: το μοντέλο της είναι το Profile και έχει μία μέθοδο get_query_set που χρησιμοποιείται για να γίνει ταξινόμηση των χρηστών με βάση τους πόντους που έχουν.

Το app αυτό δεν έχει μοντέλα αφού χρησιμοποιεί μόνο το μοντέλο profile που ανήκει στην υποεφαρμογή Account.

3.3.3.5 Account App

Η υποεφαρμογή Account περιέχει τη λειτουργικότητα που χρειάζεται για τη διαχείριση των προφίλ των χρηστών, της σελίδας προόδου τους και τις φόρμες εγγραφής και σύνδεσης στην ιστοσελίδα. Ο σκοπός αυτής της εφαρμογής είναι να επεκτείνει τη λειτουργικότητα της κλάσης User, το μοντέλο που αντιπροσωπεύει ένα χρήστη στο Django, ώστε να μπορέσουμε να αποθηκεύσουμε πληροφορίες μοναδικές στην περίπτωση χρήσης μας όπως οι ολοκληρωμένες ασκήσεις ενός χρήστη ή τα επιτεύγματα που έχει πετύχει.

Οι προβολές του account είναι:

- RegisterUserView: κληρονομεί από το CreateView και δεν αντιστοιχίζεται με κάποιο μοντέλο αλλά με μία φόρμα. Έχει μια μέθοδο dispatch η οποία ελέγχει αν ο χρήστης δεν είναι ήδη συνδεδεμένος και μόνο τότε μπορεί να κάνει εγγραφή στην ιστοσελίδα και τη μέθοδο form_valid που ελέγχει ότι η φόρμα εγγραφής του χρήστη είναι έγκυρη. Είναι υπεύθυνη για την προβολή της φόρμας εγγραφής.
  - UserProfileView: το μοντέλο της είναι το Profile και η μόνη μέθοδος που έχει είναι η get_context_data για να προσθέτει τη λίστα με τις ολοκληρωμένες ασκήσεις στο context. Είναι υπεύθυνη για την προβολή των προφίλ των χρηστών.

- ProggressView: το μοντέλο της είναι το Profile και έχει τη μέθοδο get_context_data για να προσθέτει τη λίστα με τα τρόπαια που έχει κερδίσει ο χρήστης και μέθοδο get όπου ελέγχεται ότι ο χρήστης έχει εξουσιοδότηση να δει τη σελίδα. Οι χρήστες μπορούν να δουν μόνο τη δική τους πρόοδο. Το μοντέλο του είναι το Profile και έχει τη μέθοδο get_context_data για να προσθέτει απαραίτητες πληροφορίες στο Template και μέθοδο get όπου ελέγχεται ότι ο χρήστης έχει εξουσιοδότηση να δει τη σελίδα. Οι χρήστες μπορούν να δουν μόνο τη δική τους πρόοδο. Είναι υπεύθυνη για την προβολή των προφίλ των χρηστών. Είναι υπεύθυνη για την προβολή της σελίδας προόδου των χρηστών.

Τα μοντέλα του account είναι:

- Profile που αντιπροσωπεύει το προφίλ ενός χρήστη με ιδιότητες:
  - user που είναι OneToOne field και αντιστοιχεί κάθε αντικείμενο User (μοντέλο της Django, έτσι διαχειρίζεται τους χρήστες) με το Profile
  - points (Integer)
  - totalQuizScore (Float)
  - quizAvgScore (Float)
  - num_of_competed_exercises (Integer)
  - num_of_competed_quizes (Integer)
  - num_of_achievements (Integer)

- CompletedQuiz που συσχετίζει το προφίλ ενός χρήστη με ένα τέστ ερωτήσεων πολλαπλής επιλογής που έχει ολοκληρώσει με ιδιότητες:
  - userProfile (Profile) που είναι foreign key
  - quiz_name (CharField)
  - score (CharField)
  - date (DateTimeField)

- CompletedExercise που συσχετίζει το προφίλ ενός χρήστη με μια προγραμματιστική άσκηση που έχει ολοκληρώσει με ιδιότητες:
  - userProfile (Profile) που είναι foreign key

- exercise_name (CharField)
- answer (TextField)
- date (DateTimeField)

Η υποεφαρμογή Account περιέχει επίσης ένα αρχείο Python με το όνομα meta_badges.py μέσα στο οποίο ορίζονται τα επιτεύγματα που μπορεί να πετύχει ο χρήστης. Για τη δημιουργία των επιτευγμάτων χρησιμοποιήθηκε η εφαρμογή "django-badges" η οποία ορίζει πως για κάθε επίτευγμα ορίζεται ένα μοναδικό αναγνωριστικό, ο τίτλος, η περιγραφή, το επίπεδο δυσκολίας του, το μοντέλο με το οποίο αντιστοιχίζεται, αν κάποιος μπορεί να το αποκτήσει μόνο μία ή παραπάνω φορές και ο στόχος που πρέπει να πετύχει κάποιος για να το αποκτήσει. Κάθε επίτευγμα αντιπροσωπεύεται από μια κλάση (Κώδικας 3-12). Οι κλάσεις αυτές έχουν μια μέθοδο get_user που επιστρέφει ένα αντικείμενο τύπου User, μια μέθοδο get_progress που επιστρέφει την πρόοδο του χρήστη και μία μέθοδο check_points που επιστρέφει True ή False ανάλογα με το αν ο χρήστης έχει πετύχει τον στόχο.

```
class BasePointsCollector(MetaBadge):
    id = "BasePointsCollector"
    title = "Points Collector Base Class"
    description = "This is the Points Collector Base Class.DO NOT
CHANGE IT"
    level = "5"

    model = Profile
    one_time_only = True
    target = 1000000

    progress_start = 0
    progress_finish = target

    def get_user(self, instance):
        return instance.user

    def get_progress(self, user):
        progress = self.target if user.profile.points >= self.target
else user.profile.points
        return progress

    def check_points(self, instance):
        return instance.user.profile.points >= self.target


class NovicePointsCollector(BasePointsCollector):
    id = "NovicePointsCollector"

    title = "Novice Points Collector"
    description = "Collected 10 points"
    level = "1"
    target = 10
```

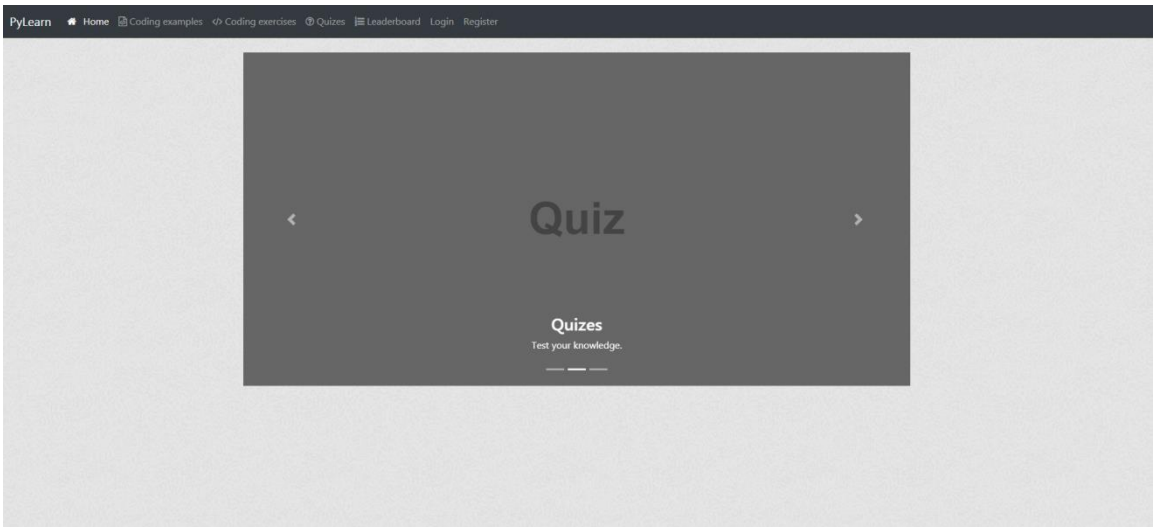
Κώδικας 3-12: Παράδειγμα Επιτεύγματός

4 Παρουσίαση Εφαρμογής

Μια εφαρμογή Django αποτελείται από δύο μέρη, ένα δημόσιο ιστότοπο μέσω του οποίου έχουν πρόσβαση η χρήστες στις λειτουργίες της εφαρμογής και ένα ιδιωτικό ιστότοπο για την διαχείριση της βάσης δεδομένων.

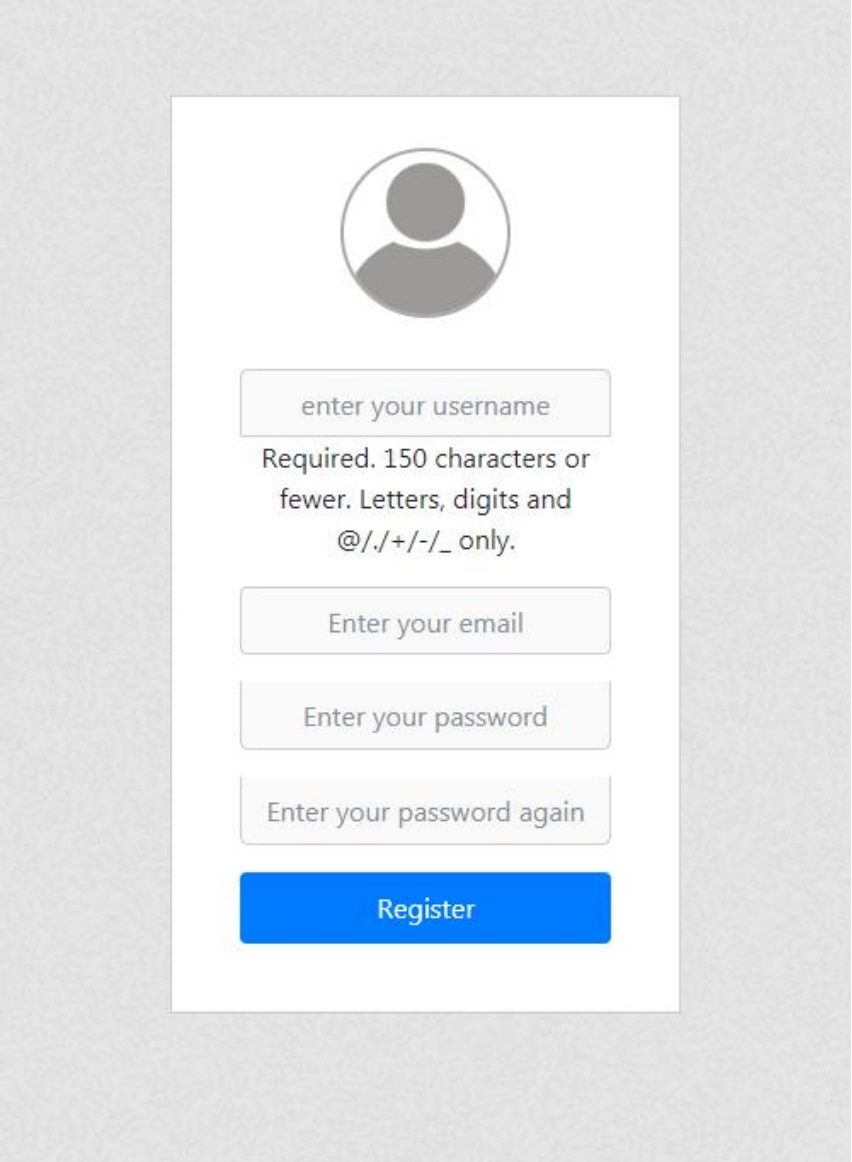
4.1 Λειτουργία για Χρήστες

Στην αρχική σελίδα ο χρήστης βλέπει το κεντρικό μενού από όπου μπορεί να περιηγηθεί στον ιστότοπο και ένα carousel με μια σύντομη περιγραφή του τι προσφέρει η εφαρμογή (Εικόνα 4-1).



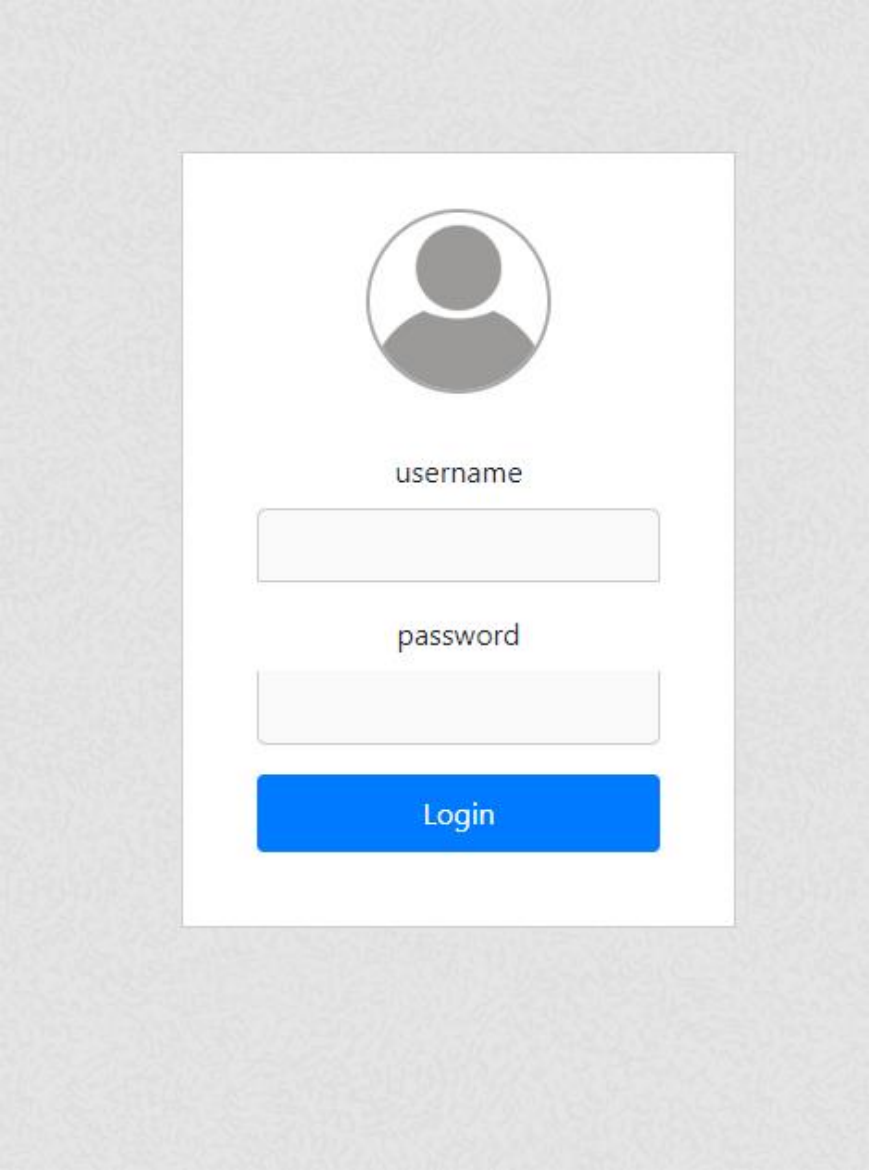
Εικόνα 4-1: Αρχική Οθόνη

Αρχικά ο χρήστης πρέπει να κάνει εγγραφή πατώντας στην επιλογή “Register” και συμπληρώνοντας τη φόρμα εγγραφής (Εικόνα 4–2).



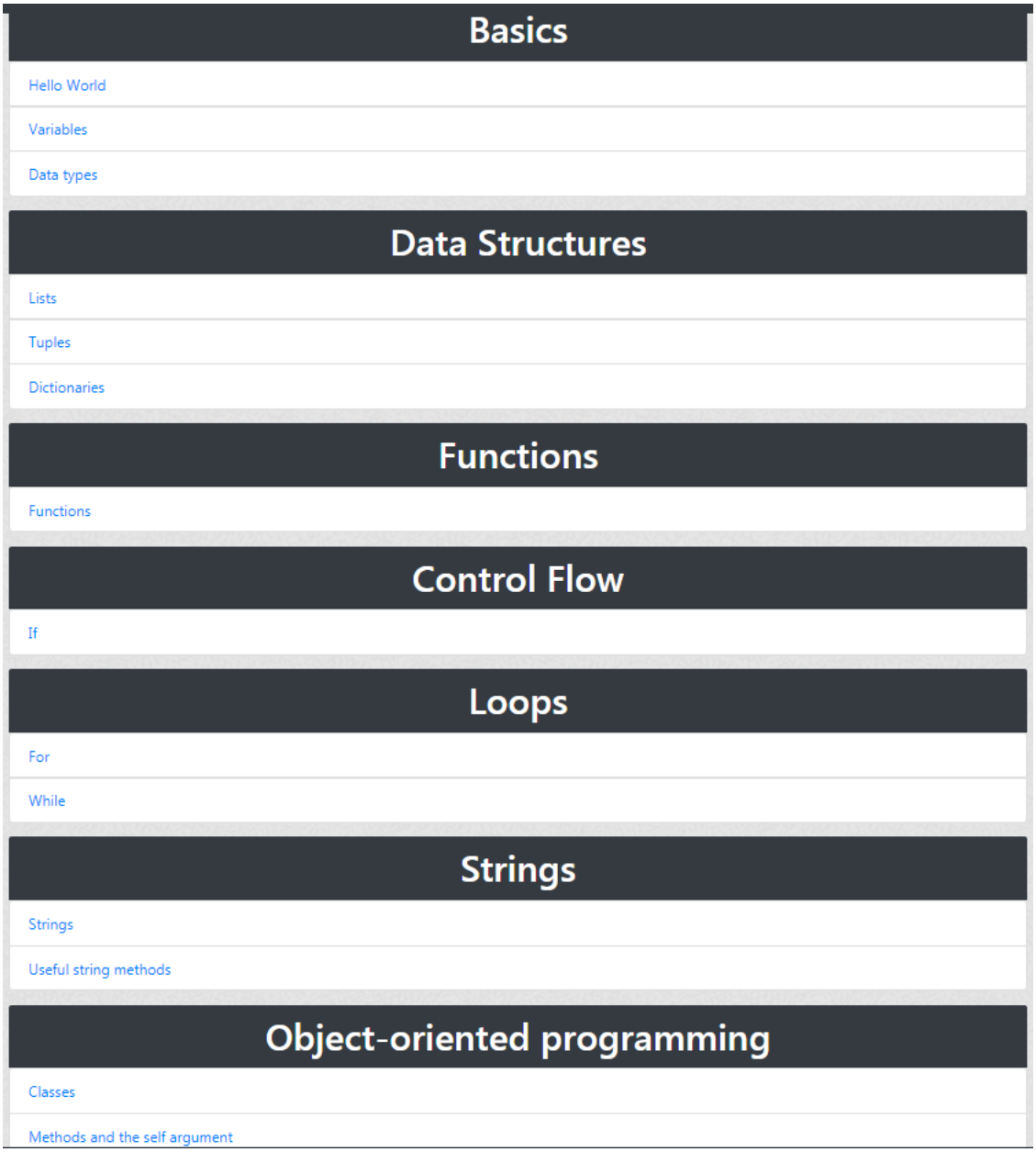
Εικόνα 4-2: Φόρμα Εγγραφής

Έπειτα από την επιλογή “Login” μπορεί να συνδεθεί με το όνομα χρήστη και τον κωδικό που όρισε κατά την εγγραφή (Εικόνα 4-3).

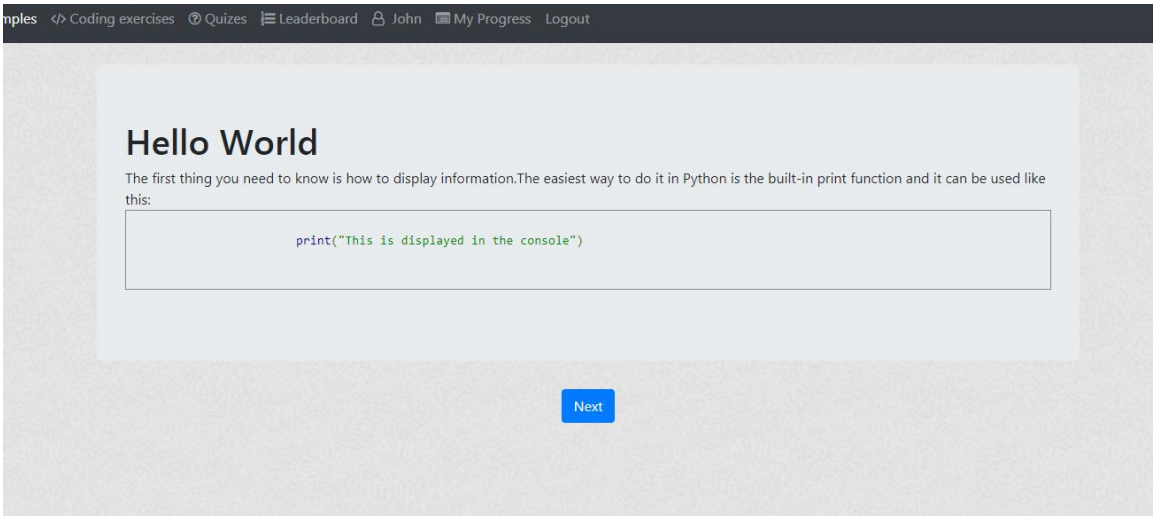


Εικόνα 4-3: Φόρμα Σύνδεσης

Αφού συνδεθεί επιτυχώς μπορεί να δει πλέον όλο το περιεχόμενο της εφαρμογής. Στα "Coding examples" υπάρχουν παραδείγματα κώδικα (Εικόνα 4-5) χωρισμένα σε θεματικές ενότητες (Εικόνα 4-4).

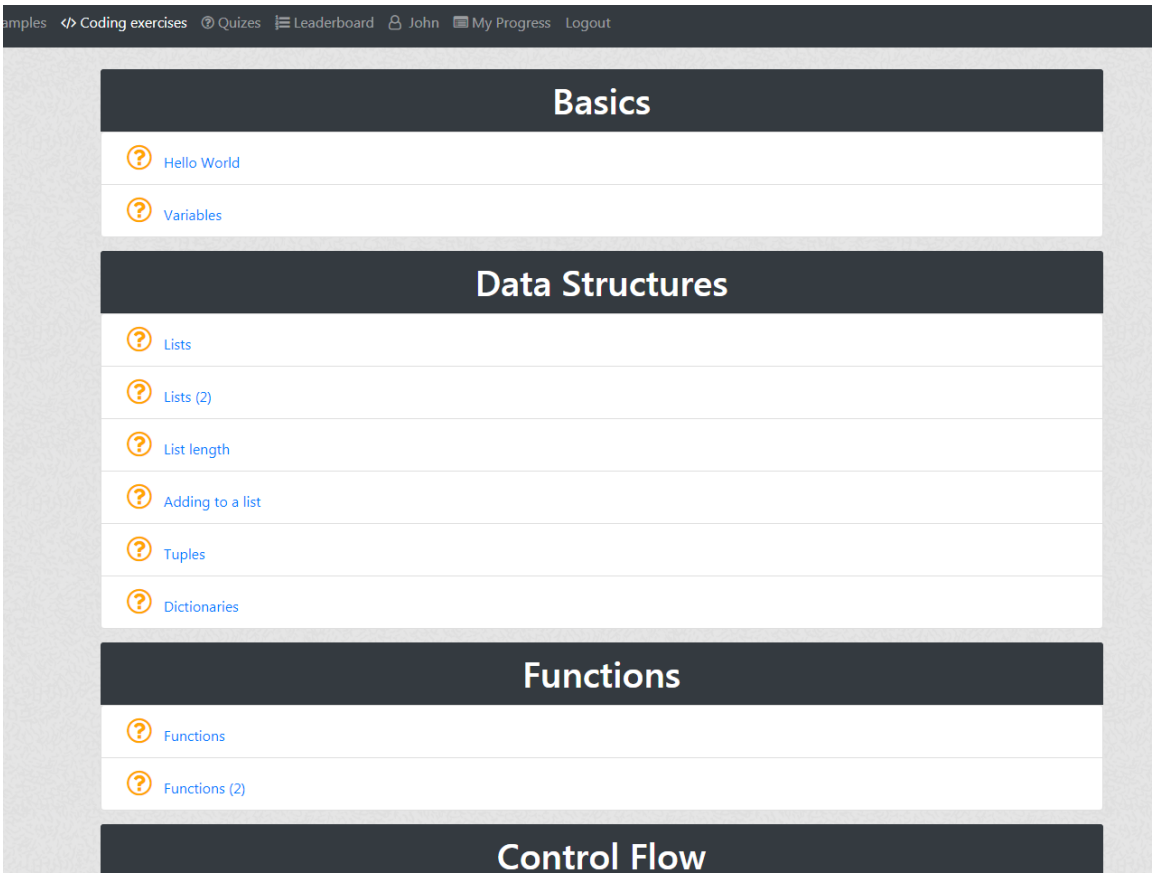


Εικόνα 4-4: Λίστα με Παραδείγματα Κώδικα



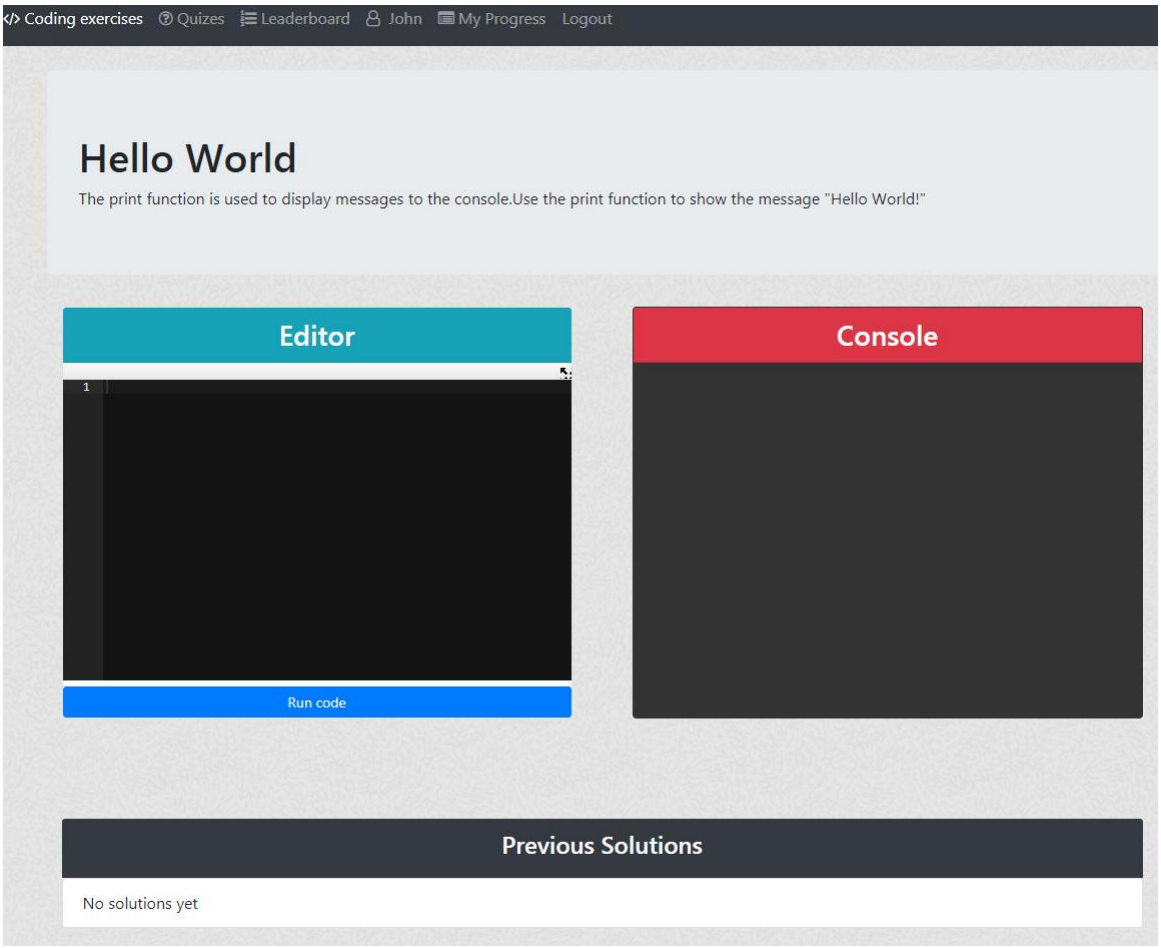
Εικόνα 4-5: Παράδειγμα Κώδικα

Στα "Coding exercises" υπάρχουν προγραμματιστικές ασκήσεις (Εικόνα 4-6) που αντιστοιχούν στις ίδιες θεματικές ενότητες.



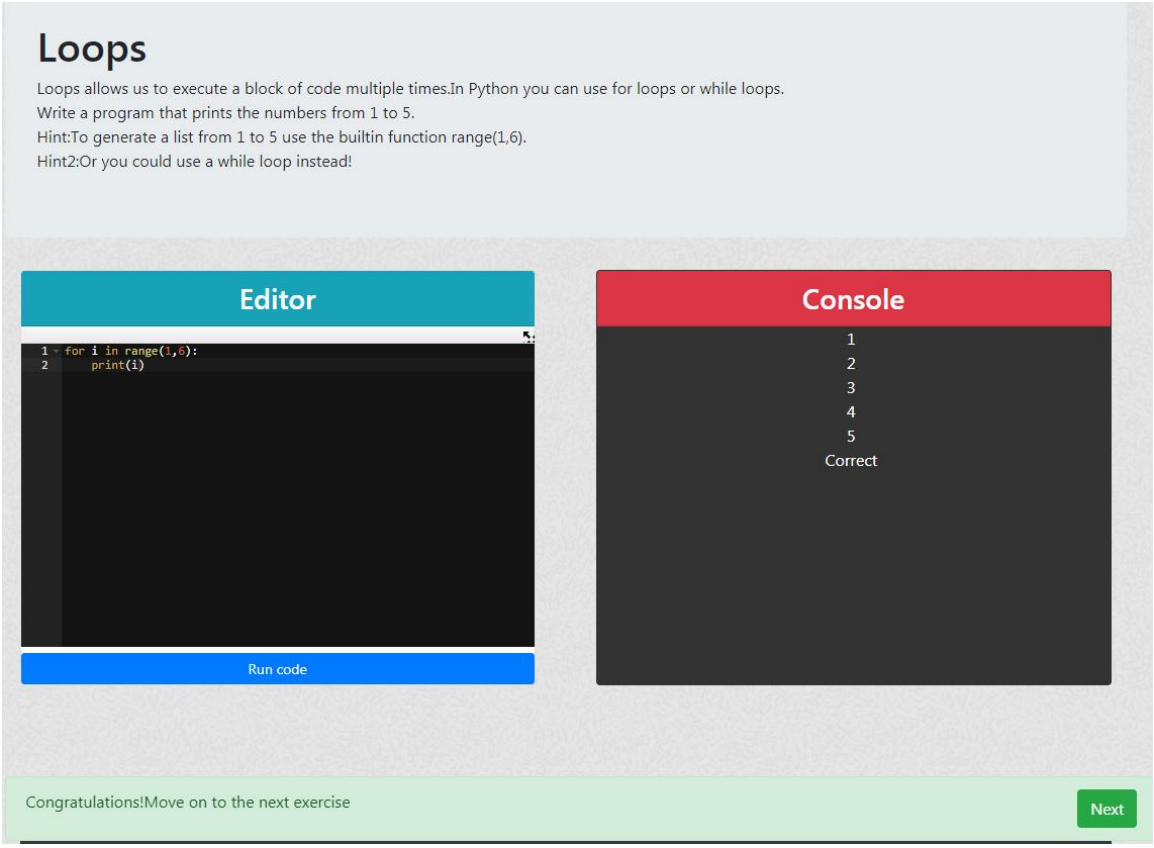
Εικόνα 4-6: Λίστα με Προγραμματιστικές Ασκήσεις

Ανοίγοντας μια άσκηση (Εικόνα 4-7) ο χρήστης βλέπει την εκφώνηση του προβλήματος στην κορυφή της σελίδας και από κάτω έναν επεξεργαστή κειμένου στον οποίο μπορεί να γράψει τον κώδικα του και δίπλα την κονσόλα στην οποία θα εκτυπωθεί το αποτέλεσμα. Κάτω από αυτά βρίσκονται οι προηγούμενες λύσεις που έχει προσφέρει ο χρήστης εάν αυτές υπάρχουν (Εικόνα 4-11).



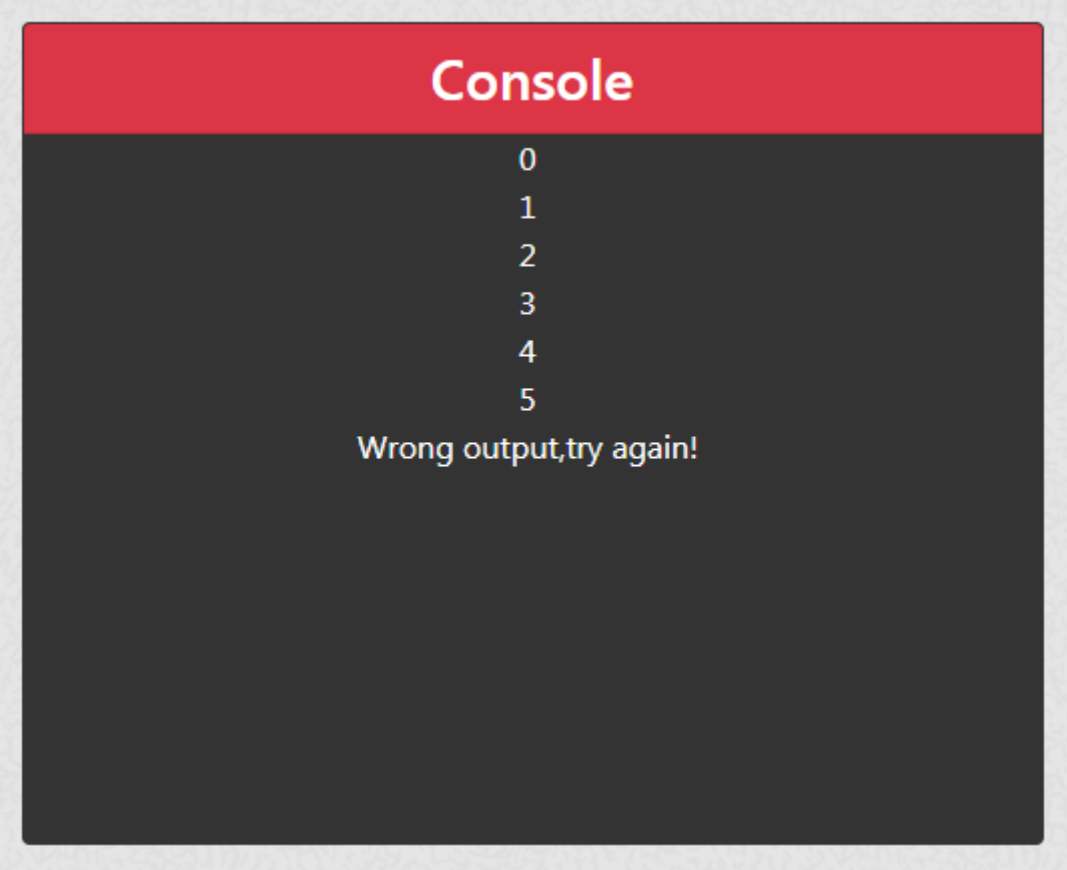
Εικόνα 4-7: Παράδειγμα Άσκησης

Σε περίπτωση σωστής λύσης (Εικόνα 4-8) τυπώνεται στη κονσόλα το μήνυμα “Correct” και εμφανίζεται ένα “modal” που λέει στον χρήστη να προχωρήσει.



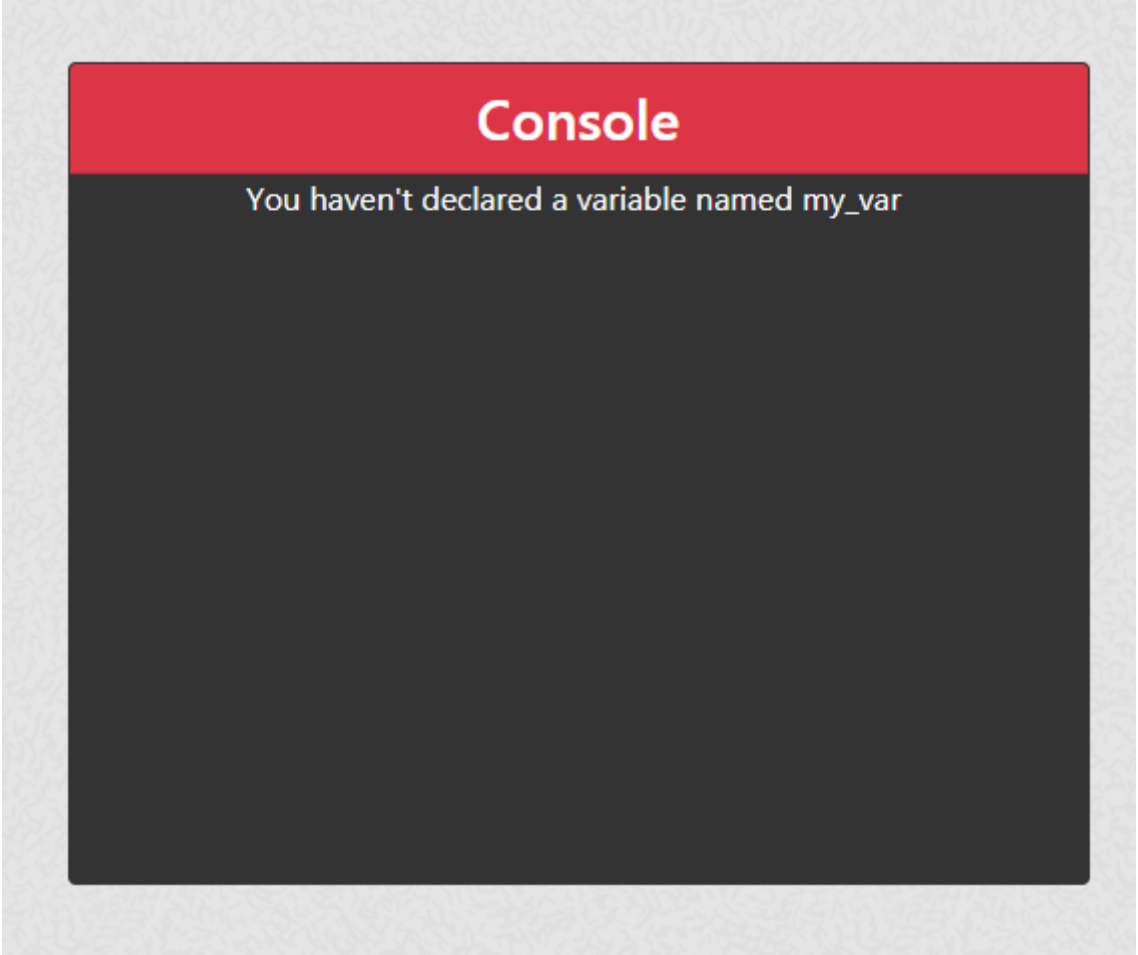
Εικόνα 4-8: Παράδειγμα Αποτελέσματος Σωστής Λύσης

Σε περίπτωση που η λύση του χρήστη είναι λάθος (Εικόνα 4-9) τυπώνεται στην κονσόλα το μήνυμα πως η έξοδος της λύσης είναι λάθος.



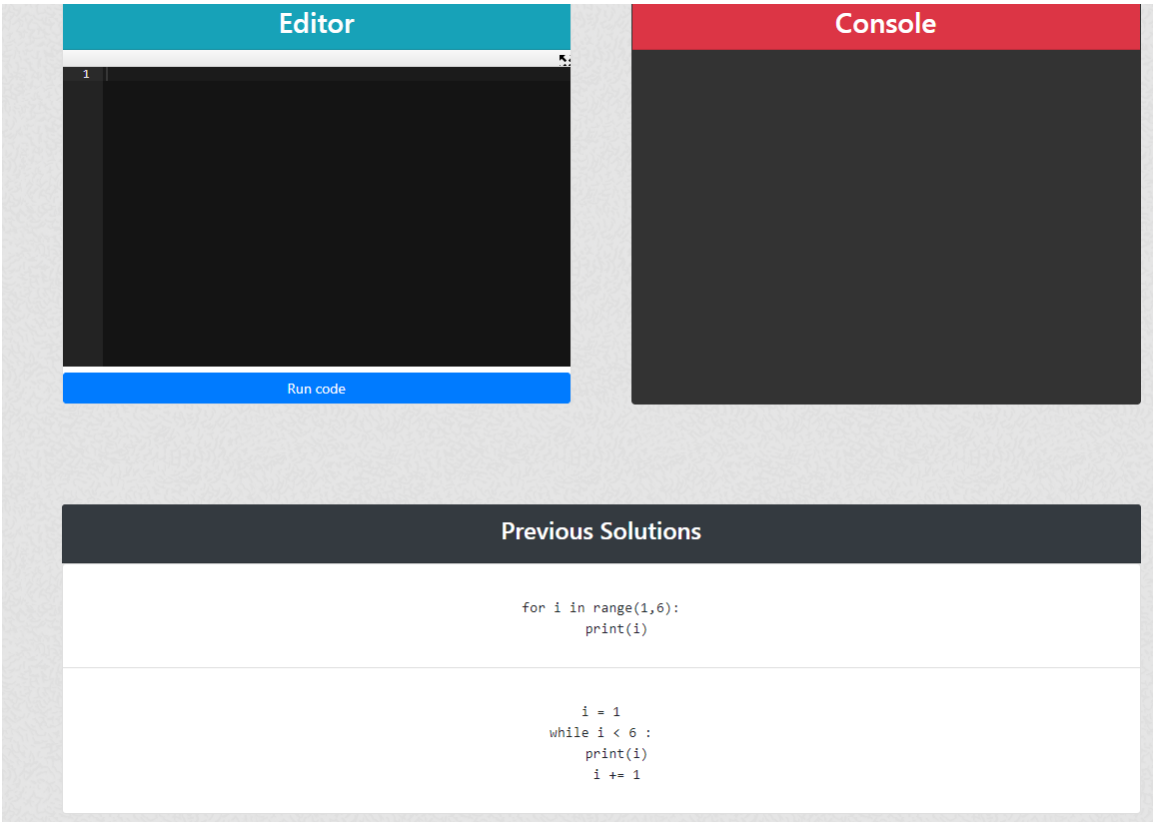
Εικόνα 4-9: Παράδειγμα Αποτελέσματος Λάθος Λύσης

Σε περίπτωση που η άσκηση ζητά από τον χρήστη να δώσει σε μια μεταβλητή ή μια μέθοδο κάποιο συγκεκριμένο όνομα και ο χρήστης δεν το χρησιμοποιήσει τυπώνεται στην κονσόλα αντίστοιχο μήνυμα λάθους (Εικόνα 4-10).



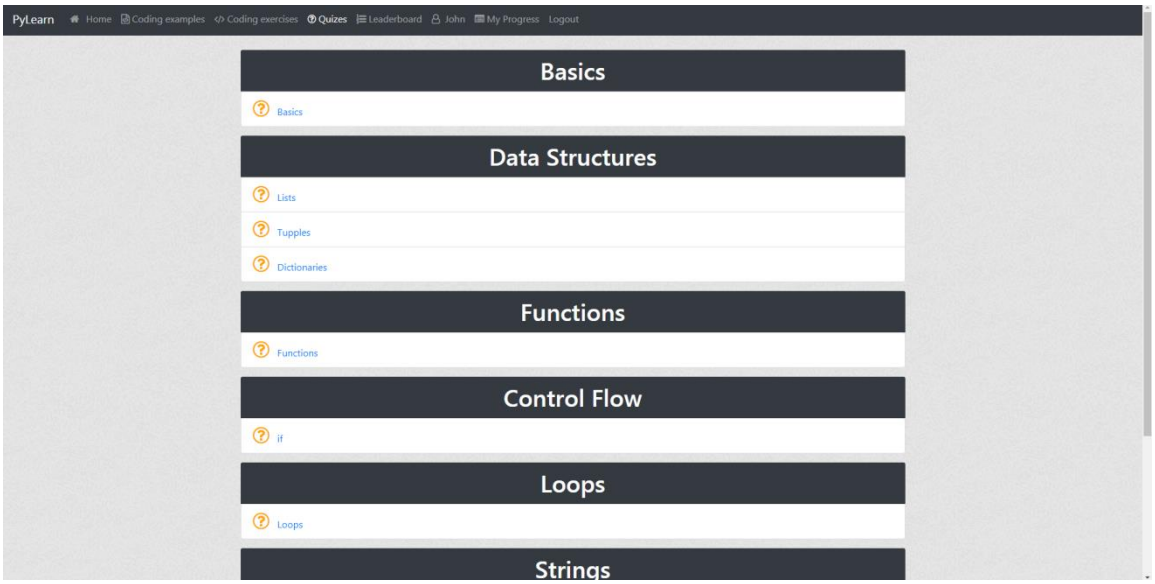
Εικόνα 4-10: Παράδειγμα Χρήσης Λάθος Ονόματος Μεταβλητής

Αν ο χρήστης εισάγει πανομοιότυπη λύση αυτή θα γίνει δεκτή αλλά δεν θα προστεθεί ξανά στη λίστα (Εικόνα 4-11).

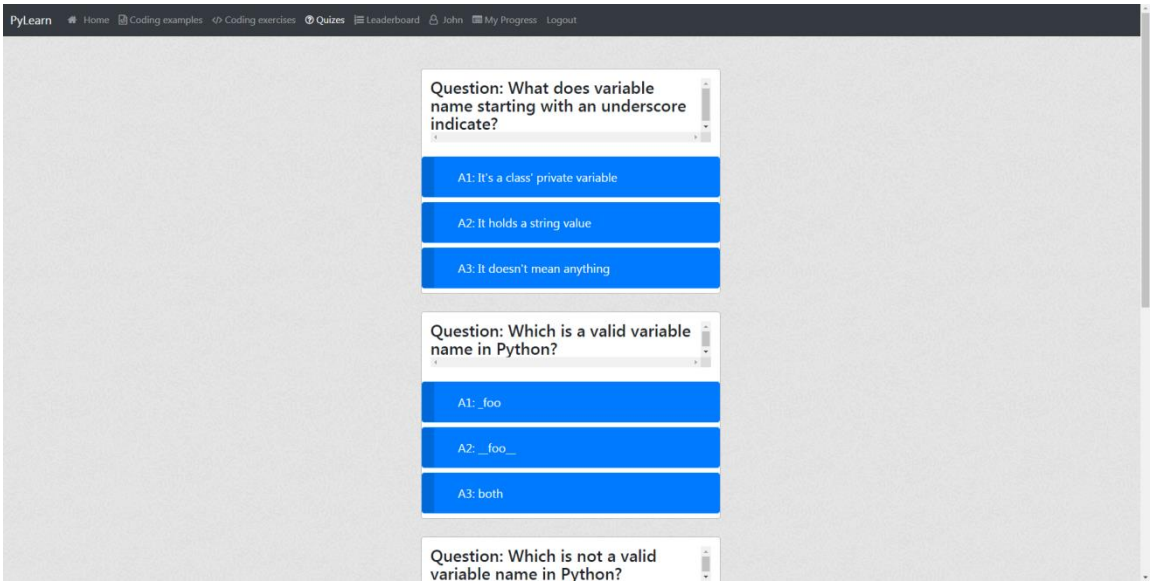


Εικόνα 4-11: Παράδειγμα Αποθηκευμένης Λύσης

Για κάθε θεματική ενότητα υπάρχουν και αντίστοιχα κουιζ με ερωτήσεις πολλαπλής επιλογής (Εικόνα 4–12, 4-13).

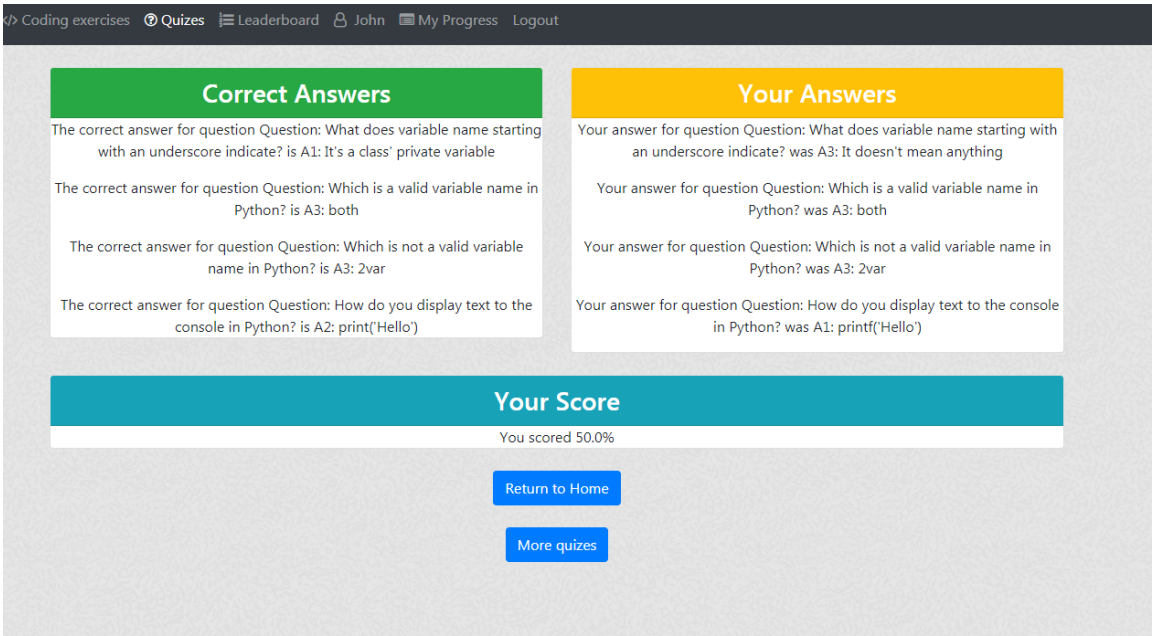


Εικόνα 4-12: Λίστα με Κουιζ



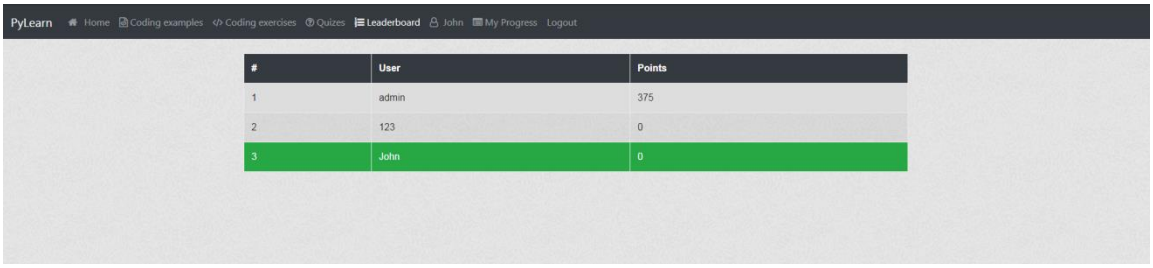
Εικόνα 4-13: Παράδειγμα Κουιζ

Αφού ολοκληρώσει το κουιζ ο χρήστης του παρουσιάζονται τα αποτελέσματα (Εικόνα 4-10). Ο χρήστης έχει τη δυνατότητα να επαναλάβει το κουίζ, οι ερωτήσεις όμως είναι στατικές.



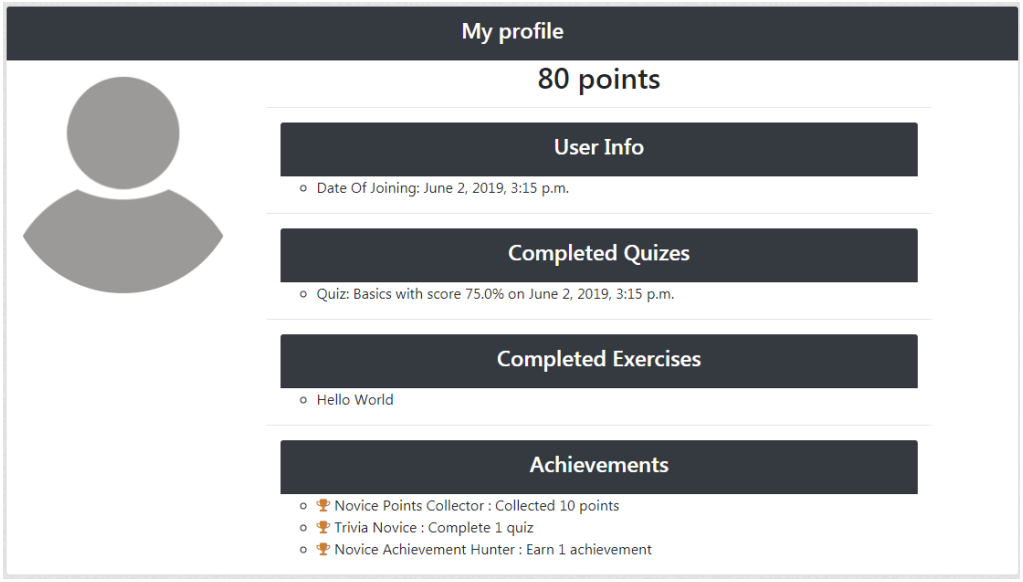
Εικόνα 4-14: Παράδειγμα Αποτελεσμάτων Κουιζ

Στο “Leaderboard” ο χρήστης μπορεί να δεί την κατάταξη όλων των χρηστών της εφαρμογής (Εικόνα 4–11).



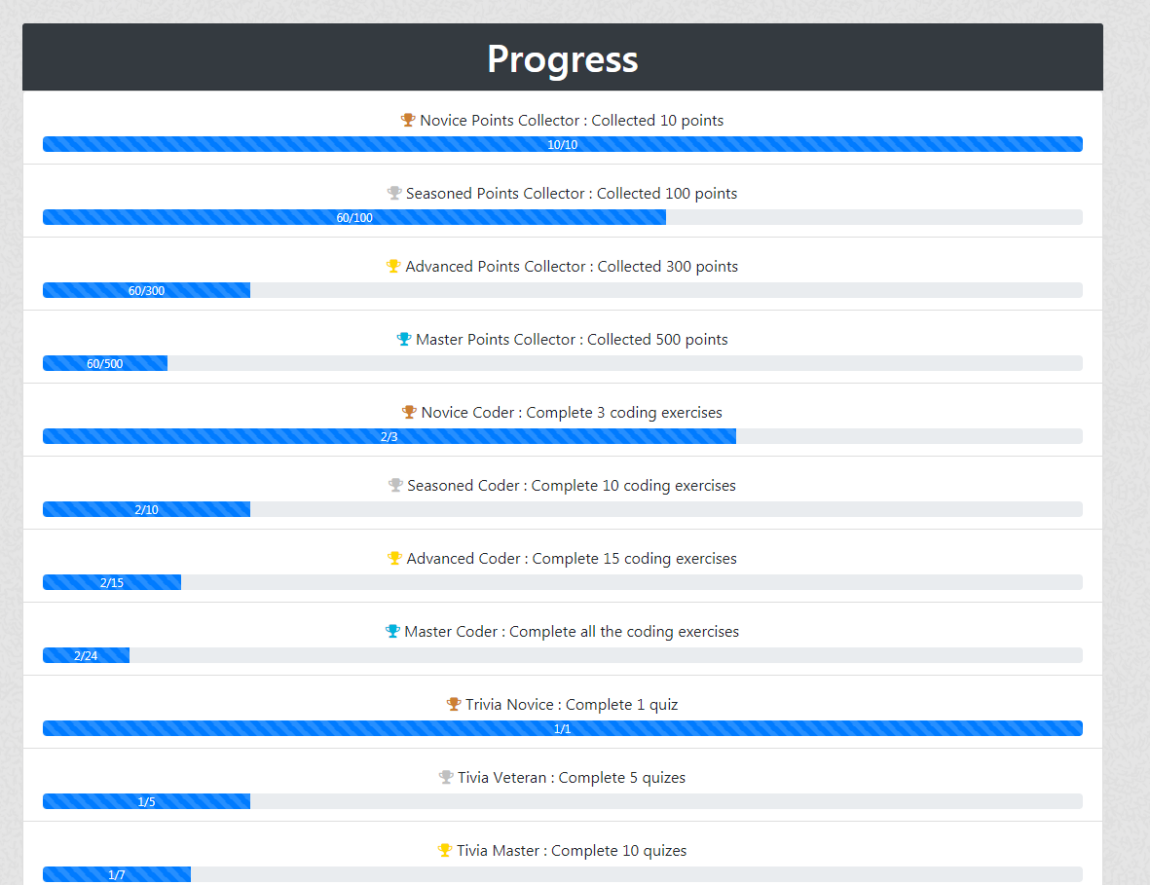
Εικόνα 4-15: Πίνακας Κατάταξης

Αφού συνδεθεί ο χρήστης οι επιλογές για σύνδεση και εγγραφή αλλάζουν και τη θέση τους παίρνουν οι επιλογές προβολής του προφίλ, της προόδου και της αποσύνδεσης. Στο προφίλ του χρήστη (Εικόνα 4–12) αναγράφονται πληροφορίες όπως η ημερομηνία εγγραφής και τι ασκήσεις και κουιζ έχει ολοκληρώσει καθώς και τα επιτεύγματα του.



Εικόνα 4-16: Παράδειγμα Προφίλ Χρήστη

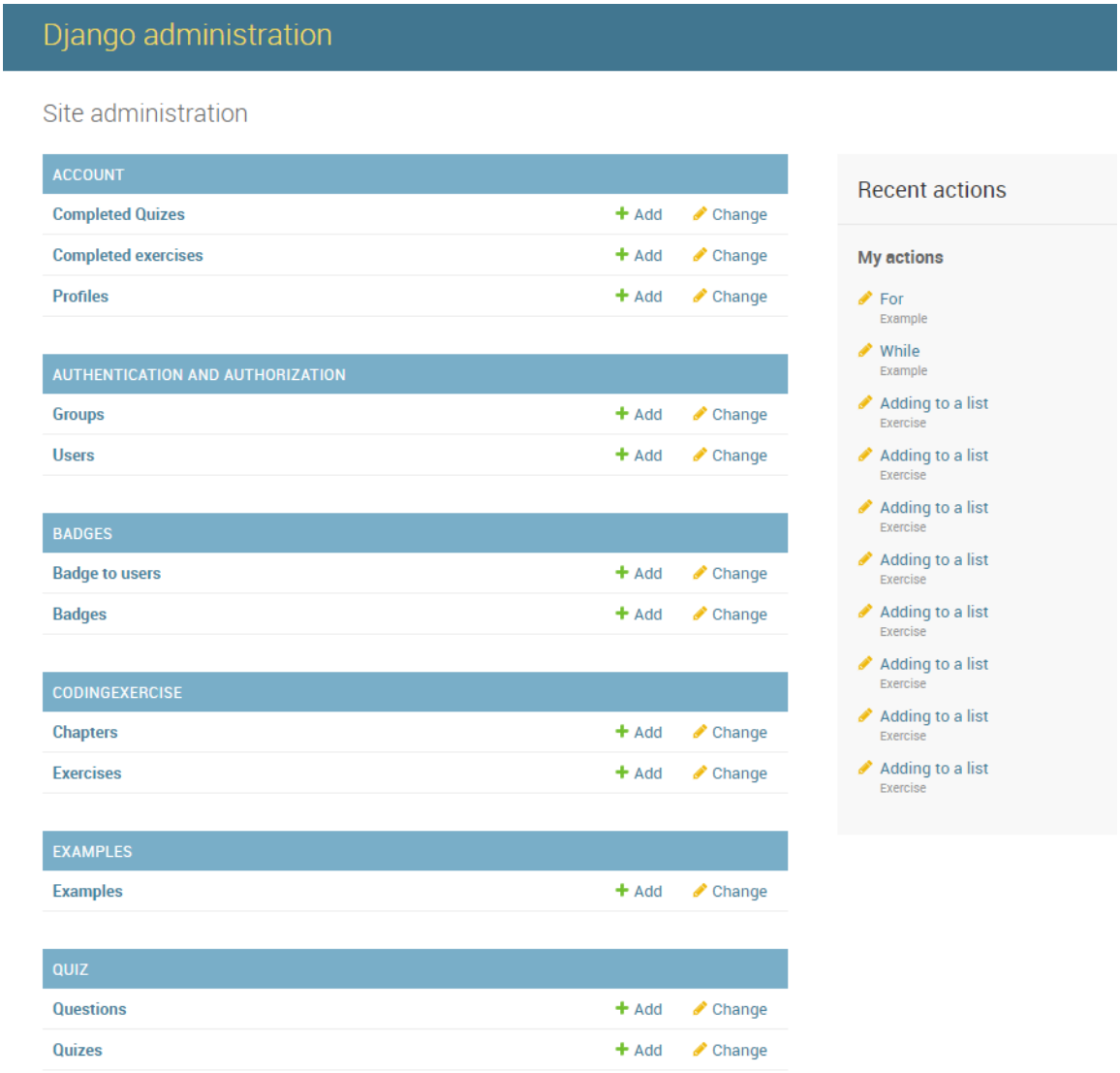
Τέλος, ο χρήστης μπορεί να δει σε τι ποσοστό έχει ολοκληρώσει τους στόχους της εφαρμογής στο “My progress” (Εικόνα 4-13).



Εικόνα 4-17: Παράδειγμα Σελίδας Προόδου

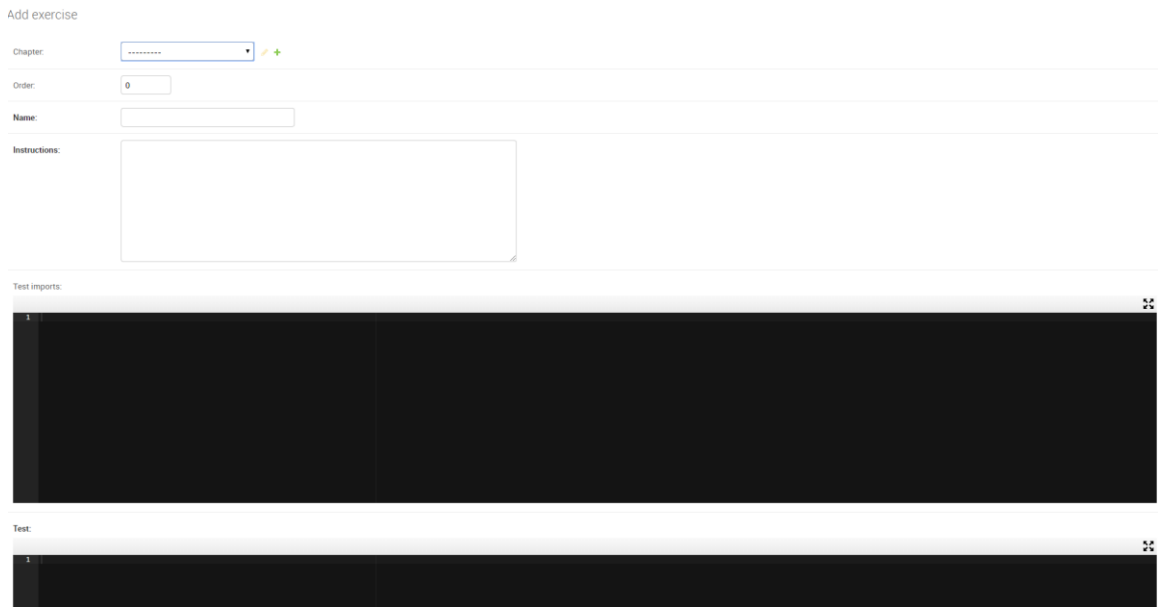
4.2 Λειτουργία για Διαχειριστή

Το Django περιλαμβάνει ένα app που δίνει τη δυνατότητα διαχείρισης της βάσης δεδομένων μέσω γραφικού περιβάλλοντος. Ο διαχειριστής (superuser) μπορεί να καταχωρήσει νέα δεδομένα ή να κάνει αλλαγές στα υπάρχοντα (Εικόνα 4-14).



Εικόνα 4-18: Κεντρική σελίδα διαχείρισης

Για να προσθέσει ο διαχειριστής μια νέα άσκηση μπορεί να κάνει επιλέξει "Add" δίπλα στον πίνακα "Exercises" που φαίνεται στην Εικόνα 4-14 και θα εμφανιστεί μια φόρμα με όλα τα πεδία που πρέπει να συμπληρώσει για την δημιουργία μιας άσκησης (Εικόνα 4-15) και αφού τη συμπληρώσει μπορεί να αποθηκεύσει πατώντας το κουμπί "Save".

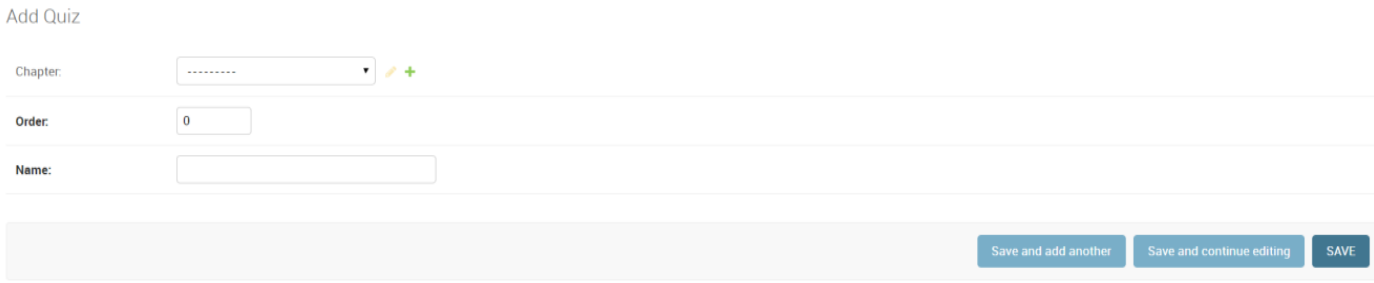


Εικόνα 4-19: Φόρμα Προσθήκης Άσκησης

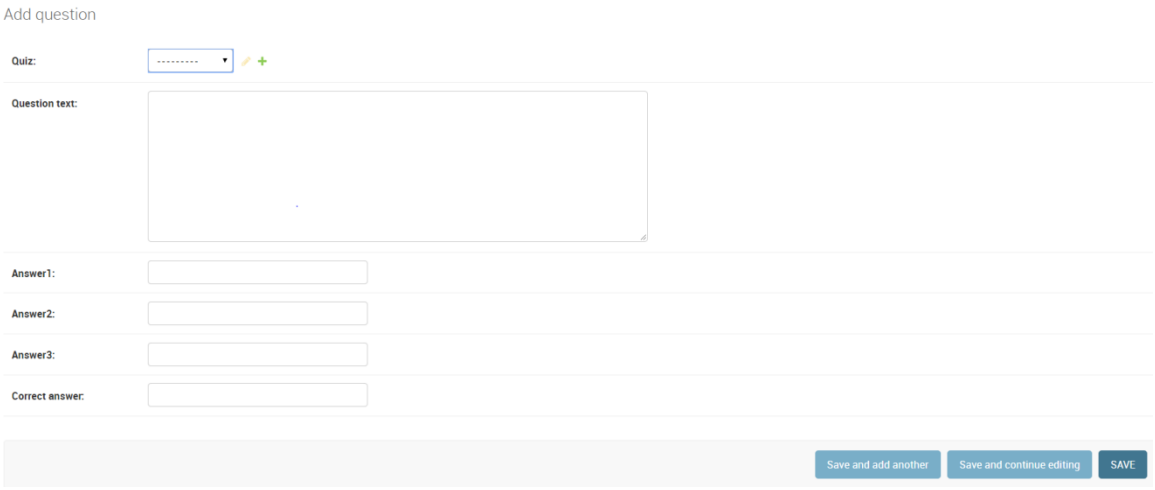
Για να κάνει αλλαγές σε υπάρχουσες ασκήσεις αρκεί να πάει στον πίνακα “Exercises” και να επιλέξει την άσκηση που θέλει να αλλάξει ή να διαγράψει (Εικόνα 4-16).

Εικόνα 4-20: Φόρμα Αλλαγής ή Διαγραφής Άσκησης

Ομοίως μπορεί να δημιουργήσει τα κουίζ (Εικόνα 4-17) και τις ερωτήσεις τους (Εικόνα 4-18).



Εικόνα 4-21: Φόρμα Δημιουργίας Κουίζ



Εικόνα 4-22: Φόρμα Δημιουργίας Ερώτησης

5 Επίλογος

5.1 Σύνοψη και συμπεράσματα

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας έχει ως στόχο την εκμάθηση βασικών τεχνικών και εννοιών προγραμματισμού με Python μέσω ενός συστήματος που χρησιμοποιεί στοιχεία παιχνιδοποίησης.

Η εφαρμογή περιλαμβάνει εκπαιδευτικό υλικό, τεστ ερωτήσεων πολλαπλής επιλογής και προγραμματιστικές ασκήσεις χωρισμένα σε θεματικές ενότητες. Καλύπτονται τα θέματα που αφορούν τη χρήση μεταβλητών, δομών δεδομένων, δομών επιλογής και επανάληψης καθώς και οι βασικές έννοιες του αντικειμενοστρεφούς προγραμματισμού. Τα στοιχεία παιχνιδοποίησης που ενσωματώθηκαν είναι η ανταμοιβή του χρήστη με πόντους για τη σωστή λύση ασκήσεων, η χρήση πίνακα κατάταξης των χρηστών με βάση τους πόντους που έχουν κερδίσει και η ανταμοιβή των χρηστών με τρόπαια όταν ολοκληρώσουν κάποια δραστηριότητα, όπως για παράδειγμα την λύση δέκα προγραμματιστικών ασκήσεων ή τη συλλογή εκατό πόντων. Σε αντίθεση με πολλές από τις εφαρμογές που υπάρχουν ήδη η εκτέλεση του κώδικα των χρηστών δεν γίνεται σε κάποιον απομακρυσμένο διακομιστή αλλά στο πρόγραμμα περιήγησης του χρήστη.

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python και το “framework” Django. Το Django:

- είναι γραμμένο σε Python που είναι απλή στην κατανόηση για τον προγραμματιστή ακόμα και για αρχάριους
- είναι σχεδιασμένο για να βοηθάει στη γρήγορη δημιουργία εφαρμογών
- περιέχει έτοιμα πολλά από τα εργαλεία που χρειάζονται για την ανάπτυξη μιας εφαρμογής ιστού όπως το σύστημα δρομολόγησης διευθύνσεων
- παρέχει γραφικό περιβάλλον διαχείρισης της βάσης δεδομένων
- μπορεί να κλιμακωθεί οριζόντια και
- έχει πολύ καλή τεκμηρίωση, καθώς και μια μεγάλη κοινότητα προγραμματιστών που το χρησιμοποιούν και το στηρίζουν.

Τα μειονεκτήματα του Django είναι:

- η χρήση κανονικών εκφράσεων για τη δήλωση διευθύνσεων δρομολόγησης, καθώς οι κανονικές εκφράσεις μπορούν να γίνουν δύσκολες στην κατανόηση

- η απαίτηση για κατανόηση του τρόπου λειτουργίας του πυρήνα του "framework" και
- η έλλειψη ευελιξίας όσον αφορά σε αλλαγές όταν μεγαλώνει το μέγεθος της εφαρμογής.

5.2 Όρια και περιορισμοί της έρευνας

Η επιλογή της εκτέλεσης του κώδικα των χρηστών στο πρόγραμμα περιήγησης τους έχει ως μειονέκτημα τον περιορισμό της πολυπλοκότητας των ασκήσεων που μπορούν να ενσωματωθούν στην εφαρμογή επειδή ο έλεγχος ορθότητας των λύσεων θα ήταν αδύνατος με αποτέλεσμα την έλλειψη της δυνατότητας να καλυφθούν πιο δύσκολα θέματα.

Τέλος, κρίνεται σημαντικό η εφαρμογή να χρησιμοποιηθεί από αρχάριους στον προγραμματισμό προκειμένου να ελεγχθεί ο βαθμός στον οποίο η εφαρμογή μπορεί να τους υποστηρίξει στην κατανόηση των βασικών ενεργειών προγραμματισμού με την Python, καθώς επίσης και αν ο κώδικας της ίδιας της εφαρμογής μπορεί να χρησιμοποιηθεί ως μια μελέτη περίπτωσης ανάπτυξης μιας διαδικτυακής εφαρμογής με την Python και το Django.

5.3 Μελλοντικές Επεκτάσεις

Μια επέκταση που θα μπορούσε να γίνει είναι η περεταίρω χρήση στοιχείων παιχνιδοποίησης, όπως για παράδειγμα η χρήση ενός εικονικού χαρακτήρα (avatar) για κάθε χρήστη, Επίσης, οι χρήστες θα μπορούσαν να έχουν τη δυνατότητα να ανταλλάξουν πόντους για να παραμετροποιήσουν τον εικονικό χαρακτήρα τους.

Μια άλλη επέκταση που θα μπορούσε να γίνει είναι να δίνεται στους χρήστες η δυνατότητα να δημιουργήσουν τις δικές τους ασκήσεις και να τις μοιράζονται με την κοινότητα.

Τέλος, θα μπορούσε να αναπτυχθεί ένα ερωτηματολόγιο που θα συμπληρωθεί από χρήστες που έχουν αλληλεπιδράσει με την εφαρμογή ώστε να υπάρξει καλύτερη κατανόηση της αποτελεσματικότητας της εφαρμογής.

Βιβλιογραφία

Borges, S. S., Durelli, V. H. S., Reis, H. M., & Isotani, S. (2014). A systematic mapping on gamification applied to education. In *ACM SAC'14 Conference, Gyeongju, South Korea* (pp. 216–222).]

Connolly, Thomas & Boyle, Elizabeth & Macarthur, Ewan & Hainey, Thomas & Boyle, James. (2012). A systemic literature review of empirical evidence on computer games and serious games. Computers & Education. 59. 661 – 686. 10.1016/j.compedu.2012.03.004.

S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification", In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, September 28-30, 2011, Tampere, Finland, ACM, pp. 9-15

Dickey, J. (2014) Write Modern Web Apps with the Mean Stack: Mongo, Express, AngularJS, and Node.js (Develop and Design), first ed. Peachpit Press. ISBN-10: 0133930157.

Glover, I. (2013). Play As You Learn: Gamification as a Technique for Motivating Learners. In J. Herrington, A. Couros & V. Irvine (Eds.), *Proceedings of EdMedia 2013--World Conference on Educational Media and Technology* (pp. 1999-2008).

King D, Greaves F, Exeter C, Darzi A. 'Gamification': Influencing health behaviours with games. J R Soc Med 2013 Mar;106(3):76-78

Kiryakova, G., Angelova, N., & Yordanova, L. (2014). Gamification in education, Proceedings of 9th International Balkan Education and Science Conference.

J. Lee and B. Ware. Open source Web development with LAMP: using Linux, Apache, MySQL, Perl, and PHP. Addison-Wesley Professional, 2003

Mekler, E. D., Brühlmann, F., Opwis, K., & Tuch, A. N. (2013, October). Do points, levels and leaderboards harm intrinsic motivation?: an empirical analysis of common gamification elements. In *Proceedings of the First International Conference on gameful design, research, and applications* (pp. 66-73). ACM

Robert Mitchell, Lisa Schuster, Hyun Seung Jin. (2018) Gamification and the impact of extrinsic motivation on needs satisfaction: Making work fun?. *Journal of Business Research*.

Robinson, D., & Bellotti, V. (2013, April). A preliminary taxonomy of gamification elements for varying anticipated commitment. In *Proceedings CHI 2013*

Workshop on Designing Gamification: Creating Gameful and Playful Experiences. ACM. Recuperado de https://goo. gl/hElEnm.

Vasan Subramanian (2017). Pro MERN stack: full stack web app development with Mongo, Express, React, and Node. Springer.

Brython, Available at: http://brython.info/index.html (May 25 2019).

CodinGame, Available at https://www.codingame.com/ (May 25 2019).

Django, Django: The Web framework for perfectionists with deadlines, Available at: https://www.djangoproject.com/ (May 25 2019).

Foldit: Solve Puzzles for Science, Available at: https://fold.it/portal/ (May 25 2019).

FreeRice, Available at: http://freerice.com (May 25 2019).

Python, Python Software Foundation, Available at: https://www.python.org/ (May 25 2019).

SoloLearn: Learn to Code for Free!, Available at https://www.sololearn.com/ (May 25 2019).

Παράρτημα Α - Κώδικας

Σε αυτό το παράρτημα συμπεριλαμβάνεται ο κώδικας για τις προβολές, τα μοντέλα, τις διευθύνσεις δρομολόγησης και τα πρότυπα της εφαρμογής καθώς και ο κώδικας του αρχείου ρυθμίσεων. Όλος ο κώδικας της εφαρμογής είναι διαθέσιμος στο GitHub.

settings.py

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = SECRET_KEY =
os.environ.get('SECRET_KEY','1111111111111111111111111111111111111111')

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False

ALLOWED_HOSTS = ['pylearn.herokuapp.com']


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'codingExercise',
    'django_ace',
    'account',
    'examples',
    'quiz',
    'leaderboard',
    'badges',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

```
]

ROOT_URLCONF = 'editorModule.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, '../templates/')]
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'editorModule.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, '../db3.sqlite3'),
    }
}

import dj_database_url
db_from_env = dj_database_url.config()
DATABASES['default'].update(db_from_env)
#DATABASES['default']['CONN_MAX_AGE'] = 500


# Password validation
# https://docs.djangoproject.com/en/1.11/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidat
or',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/1.11/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.11/howto/static-files/



STATIC_URL = '/static/'

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "../static"),
]

STATIC_ROOT = os.path.join(BASE_DIR, '../static-root')
STATICFILES_STORAGE =
'whitenoise.storage.CompressedManifestStaticFilesStorage'

LOGIN_URL = '/account/login/'

LOGIN_REDIRECT_URL='/'

CORS_REPLACE_HTTPS_REFERER      = True
HOST_SCHEME                     = "https://"
SECURE_PROXY_SSL_HEADER         = ('HTTP_X_FORWARDED_PROTO', 'https')
SECURE_SSL_REDIRECT             = True
SESSION_COOKIE_SECURE           = True
CSRF_COOKIE_SECURE              = True
SECURE_HSTS_INCLUDE_SUBDOMAINS  = True
SECURE_HSTS_SECONDS             = 1000000
SECURE_FRAME_DENY               = True
```

account/models.py

```
from datetime import datetime

from django.contrib.auth.models import User
from django.db import models
from django.utils import formats
```

```
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,
related_name='profile')
    points = models.IntegerField(default=0)
    totalQuizScore = models.FloatField(default=0)
    quizAvgScore = models.FloatField(default=0)
    num_of_copmleted_exercises = models.IntegerField(default=0)
    num_of_copmleted_quizes = models.IntegerField(default=0)
    num_of_achievements = models.IntegerField(default=0)

    def __str__(self):
        return self.user.get_username()

    def getCompletedQuizes(self, flatFlag):
        if flatFlag:
            completed_quizes =
CompletedQuiz.objects.filter(userProfile=self).values_list('quiz_name',
flat=flatFlag)
        else:
            completed_quizes =
CompletedQuiz.objects.filter(userProfile=self)
        return completed_quizes

    def getCompletedExercises(self, flatFlag):
        if flatFlag:
            completed_exercises =
CompletedExercise.objects.filter(userProfile=self).values_list('exercis
e_name',

flat=flatFlag)
        else:
            completed_exercises =
CompletedExercise.objects.filter(userProfile=self)
        return completed_exercises

    def get_num_of_completed_exercises(self):
        return
CompletedExercise.objects.filter(userProfile=self).distinct().count()

    def setAvgQuizSore(self, newScore):
        self.totalQuizScore += newScore
        tmpCompletedQuizes = self.user.profile.num_of_copmleted_quizes
if self.user.profile.num_of_copmleted_quizes > 0 else 1
        self.quizAvgScore = self.totalQuizScore / tmpCompletedQuizes


class CompletedQuiz(models.Model):
    userProfile = models.ForeignKey(Profile, on_delete=models.CASCADE)
    quiz_name = models.CharField(max_length=50, blank=True, null=True)
    score = models.CharField(max_length=10, default="0")
    date = models.DateTimeField(default=datetime.now)

    def __str__(self):
        return self.quiz_name

    class Meta:
        verbose_name = 'Completed Quiz'
        verbose_name_plural = 'Completed Quizes'
```

```
class CompletedExercise(models.Model):
    userProfile = models.ForeignKey(Profile, on_delete=models.CASCADE)
    exercise_name = models.CharField(max_length=50, blank=True,
null=True)
    answer = models.TextField()
    date = models.DateTimeField(default=datetime.now)

    def __str__(self):
        formatted_datetime = formats.date_format(self.date,
"SHORT_DATETIME_FORMAT")
        return "Exercise: " + self.exercise_name + " on : " +
str(formatted_datetime)


import account.meta_badges
```

account/urls.py

```
from django.conf.urls import url
from django.contrib.auth.views import LoginView, LogoutView

from .views import RegisterUserView, UserProfileView, ProgressView

urlpatterns = [

    url(r'^register/$', RegisterUserView.as_view(), name='register'),
    url(r'^profile/(?P<pk>[0-9]+)$', UserProfileView.as_view(),
name='profile'),
    url(r'^login/$',
LoginView.as_view(template_name='account/login.html',
redirect_authenticated_user=True),
        name='login'),
    url(r'^logout/$', LogoutView.as_view(), name='logout'),
    url(r'^progress/(?P<pk>[0-9]+)$', ProgressView.as_view(),
name='progress'),

]
```

account/views.py

```
from badges.models import Badge
from django.contrib.auth import authenticate
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib.auth.views import login
from django.http import HttpResponseForbidden
from django.shortcuts import render
from django.views.generic import CreateView, DetailView

from account.models import Profile, CompletedExercise
from .forms import RegisterForm


class RegisterUserView(CreateView):
    form_class = RegisterForm
```

```
    template_name = 'account/register.html'
    success_url = 'home'

    def dispatch(self, request, *args, **kwargs):
        if request.user.is_authenticated():
            return HttpResponseForbidden()

        return super(RegisterUserView, self).dispatch(request, *args,
**kwargs)

    def form_valid(self, form):
        user = form.save(commit=False)
        user.set_password(form.cleaned_data['password'])
        user.save()
        # create user profile
        Profile.objects.create(user=user)
        # login user after registrations
        user = authenticate(username=form.cleaned_data['username'],
                            password=form.cleaned_data['password'],
                            )
        login(self.request, user)
        return super(RegisterUserView, self).form_valid(form)


class UserProfileView(LoginRequiredMixin, DetailView):
    model = Profile
    template_name = 'account/profile.html'
    context_object_name = 'profile'

    def get_context_data(self, **kwargs):
        context = super(UserProfileView,
self).get_context_data(**kwargs)
        profileObject =
Profile.objects.get(user__username=self.request.user)
        completed_exercises =
CompletedExercise.objects.filter(userProfile=profileObject).values_list
('exercise_name',

flat=True).distinct()
        context['completed_exercises'] = completed_exercises
        return context


class ProgressView(LoginRequiredMixin, DetailView):
    model = Profile
    template_name = 'account/progress.html'
    context_object_name = 'profile'

    def get_context_data(self, **kwargs):
        context = super(ProgressView, self).get_context_data(**kwargs)
        context['badges'] = Badge.objects.exclude(level__iexact="5")
        return context

    def get(self, request, *args, **kwargs):
        self.object = self.get_object()
        if int(kwargs['pk']) == request.user.profile.id:
            context = self.get_context_data()
            return self.render_to_response(context)
        else:
```

```
            return HttpResponseForbidden(render(request,
"PermissionDenied.html"))
```

codingExercise/models.py

```
from django.db import models


class Chapter(models.Model):
    name = models.TextField(max_length=100)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['id']


class Exercise(models.Model):
    chapter = models.ForeignKey(Chapter, on_delete=models.CASCADE,
blank=True, null=True)
    order = models.IntegerField(blank=True, null=True, default=0)
    name = models.CharField(max_length=100, null=False, blank=False, )
    instructions = models.TextField(null=False, blank=False)
    test_imports = models.TextField(blank=True)
    test = models.TextField(null=False, blank=False)
    text = models.TextField(null=True, blank=True)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['order']
```

codingExercise/urls.py

```
from django.conf.urls import url

from .views import CodingExerciseListView, CodingExerciseDetailView,
Finished

app_name = 'codingExercise'
urlpatterns = [
    url(r'^finished$', Finished.as_view(), name='finished'),
    url(r'^$', CodingExerciseListView.as_view(), name='exercise_list'),
    url(r'^(?P<pk>[0-9]+)$', CodingExerciseDetailView.as_view(),
name='exercise'),

]
```

codingExercise/views.py

```
from django.db import models
from django.db.models.signals import post_save
from django.dispatch import receiver

from codingExercise.models import Chapter


class Example(models.Model):
    chapter = models.ForeignKey(Chapter, on_delete=models.CASCADE)
    order = models.IntegerField(blank=True, null=True, default=0)
    title = models.CharField(max_length=200)
    content = models.TextField()

    def __str__(self):
        return self.title

    class Meta:
        ordering = ['order']
```

examples/models.py

```
from django.db import models
from django.db.models.signals import post_save
from django.dispatch import receiver

from codingExercise.models import Chapter


class Example(models.Model):
    chapter = models.ForeignKey(Chapter, on_delete=models.CASCADE)
    order = models.IntegerField(blank=True, null=True, default=0)
    title = models.CharField(max_length=200)
    content = models.TextField()

    def __str__(self):
        return self.title

    class Meta:
        ordering = ['order']
```

examples/urls.py

```
from django.conf.urls import url

from .views import CodingExerciseListView, CodingExerciseDetailView,
Finished

app_name = 'codingExercise'
urlpatterns = [
    url(r'^finished$', Finished.as_view(), name='finished'),
    url(r'^$', CodingExerciseListView.as_view(), name='exercise_list'),
    url(r'^(?P<pk>[0-9]+)$', CodingExerciseDetailView.as_view(),
```

```
name='exercise'),

]
```

examples/views.py

```
from django.contrib.auth.mixins import LoginRequiredMixin

from django.http import HttpResponseRedirect, Http404
from django.shortcuts import redirect
from django.views.generic import ListView, DetailView

from codingExercise.models import Chapter
from examples.models import Example


class ExampleListView(LoginRequiredMixin, ListView):
    model = Chapter
    template_name = 'examples/examples_list.html'


class ExampleDetailView(LoginRequiredMixin, DetailView):
    model = Example
    template_name = 'examples/example.html'
    context_object_name = 'example'

    def post(self, *args, **kwargs):
        next_pk = int(kwargs['pk']) + 1
        # if example with id next_pk exists redirect to it else try the
next id
        if next_pk <= int(Example.objects.latest("id").id):
            while Example.objects.filter(id=next_pk).count() == 0:
                next_pk += 1
            else:
                return HttpResponseRedirect(str(next_pk))
        else:
            return redirect('codingExercise:finished')
```

leaderboard/urls.py

```
from django.conf.urls import url
from .views import LeaderboardListView

app_name = 'leaderboard'
urlpatterns = [
    url(r'^$', LeaderboardListView.as_view(), name='leaderboard_list'),

]
```

leaderboard/views.py

```
from django.shortcuts import render
from django.views.generic import ListView
from account.models import Profile


class LeaderboardListView(ListView):
    model = Profile
    template_name = 'leaderBoard/leaderboard_list.html'
    paginate_by = 25

    def get_queryset(self):
        queryset = Profile.objects.all().order_by('-points')
        return queryset
```

quiz/models.py

```
from django.db import models

from codingExercise.models import Chapter


class Quiz(models.Model):
    chapter = models.ForeignKey(Chapter, on_delete=models.CASCADE,
blank=True, null=True)
    order = models.IntegerField(null=True, default=0)
    name = models.CharField(max_length=120, default="")

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = 'Quiz'
        verbose_name_plural = 'Quizes'
        ordering = ['order']


class Question(models.Model):
    quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE, null=True)
    question_text = models.TextField()
    answer1 = models.CharField(max_length=255)
    answer2 = models.CharField(max_length=255)
    answer3 = models.CharField(max_length=255)
    correct_answer = models.CharField(max_length=255)

    def __str__(self):
        return self.question_text
```

quiz/urls.py

```
from django.conf.urls import url
from .views import QuizView, QuizListView
```

```
app_name = 'quiz'
urlpatterns = [
    url(r'^$', QuizListView.as_view(), name='quiz_list'),
    url(r'^(?P<pk>[0-9]+)$', QuizView.as_view(), name='quiz_details'),

]
```

quiz/views.py

```
from django.contrib.auth.mixins import LoginRequiredMixin
from django.http import Http404, HttpResponseNotFound
from django.shortcuts import render
from django.views.generic import DetailView, ListView

from account.models import Profile, CompletedQuiz
from codingExercise.models import Chapter
from .models import Quiz


class QuizListView(LoginRequiredMixin, ListView):
    model = Chapter
    template_name = 'quiz/quiz_list.html'

    def get_context_data(self, **kwargs):
        context = super(QuizListView, self).get_context_data(**kwargs)
        profile_object =
Profile.objects.get(user__username=self.request.user)
        completed_quizes = profile_object.getCompletedQuizes(True)
        context['active'] = 'quiz'
        context['completed_quizes'] = completed_quizes
        return context


class QuizView(LoginRequiredMixin, DetailView):
    model = Quiz
    context_object_name = 'quiz'

    template_name = 'quiz/quiz.html'

    def get(self, request, *args, **kwargs):
        try:
            self.object = self.get object()
        except Http404:
            # redirect here
            return HttpResponseNotFound(render(request,
"NotFound.html"))
        context = self.get_context_data(object=self.object)
        return self.render_to_response(context)

    def post(self, *args, **kwargs):
        first_attempt = False

        # get quiz name
        current_quiz = Quiz.objects.filter(id=str(self.kwargs['pk']))
        current_quiz_name = current_quiz.values('name')[0]['name']

        qn = 1
        correct = {}  # dict with {QuestionName : correct answer} pairs
```

```
        for question_text, correct_answer in
zip(current_quiz.values('question__question_text'),

current_quiz.values('question__correct_answer')):
            correct.update({question_text['question__question_text']:
correct_answer['question__correct_answer']})
            qn += 1

        userAnswers = {}
        correct_count = 0

        for i in self.request.POST:

            if i.startswith('Q'):

                if str(self.request.POST[i]) == correct[i]:
                    correct_count += 1  # count correct answers
                userAnswers.update({i: self.request.POST[i]})
        correct_percentage = float(correct_count / (qn - 1))
        correct_percentage = float("{0:.1f}".format(correct_percentage
* 100))

        # check if quiz completed previously

        profileObject =
Profile.objects.get(user__username=self.request.user)

        completed_quiz, created =
CompletedQuiz.objects.get_or_create(userProfile=profileObject,

quiz_name=current_quiz_name)
        print(completed_quiz)
        if created:
            completed_quiz.save()
            # first attempt
            first_attempt = True
            previous_attempt_score = 0
            newHighscore = False
            # save new attempt

            profileObject.num_of_copmleted_quizes += 1
            profileObject.setAvgQuizSore(correct_percentage)
            profileObject.points += correct_percentage

            completed_quiz.score = str(correct_percentage)
            profileObject.save()
            completed_quiz.save()
        else:
            previous_attempt_score = completed_quiz.score
            newHighscore = False
            if correct_percentage > float(previous_attempt_score):  #
replace when new score is better
                newHighscore = True
                completed_quiz.score = str(correct_percentage)
                completed_quiz.save()

        context = {
            'quiz': current_quiz,
            'correct': correct.items(),
            'correct_percentage': correct_percentage,
```

```
            'userAnswers': userAnswers.items(),
            'newHighsScore': newHighscore,
            'prevScore': previous_attempt_score,
            'first_attempt': first_attempt,

        }

        # show results
        return render(self.request, 'quiz/results.html', context)
```

templates/base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">


    {% load static %}


    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta/css/bootstrap.min.css">
    <link rel="stylesheet" href="{% static 'css/base.css' %}">

    {% block extrahead %}

    {% endblock %}
    <title>{% block title %}Home{% endblock %} | site.com</title>

</head>
<body {% block brython %}{% endblock %}>

{% include 'navbar.html' %}



{% block content %}

{% endblock %}



<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
        integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
        crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper
.min.js"
        integrity="sha384-
b/U6ypiBEHpOf/4+1nzFpr53nxSS+GLCkfwBdFNTxtclqqenISfwAzpKaMNFNmj4"
        crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta/js/bootstrap.min.js"
        integrity="sha384-
h0AbiXch4ZDo7tp9hKZ4TsHbi047NrKGLO3SEJAg45jXxnGIfYzk4Si90RDIqNm1"
```

```
        crossorigin="anonymous"></script>
<script src="https://use.fontawesome.com/56f1d9c36d.js"></script>


{% block scripts %}{% endblock %}

</body>
</html>
```

templates/home.html

```
{% extends 'base.html' %}

{% block title %}Home{% endblock %}

{% block content %}
    <div class="container">


        <div class="bd-example">
            <div id="carouselExampleCaptions" class="carousel slide"
data-ride="carousel">
                <ol class="carousel-indicators">
                    <li data-target="#carouselExampleCaptions" data-
slide-to="0" class="Examples"></li>
                    <li data-target="#carouselExampleCaptions" data-
slide-to="1" class=""></li>
                    <li data-target="#carouselExampleCaptions" data-
slide-to="2" class=""></li>
                    <li data-target="#carouselExampleCaptions" data-
slide-to="3" class=""></li>
                </ol>
                <div class="carousel-inner">
                    <div class="carousel-item active caro">
                        <a href="{% url 'examples:examples_list' %}"
class="carousel-link">
                            <img class="d-block w-100"
                                 data-
src="holder.js/800x400?auto=yes&amp;bg=777&amp;fg=555&amp;text=Examples
"
                                 alt="First slide [800x400]"
                                 data-holder-rendered="true">

                            <div class="carousel-caption d-none d-md-
block">
                                <h3>Coding Examples</h3>
                                <p>Study the basics</p>
                            </div>
                        </a>
                    </div>
                    <div class="carousel-item">
                        <a href="{% url 'codingExercise:exercise_list'
%}" class="carousel-link">
                            <img class="d-block w-100"
                                 data-
src="holder.js/800x400?auto=yes&amp;bg=777&amp;fg=555&amp;text=Programm
ing"
                                 alt="First slide [800x400]"
                                 data-holder-rendered="true">
```

```
                            <div class="carousel-caption d-none d-md-
block">
                                <h3>Coding Exercises</h3>
                                <p>Practice coding in python online</p>
                            </div>
                        </a>
                    </div>
                    <div class="carousel-item">
                        <a class="carousel-link" href="{% url
'quiz:quiz_list' %} ">
                            <img class="d-block w-100"
                                 data-
src="holder.js/800x400?auto=yes&amp;bg=666&amp;fg=444&amp;text=Quiz"
                                 alt="Second slide [800x400]"
                                 data-holder-rendered="true">


                            <div class="carousel-caption d-none d-md-
block">
                                <h3>Quizes</h3>
                                <p>Test your knowledge.</p>
                            </div>
                        </a>
                    </div>
                    <div class="carousel-item">
                        <a class="carousel-link" href="{% url
'leaderboard:leaderboard_list' %} ">
                            <img class="d-block w-100"
                                 data-
src="holder.js/800x400?auto=yes&amp;bg=555&amp;fg=333&amp;text=Scores"
                                 alt="Third slide [800x400]"
                                 data-holder-rendered="true">

                            <div class="carousel-caption d-none d-md-
block">
                                <h3>Leaderdoard</h3>


                                <p>Check your progress compared to
other users</p>
                            </div>
                        </a>
                    </div>
                </div>
                <a class="carousel-control-prev"
href="#carouselExampleCaptions" role="button" data-slide="prev">
                    <span class="carousel-control-prev-icon" aria-
hidden="true"></span>
                    <span class="sr-only">Previous</span>
                </a>
                <a class="carousel-control-next"
href="#carouselExampleCaptions" role="button" data-slide="next">
                    <span class="carousel-control-next-icon" aria-
hidden="true"></span>
                    <span class="sr-only">Next</span>
                </a>
            </div>
        </div>

    </div>
```

```
{% endblock %}


{% block scripts %}
    <script type="text/javascript"
src="https://cdn.rawgit.com/imsky/holder/master/holder.js"></script>

{% endblock %}
```

templates/navbar.html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

    {% url 'home' as home %}


    <a class="navbar-brand" href="#">PyLearn</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarsExampleDefault"
            aria-controls="navbarsExampleDefault" aria-expanded="false"
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarsExampleDefault">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item {% if request.path == home %}active{%
endif %}">

                <a class="nav-link" href="{% url 'home' %}"><i
class="fa fa-home fa-fw" aria-hidden="true"></i> 
                    Home <span class="sr-only">(current)</span></a>
            </li>
            <li class="nav-item {% if  'example' in request.path
%}active{% endif %}">
                <a class="nav-link " href="{% url
'examples:examples_list' %}"><i class="fa fa-file-code-o"

aria-hidden="true"></i> Coding
                    examples</a>
            </li>
            <li class="nav-item {% if  'ce' in request.path %}active{%
endif %}">
                <a class="nav-link " href="{% url
'codingExercise:exercise_list' %}"><i class="fa fa-code"

aria-hidden="true"></i> Coding
                    exercises</a>
            </li>
            <li class="nav-item {% if  'quiz' in request.path
%}active{% endif %}">
                <a class="nav-link " href="{% url 'quiz:quiz_list'
%}"><i class="fa fa-question-circle-o"

aria-hidden="true"></i> Quizes</a>
            </li>
            <li class="nav-item {% if  'leaderboard' in request.path
```

```
%}active{% endif %}">
                <a class="nav-link " href="{% url
'leaderboard:leaderboard_list' %}"><i class="fa fa-list-ol"

aria-hidden="true"></i> Leaderboard</a>
            </li>


            {% if user.is_authenticated %}
                <li class="nav-item {% if 'profile' in request.path
%}active{% endif %}">
                    <a class="nav-link " href="{% url 'profile'
user.profile.id %}"><i class="fa fa-user-o"

aria-hidden="true"></i>
                         {{ user.get_username }}</a>


                </li>

                <li class="nav-item {% if 'progress' in request.path
%}active{% endif %}">

                    <a class="nav-link " href="{% url 'progress'
user.profile.id %}"> <i class="fa fa-list-alt"

aria-hidden="true"></i> My
                        Progress</a>
                </li>

                <li>
                    <a class="nav-link " href="{% url 'logout'
%}?next={{ request.path }}">Logout</a>
                </li>
            {% else %}
                <li><a class="nav-link" href="{% url 'login' %}?next={{
request.GET.next }}">Login</a></li>

                <li><a class="nav-link" href="{% url 'register'
%}">Register</a></li>
            {% endif %}
        </ul>

    </div>


</nav>
```

templates/account/login.html

```
{% extends 'base.html' %}


{% block extrahead %}

    {% load static %}

    <link rel="stylesheet" href="{% static 'css/form.css' %}">

{% endblock %}
```

```
{% block title %}Login{% endblock %}

{% block content %}


    <div class="container">


        <div class="login-container">
            <div id="output">
                {% for non_field_error in form.non_field_errors %}
                    <div id="signupalert" class="alert alert-danger">

                        {{ non_field_error }}

                    </div>
                {% endfor %}

            </div>


            <div class="avatar">
                <img alt="User Pic"
                     src="https://aminoapps.com/static/img/user-icon-
placeholder.png"
                     id="profile-image1" class="rounded-circle img-
fluid">

            </div>
            <div class="form-box">
                <form action="" method="post">
                    {% csrf_token %}
                    {% for field in form %}
                        <div class="form-group">
                            <label for="{{ field.id_for_label }}">{{
field.html_name }}</label>
                            <div>
                                {{ field }}
                            </div>
                            {% if field.help_text %}
                                <p>{{ field.help_text }}</p>

                            {% endif %}
                            {% for field_error in field.errors %}
                                <div id="signupalert" class="alert
alert-danger">

                                    {{ field_error }}

                                </div>
                            {% endfor %}
                        </div>
                    {% endfor %}
                    <button class="btn btn-primary btn-block login"
type="submit">Login</button>
                </form>
            </div>
```

```
        </div>

    </div>



{% endblock %}
```

templates/account/profile.html

```
{% extends 'base.html' %}

{% block title %}{{ profile.user.username }} {% endblock %}

{% block content %}


    <div class="container">


        <div class="card">
            <div class="card-header bg-dark  text-center text-white">
                <h4>

                    {% if user.get_username ==  profile.user.username
%}
                        My profile
                    {% else %}
                        {{ profile.user.username }}'s profile
                    {% endif %}

                </h4>
            </div>


            <div class="card-block">
                <div class="row">
                    <div class="col-md-4 col-xs-12 col-sm-6 col-lg-3">

                        <img alt="User Pic"
src="https://aminoapps.com/static/img/user-icon-placeholder.png"
                             id="profile-image1" class="rounded-circle
img-fluid">


                    </div>
                    <div class="col-md-8 col-xs-12 col-sm-6 col-lg-8">
                        <div class="container text-center">
                            <h2>{{ profile.points }} points</h2>


                        </div>
                        <hr>

                        <ul class="container details">
                            <div class="card-header bg-dark text-center
text-white">
```

```
                                <h4>User Info</h4>
                            </div>

                            <ul>

                                {% if profile.user.email %}
                                    <li>Email : {{ profile.user.email
}}</li>{% endif %}

                                <li>Date Of Joining: {{
user.date_joined }}</li>
                            </ul>


                        </ul>
                        <hr>

                        <ul class="container details">
                            <div class="card-header bg-dark text-center
text-white">
                                <h4>Completed Quizes</h4>
                            </div>

                            <ul>

                                {% if not profile.completedquiz_set.all
%}
                                    No completed  quizes.

                                {% endif %}

                                {% for quiz in
profile.completedquiz_set.all %}
                                    <li>
                                        Quiz: {{ quiz.quiz_name }} with
score {{ quiz.score }}% on {{ quiz.date }}
                                    </li>
                                {% endfor %}
                            </ul>


                        </ul>
                        <hr>

                        <ul class="container details">
                            <div class="card-header bg-dark text-center
text-white accordion">
                                <h4>Completed Exercises</h4>
                            </div>

                            <ul>

                                {% for exercise in completed_exercises
%}
                                    <li>
                                        {{ exercise }}
                                    </li>
                                {% endfor %}
```

```
                            </ul>


                        </ul>
                        <hr>

                        <ul class="container details">
                            <div class="card-header bg-dark text-center
text-white">
                                <h4>Achievements</h4>
                            </div>

                            <ul>
                                {% if not profile.user.badges.all %}
                                    No achievements.

                                {% endif %}

                                {% for badge in profile.user.badges.all
%}
                                    {% if badge.level == '1' %}

                                        <li>
                                            <i class="fa fa-trophy
level{{ badge.level }}" aria-hidden="true"></i>
                                            {{ badge.title }} : {{
badge.description }}
                                        </li>
                                    {% endif %}
                                {% endfor %}

                                {% for badge in profile.user.badges.all
%}
                                    {% if badge.level == '2' %}
                                        <li>
                                            <i class="fa fa-trophy
level{{ badge.level }}" aria-hidden="true"></i>
                                            {{ badge.title }} : {{
badge.description }}
                                        </li>
                                    {% endif %}
                                {% endfor %}
                                {% for badge in profile.user.badges.all
%}
                                    {% if badge.level == '3' %}
                                        <li>
                                            <i class="fa fa-trophy
level{{ badge.level }}" aria-hidden="true"></i>
                                            {{ badge.title }} : {{
badge.description }}
                                        </li>
                                    {% endif %}
                                {% endfor %}
                                {% for badge in profile.user.badges.all
%}
                                    {% if badge.level == '4' %}
                                        <li>
                                            <i class="fa fa-trophy
level{{ badge.level }}" aria-hidden="true"></i>
                                            {{ badge.title }} : {{
```

```
badge.description }}
                                        </li>
                                    {% endif %}
                                {% endfor %}


                            </ul>


                        </ul>

                    </div>
                </div>
            </div>

        </div>
    </div>


{% endblock %}

{% block scripts %}

    <script>
        $(function () {
            $(".accordion").accordion({
                collapsible: true
            });
        });
    </script>
{% endblock %}
```

templates/account/progress.html

```
{% extends 'base.html' %}

{% block content %}

    {% load badges_tags %}
    <div class="container">

        <h1 class="card-header bg-dark text-white text-center ">
Progress</h1>
        <ul class="list-group">
            {% for badge in badges %}

                <li class="list-group-item">
                    <div class="progressbar-title">
                          <i class="fa fa-trophy level{{ badge.level
}}" aria-hidden="true"></i>
                        {{ badge }} : {{ badge.description }}

                    </div>


                    <div class="progress">
                        <div class="progress-bar progress-bar-striped
progress-bar-animated"
                             style="width:calc(({{
badge|progress:profile.user }}/{{ badge.meta_badge.target }})*100%)">{{
```

```
badge|progress:profile.user }}/{{ badge.meta_badge.target }}</div>
                    </div>


                </li>


            {% empty %}

                <li>Nada</li>

            {% endfor %}

        </ul>

    </div>

{% endblock %}
```

templates/account/register.html

```
{% extends 'base.html' %}

{% block extrahead %}

    {% load static %}

    <link rel="stylesheet" href="{% static 'css/form.css' %}">

{% endblock %}


{% block title %}Register{% endblock %}

{% block content %}

    <div class="container">

        <div class="login-container">

            <div id="output">
                {% for non_field_error in form.non_field_errors %}
                    <div id="registeralert" class="alert alert-danger">

                        {{ non_field_error }}

                    </div>
                {% endfor %}

            </div>

            <div class="avatar">
                <img alt="User Pic"
                     src="https://aminoapps.com/static/img/user-icon-
placeholder.png"
                     id="profile-image1" class="rounded-circle img-
fluid">

            </div>
            <div class="form-box">
```

```
                <form action="" method="post">
                    {% csrf_token %}
                    {% for field in form %}
                        <div class="form-group">

                            <div>
                                {{ field }}
                            </div>
                            {% if field.help_text %}
                                <p>{{ field.help_text }}</p>

                            {% endif %}
                            {% for field_error in field.errors %}
                                <div id="registeralert" class="alert
alert-danger">

                                    {{ field_error }}

                                </div>
                            {% endfor %}
                        </div>
                    {% endfor %}
                    <button class="btn btn-primary btn-block login"
type="submit">Register</button>
                </form>
            </div>
        </div>

    </div>




{% endblock %}
```

templates/codingExercise/exercise.html

```
{% extends 'base.html' %}
{% block extrahead %}
    {{ form.media }}
{% endblock %}

{% block brython %}onload="brython()"{% endblock %}



{% block content %}



    <div class="container">
        <div class="jumbotron">
            <h1>{{ ex.name }}</h1>
            {{ ex.instructions|safe }}


        </div>
```

```
        <div class="card-deck">

            <div class="col-md-12 col-lg-6">
                <div class="card card-inverse card-primary  text-center
card h-100">
                    <h3 class="card-header bg-info text-white
">Editor</h3>
                    <div class="card-block">

                        <form id="codeForm" action="" method="post">
                            {% csrf_token %}
                            {{ form.text }}


                        </form>


                    </div>
                    <button id="runCode" class="btn btn-primary btn-
sm">Run code</button>
                </div>

            </div>







            <div class="col-md-12 col-lg-6">
                <div class="card card-inverse text-center  card h-100"
                     style="background-color: #333; border-color: #333;
color: white;">
                    <h3 class="card-header bg-danger text-
white">Console</h3>
                    <div class="card-block" id="console" style="height:
350px; overflow-y: auto;">


                    </div>
                </div>


            </div>
        </div>

        <div class="container completed-list">
            <div class="card-header bg-dark text-center text-white
accordion">
                <h4>Previous Solutions</h4>
            </div>


            <ul class="list-group">

                {% for exercise in
user.profile.completedexercise_set.all %}

                    {% if ex.name == exercise.exercise_name %}
```

```
                        <li class="list-group-item prev-solultion">
                            {{ exercise.answer|safe }}
                        </li>
                    {% endif %}

                {% empty %}
                    <li class="list-group-item">
                        No solutions yet
                    </li>
                {% endfor %}


            </ul>


        </div>


        <div class="alert alert-success" role="alert" id="success-bar">
            Congratulations!Move on to the next exercise
            <input class="btn btn-success" type="submit"
form="codeForm" id="next-exercise" value="Next">
        </div>


    </div>


{% endblock %}


{% block scripts %}

    {% load static %}

    <script src="{% static 'js/codingExercise/brython.js' %}"></script>
    <script src="{% static 'js/codingExercise/brython_stdlib.js'
%}"></script>
    <script src="{% static 'js/codingExercise/consoleRedirect.js'
%}"></script>
    <link
href="http://alexgorbatchev.com/pub/sh/current/styles/shThemeDefault.cs
s" rel="stylesheet" type="text/css"/>
    <script
src="http://alexgorbatchev.com/pub/sh/current/scripts/shCore.js"
type="text/javascript"></script>
    <script
src="http://alexgorbatchev.com/pub/sh/current/scripts/shAutoloader.js"
type="text/javascript"></script>

    <!--Run the test-->
    <script type="text/python">
        from browser import document, html

        #get test from context
        test="""{{ ex.test }}"""
        test_imports="""{{ ex.test_imports }}"""

        #fix the test string
        test= test.replace("&quot;","'")
```

```
        test_imports= test_imports.replace("&quot;","'")

        def run(ev):
            document['console'].clear()
            #get user code and test code and append them
            user_code = document["id_text"].value
            myCode = test_imports
            myCode += "\n"
            myCode += user_code
            myCode += "\n"
            myCode += test
            #run the code
            exec(myCode)
            #if the test is passed allow the user to progress
            if "Correct" in  document['console'].text and "Correct" not
in user_code:
                document['success-bar'].style.display='block'

        #bind the run function the the button runCode
        document['runCode'].bind('click',run)

    </script>
{% endblock %}
```

templates/codingExercise/exercise_list.html

```
{% extends 'base.html' %}

{% block content %}


    <div class="container">
        {% for chapter in object_list %}
            <div class="chapter-container">
                <h1 class="card-header bg-dark text-white text-center
"> {{ chapter }}</h1>
                <ul class="list-group">

                    {% for exercise in chapter.exercise_set.values %}
                        <li class="list-group-item">
                            {% if exercise.name in completed_exercises
%}
                                <i class="fa fa-check-circle-o fa-2x"
aria-hidden="true" style="color: green;"></i>
                            {% else %} 

                                <i class="fa fa-question-circle-o fa-
2x" aria-hidden="true" style="color: orange;"></i>
                            {% endif %} 

                            <a href="{% url 'codingExercise:exercise'
pk=exercise.id %}">{{ exercise.name }} </a>

                        </li>
```

```
                    {% empty %}
                        <li>No exercises yet.</li>
                    {% endfor %}

                </ul>
            </div>
        {% endfor %}

    </div>

{% endblock %}
```

templates/codingExercise/finished.html

```
{% extends 'base.html' %}

{% block content %}



    <div class="container">


        <div class="card-block">
            <div class="row">

                <ul class="container details">
                    <div class="card-header bg-success text-center
text-white accordion">
                        <h1>Congratulations</h1>
                    </div>

                    <ul class="list-group">


                        <li class="list-group-item">
                            <h3> You have completed this section !!!
</h3>
                        </li>


                    </ul>


                </ul>

            </div>
        </div>
    </div>


{% endblock %}
```

templates/examples/example.html

```
{% extends 'base.html' %}


{% block title %}{{ example.title }}{% endblock %}

{% block content %}

    <div class="container">
        <div class="jumbotron">
            <h1>{{ example.chapter }} : {{ example.title }}</h1>

            {{ example.content|safe }}


        </div>
        <form action="" method="post">
            {% csrf_token %}

            <div class="row">
                <div class="col-sm-12 offset-sm-1 text-center">
                    {#                    <a href="{% url
'examples:examples_list' %}">#}
                    <button class="btn btn-primary"
type="submit">Next</button>
                    {#                </a>#}
                </div>
            </div>
        </form>


    </div>

{% endblock %}


{% block scripts %}

    <script src="https://cdn.rawgit.com/google/code-
prettify/master/loader/run_prettify.js"></script>

{% endblock %}
```

templates/examples/example_list.html

```
{% extends 'base.html' %}

{% block content %}


    <div class="container">
        {% for chapter in object_list %}
            <div class="chapter-container">
                <h1 class="card-header bg-dark text-white text-center
"> {{ chapter }}</h1>
                <ul class="list-group">
```

```
                    {% for example in chapter.example_set.values %}
                        <li class="list-group-item">

                            <a href="{% url 'examples:example'
pk=example.id %}">{{ example.title }} </a>

                        </li>




                    {% empty %}
                        <li class="list-group-item">No examples
yet.</li>
                    {% endfor %}

                </ul>
            </div>
        {% endfor %}

    </div>

{% endblock %}
```

templates/leaderBoard/leaderboard_list.html

```
{% extends 'base.html' %}

{% block title %}Leaderboard {% endblock %}


{% block content %}

    <div class="container">
        {% if object_list %}
            <table class="table table-striped table-bordered">
                <tr class="bg-dark text-white text-center">
                    <th>#</th>
                    <th>User</th>
                    <th>Points</th>
                </tr>
                {% for userProfile in object_list %}
                    <tr {% if userProfile.user.get_username ==
user.get_username %}class="bg-success text-white" {% endif %}>
                        <td></td>
                        <td>
                            <a class="leader-link"
                               href="{% url 'profile' userProfile.id
%}">{{ userProfile.user.get_username }}</a>
                        </td>
                        <td>
                            {{ userProfile.points }}
                        </td>
                    </tr>
                {% endfor %}
            </table>

            {% if is_paginated %}
```

```
                <ul class="pagination pagination-lg justify-content-
center">

                    <li class="page-item{% if not page_obj.has_previous
%} disabled{% endif %}">
                        <a class="page-link" href="
                                {% if  page_obj.has_previous %}?page=
                            {{ page_obj.previous_page_number }}{% else
%}#{% endif %}"
                        >  </a>
                    </li>

                    <li class="page-link">

                        {{ page_obj.number }} of {{
page_obj.paginator.num_pages }}

                    </li>

                    <li class="page-item   {% if not page_obj.has_next
%} disabled{% endif %}">
                        <a class="page-link"
                           href="{% if  page_obj.has_next %}?page=
                               {{ page_obj.next_page_number }}{% else
%}#{% endif %}"> 
                            </a>
                    </li>


                </ul>

            {% endif %}
        {% else %}

            <p>No users found</p>
        {% endif %}


    </div>


{% endblock %}

{% block scripts %}
    <script type="text/javascript">
        //add serial number to the list
        (function () {

            $('table tr').each(function (index) {
                $(this).find('td:nth-child(1)').html(index +
(parseInt({{ page_obj.number }}) - 1) * parseInt({{
page_obj.paginator.per_page }}));
                console.log(index)
            });
        })()


    </script>

{% endblock %}
```

templates/quiz/quiz.html

```
{% extends 'base.html' %}

{% block extrahead %}

    {% load static %}

    <link rel="stylesheet" href="{% static 'css/quiz.css' %}">


{% endblock %}

{% block title %}{{ quiz.name }}{% endblock %}

{% block content %}

    <div class="container">


        <form action="" method="post">
            {% csrf_token %}
            {% for question in quiz.question_set.all %}


                <div class="modal-dialog">
                    <div class="modal-content">
                        <div class="modal-header">
                            <h3 class="question-text-box">{{
question.question_text|safe }}</h3>
                        </div>


                        <div class="quiz" id="quiz" data-
toggle="buttons">
                            <label class=" btn btn-lg btn-primary btn-
block"><span class="btn-label"></span><input
                                    type="radio"
                                    name="{{ question.question_text }}"
                                    value="{{ question.answer1 }}"
                                    required
                                    value="1">{{ question.answer1 }}
                            </label>

                            <label class=" btn btn-lg btn-primary btn-
block"><span class="btn-label"></span><input
                                    type="radio"
                                    name="{{ question.question_text }}"
                                    value="{{ question.answer2 }}">{{
question.answer2 }}
                            </label>

                            <label class=" btn btn-lg btn-primary btn-
block"><span class="btn-label"></span><input
                                    type="radio"
                                    name="{{ question.question_text }}"
                                    value="{{ question.answer3 }}">{{
question.answer3 }}
```

```
                                </label>


                            </div>
                        </div>

                    </div>


                {% endfor %}

                <div class="row">
                    <div class="col-sm-12 offset-sm-1 text-center">

                        <input class="btn btn-lg btn-success" type="submit"
value="Submit">

                    </div>

                </div>



            </form>


        </div>


{% endblock %}

{% block scripts %}

     <script src="https://cdn.rawgit.com/google/code-
prettify/master/loader/run_prettify.js"></script>

{% endblock %}
```

templates/quiz/quiz_list.html

```
{% extends 'base.html' %}

{% block content %}

    <div class="container">

        {% for chapter in object_list %}
            <div class="chapter-container">

                <h1 class="card-header bg-dark text-white text-center
"> {{ chapter }}</h1>


                <ul class="list-group">

                    {% for quiz in chapter.quiz_set.values %}

{#                            <a href="{% url 'quiz:quiz_details'
pk=quiz.id %}">{{ quiz.name }}</a>#}
```

```
{#                          {{ user.profile.completedquiz_set.values }}#}
{#                           {% if quiz.name ==  q.quiz_name %}#}
{#                           {% for q in
user.profile.completedquiz_set.values %}#}

                              <li class="list-group-item">

                                  {% if quiz.name in  completed_quizes %}
                                      <i class="fa fa-check-circle-o fa-
2x" aria-hidden="true" style="color: green;"></i>

                                  {% else %}
                                      <i class="fa fa-question-circle-o
fa-2x" aria-hidden="true"
                                         style="color: orange;"></i>
                                  {% endif %} 
                                  <a href="{% url 'quiz:quiz_details'
pk=quiz.id %}">{{ quiz.name }}</a>


                              </li>

{#                          {% endfor %}#}

                      {% empty %}
                          <li class="list-group-item">No Quiz yet for
this chapter.</li>
                      {% endfor %}
                  </ul>
              </div>
          {% endfor %}


    </div>

{% endblock %}
```

templates/quiz/results.html

```
{% extends 'base.html' %}

{% block extrahead %}

    {% load static %}

    <link rel="stylesheet" href="{% static 'css/quiz.css' %}">

{% endblock %}


{% block title %}Results{% endblock %}


{% block content %}

    <div class="container">
```

```
        <div class="card-deck">

            <div class="card card-inverse card-primary mb-3 text-
center">
                <h3 class="card-header bg-success text-white ">Correct
Answers</h3>
                <div class="card-block">

                    {% for k,v in correct %}
                        <p class="card-text">The correct answer for
question {{ k|safe }} is {{ v }}</p>

                    {% endfor %}

                </div>
            </div>

            <div class="card card-outline-primary text-center ">
                <h3 class="card-header bg-warning text-white">Your
Answers</h3>
                <div class="card-block">

                    {% for k,v in userAnswers %}
                        <p class="card-text">Your answer for question
{{ k|safe }} was {{ v }}</p>

                    {% endfor %}

                </div>
            </div>

        </div>



        <div class="card card-outline-success card-inverse text-center
">
            <h3 class="card-header bg-info text-white">Your Score</h3>
            <div class="card-block">


                <p class="card-text">
                    You scored {{ correct_percentage }}%
                    {% if newHighsScore and not first_attempt %}which
is an improvement over your previous score of
                        {{ prevScore }}
                        %{% endif %}
                    {% if not newHighsScore and not first_attempt
%}which is not an improvement over your previous score
                        of
                        {{ prevScore }}%{% endif %}
                </p>


            </div>
        </div>
```

```
        <div class="row">
            <div class="col-sm-12 offset-sm-1 text-center">

                <a href="{% url 'home' %}">
                    <button class="btn btn-primary">Return to
Home</button>
                </a>

            </div>

        </div>




        <div class="row">
            <div class="col-sm-12 offset-sm-1 text-center">

                <a href="{% url 'quiz:quiz_list' %}">
                    <button class="btn btn-primary">More
quizes</button>
                </a>


            </div>

        </div>


    </div>

{% endblock %}
```

Παράρτημα Β - Εκπαιδευτικό Υλικό

Σε αυτό το παράρτημα περιλαμβάνεται το εκπαιδευτικό υλικό της εφαρμογής.

Τίτλος Κεφαλαίου : Basics

Τίτλος Ενότητας : Hello World

```
The first thing you need to know is how to display information.The
easiest way to do it in Python is the built-in print function and it
can be used like this:



print("This is displayed in the console")
```

Τίτλος Κεφαλαίου : Basics

Τίτλος Ενότητας : Variables

```
Variables are reserved memory used to store data.Variable names can't
contain spaces but only letters,numbers and underscores( _ ) and the
first character of the name can't be a number. Some valid variable
names are:




name = "John"
age = 24
phone_num = "222-555-888"
phone_num_2 = "222-555-999"
_blood_type = "A-"



       A variable that starts with an underscore is conventionally
considered private,that means it shouldn't be accessed directly by
users.

In python you don't have to explicitly state the type of a variable and
it is created the moment you assign a value to it.
The type of a variable can change.


x = 1  #x is of type int
x  = 24 #now it is of type str
```

Τίτλος Κεφαλαίου : Basics

Τίτλος Ενότητας : Data types

The standard data types supported in Python are numbers(int, float,complex), strings, lists,tuples and dictionaries.

```
an_int = 5
a_float = 3.14
a_string = "This is a string"
a_list = [4,"data",5.6]
a_tuple = (4,"data",5.6)
a_dict = {name:"John",age:24}
```

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Ενότητας : Lists

Lists contain multiple items separated by commas inside square brackets and they may contain items of different type.
Lists can be changed and may contain duplicate items.

```
a_list = [4,"data",5.6]
```

To access the values of the list you can use the index or indices to slice a portion of the list. The first item has index 0.

```
print(a_list[0])
print(a_list[0:2])
```

The first print statement will log to the console the value <code>4 and the second statement will log the value <code>[4,"data"]

You can get the length of a list with the function <code>len().

```
print(len(a_list)) #this will return 3
```

You can add items to a list with <code>append(some_item).

```
a_list.append("a new item")
```

You can update the values in a list.

```
print(a_list[0])#it will print 4
a_list[0] = 15
print(a_list[0])#now it will print 15
```

Lastly the <code>+ (concatenation)and <code>* (repetition) operators can be used to create lists.

```
concat_list = [1,2]+["a","b"] #concat_list=[1,2,"a","b"]
repl_list = ["Hello"]*4 # repl_list = ["Hello","Hello","Hello","Hello"]
```

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Ενότητας : Tuples

Tuples are sequences just like lists with the main difference being that tuples are immutable. Tuples use parentheses instead of square brackets.

```
one_item_tuple=(1,)
my_tuple=( "a" , 1 , 5.0 , "b" , 7 )
```

Accessing values in a tuple is the same as with lists.

```
my_tuple=( "a" , 1 , 5.0 , "b" , 7 )
print(my_tuple[0]) #prints the value 'a'
print(my_tuple[1:3]) #prints the value (1,5.0)
```

The <code>len() function also works on tuples.

```
print(len(my_tuple)) #this will return 5
```

The <code>+ (concatenation)and <code>* (repetition) operators can be used to create new tuples.

```
concat_tuple = (1,2)+("a","b") #concat_tuple=(1,2,"a","b")
```

```
repl_tuple = ("Hello")*4 # repl_tuple =
("Hello","Hello","Hello","Hello")
```

Tuples are immutable, that means that you can NOT update the values in a tuple, add a new one or delete one of the existing ones.

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Ενότητας : Dictionaries

Dictionaries consist of key-value pairs. Keys are usually numbers or strings and the values can be any type of Python object. The key is separated from it's value by a colon (:) and the pairs are separated by commas.

```
person={"first_name":"John",
               "last_name":"Doe",
               "age":38}
```

To access the values of a dictionary instead of indices you use the keys.

```
print(person["first_name"]) #prints "John"
```

If you try to access a value with a key that doesn't exist you will get a <code>KeyError message.

You can update or add new items to dictionaries.

```
person["age"] = 39 #changes the value from 38 to 39
person["job"] = "Programmer" # adds the new key:value pair
"job":"Programmer" to the dictionary
```

Τίτλος Κεφαλαίου : Functions

Τίτλος Ενότητας : Functions

Functions are reusable blocks of code that provides modularity to your applications.

A function is defined by it's name and parentheses containing parameters.

```
def test_func():
        print("Hello,this is a test")

def func_with_input(func_input1,unc_input2):
        print("input 1 is: ".join(func_input1))
        print("input 2 is: ".join(func_input2))
```

To call a function use it's name followed by parentheses containing the arguments it expects.

```
test_func()
func_with_input("this","that")
```

Functions can also return values with the keyword <code>return. The return statement stops the execution of the function and continues the execution of the program from the point the function was called.

```
def increase_by_10(num):
        return num + 10
```

If nothing is explicitly returned the <code>None is returned by default.

There are a few ways arguments can be passed to functions. The most simple is passing the arguments in the order they are defined in the definition of the function as shown in previous examples. These are referred to as Required or Positional Arguments.

Another way is using Keyword Arguments. This method allows you to pass the arguments out of order.

```
def car_info(brand,model):
        print("Car brand: " + brand)
        print("Car model: " + model)

car_info(model = "Focus", brand = "Ford")
```

This allows us to give default values in case the user hasn't provided value for an argument.That is called a Default Argument.

```
def car_info(brand,model = "not defined"):
        print("Car brand: " + brand)
        print("Car model: " + model)
```

```
car_info(brand = "Ford")
```

Positional and Keyword arguments can be combined but the Keyword arguments MUST follow the positional.

```
def foo(p1,p2,p3):
        #some code


foo(1,2,3) #correct call
foo(1,2,p3=3) # also correct call
foo(1,p2=2,3) # will raise an exception
```

Lastly sometimes you may need to process a function that has a variable number of arguments.

```
def parser(command , *args):
        print("Command: " + command)
        print("arguments: " + *args)
```

In this example the first parameter will be put in the variable <code>command and all the rest in the tuple <code>*args. The name of the tuple must start with an asterisk(*) and it's conventionally named args.

With the introduction of functions the concept of local and global variables also arise. Variables created outside of functions are global while those created inside a function are local and can only be accessed inside the function.

```
result = 0 # global variable

def mult(num1,num2):
        result = num1 * num2
        print(result) # the result of multiplying num1 by num2 will be
printed
                                return result

mult(2,10)
print(result) # this will print 0
```

Τίτλος Κεφαλαίου : Control Flow

Τίτλος Ενότητας : If

In Python like in all programming languages you can execute different code based on conditions.

In Python any non-zero and non-null values are evaluated as True

There are different ways to use an if structure.

An if statement witch consists of a boolean expression followed by one or more statements.

```
if CONDITION:
        #code to execute
```

An if/else statement that is an if statement followed by an else statement that executes when the boolean expression is false.

```
if CONDITION:
        # code to execute if CONDITION == True
else:
        # code to execute if CONDITION == False
```

An if/elif/else statement

```
if CONDITION_1:
        # code to execute if CONDITION_1 == True
elif CONDITION 2:
        # code to execute if CONDITION_2 == True
else:
        # code to execute if CONDITION_1 == False and CONDITION_2 ==
False
```

The conditions can be a single boolean expression or combination of boolean expressions.The boolean operators are <code>and , or , not.

```
if CONDITION_1 and CONDITION_2:
        # this code will execute if both True

if CONDITION_1 or CONDITION_2:
```

```
        # this code will execute if one of the conditions is True

if not CONDITION_1 :
        # this code will execute if CONDITION_1 is False
```

Some more examples:

```
if num1 > num2:
        # block of code
elif num1 < num2:
        # block of code

if num3 >= num4:
        # block of code
else:
        # block of code

if string1 == string2:
        # block of code
```

Τίτλος Κεφαλαίου : Loops

Τίτλος Ενότητας : For

A loop statement allows us to execute a block of code multiple times.Python has two types of loops, for loops and while loops.

A for loop can be used to iterate through values in any of the different data structures. It can also be used to iterate through the characters of a string.

```
list_of_nums = [1,5,6]
for num in list of nums:
        print(num)
```

The output the block above will give are the numbers 1, 5 and 6 each a new line.

```
my_string = 'This is a string'
for letter in my_string:
        print(letter)
```

For loops can be used to iterate through dictionaries.

```
person={"first_name":"John",
                "last_name":"Doe",
                "age":38}

# this for will print all the keys
for item in person:
        print(item)

# this for will print tuples where the first item is the key and the
second the corresponding value
for item in person.items():
        print(item)

# this for will print key : value pairs.
for key,value in person.items():
        print('{} : {}'.format(key,value))
```

The <code>break statement can be used to exit a loop at any time.

```
list_of_nums = [1,5,6]
for num in list_of_nums:
        print(num)
        if num == 5:
                print('found 5')
                break
```

The output the block above will give are the numbers 1,5 and 'found 5'. It will not print 6 since the break condition is met and the loop is exited early.

The <code>continue statement can be used to stop the execution of a loop and return from the top.

```
list_of_nums = [1,5,6]
for num in list_of_nums:
        if num == 5:
                continue
        print(num)
```

The output the block above will give are the numbers 1 and 6.It will not print the number 5 since the continue condition is met when num == 5 and the rest of the loop body (the print statement is this case) is not executed.

Τίτλος Κεφαλαίου : Loops

Τίτλος Ενότητας : While

The other type of loop supported in python is the while.

```
while CONDITION:
        # loop body
```

The body of a while loop executes while a condition is True.

```
count = 0
while count < 5:
        print(count)
        count +=1
```

That loop will be executed until the value of the count is 5 of higher and the it will stop.

Τίτλος Κεφαλαίου : Strings

Τίτλος Ενότητας : Strings

Strings are a basic data type supported in Python. Single line strings can be inside a pair single or double quotation marks. For multiple line strings a pair of triple single or double quotation marks are used.

```
string1 = 'a string'
string2 = "also a string"
string3 = ''' multiple line
                        string
'''
string4 = """ another multiple
                        line string
"""
```

Specific characters and substrings of a string can be accessed as if the string is a list.

```
string1 = 'a string'
string2 = "also a string"
print(string1[0]) #this prints 'a'
print(string2[0:4]) #this prints 'also'
```

<code>format() is used to print formated strings.

```
string1 = 'a string'
print('this is {}'.format(string1))#this will print 'this is a string'
```

The concatenation operator can also be used to achieve the same result but be careful all the variables are string. Non string variables can be turned to strings using <code>str(var).

```
num = 5
print('this is ' + str(num)) #correct statement
print('this is ' + num) #FALSE statement
```

Τίτλος Κεφαλαίου : Strings

Τίτλος Ενότητας : Useful string methods

Some useful built-in methods are:

<code>len(str): returns the length of str

<code>str.lower(): turns all uppercase characters to lowercase

<code>str.upper(): turns all lowercase characters to uppercase

<code>str.capitalize(): capitalizes the first character of a string

<code>str.isupper():returns true if the srting contains at least one character and all the characters of a string are uppercase

<code>str.islower():returns true if the srting contains at least one character and all the characters of a string are lowercase


```
                        <code>str.count(sub_str): counts how many times
sub_str appears in a string
                </li>
                <li>
                        <code>str.isalnum(): returns true if the srting
contains at least one character and all characters are alphanumeric
                </li>
                <li>
                        <code>str.isalpha(): returns true if the srting
contains at least one character and all characters are alphabetic
                </li>
                <li>
                        <code>str.isdigit()): returns true all
characters are numeric
                </li>


        </ul>
```

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Ενότητας : Classes

```
Python is an object-oriented programming language. Objects are data
structures that combine data and functionality.

        Creating classes in Python is very easy.



class MyFirstClass:
        # class specific code
```

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Ενότητας : The constructor

```
The <code>__init__() is a special method that is called a class
constructor. It is the method that is called when an object, a new
instance of the class, is created.



class Person:

        def __init__(self,name):
                self.name=name

john=Person('John')
```

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Ενότητας : Methods and the self argument

```
Whenever a method is called from an object the object adds its self as
the first parameter. For that reason in the definition of our methods
we include the argument self first and we include it even in methods
that don't take any other arguments.
```

```
class Person:

        def __init__(self,name):
                self.name=name

        def print_info(self):
                print(self.name)


john=Person('John')
john.print_info()
```

In this example the method print_info is called and the <code>self argument is used to access it's name and display it to the user.

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Ενότητας : Inheritance

Inheritance is at the core of object-oriented programming. A class can inherit from any other class. That means the transfer of characteristics from one class to another.

```
class Person:

        def __init__(self,name):
                self.name=name

        def print_info(self):
                print('Name: 'self.name)

class Student(Person):

        def   init  (self,name,sid):
                Person.__init__(self,name)
                self.student_id = sid

        def print_info(self):
                print('Name: '+ self.name + ' Id: ' +
str(self.student_id))

john=Person('John')
john.print_info()
mary = Student('Mary',001)
mary.print_info()
```

In this example the class Student inherits from the class Person.

In the sub-class' constructor we call the parent-class' constructor to take care of the common data members and then we handle the ones specific to the sub-class.

The method <code>print_info() is also overwritten. When it's called from the instance named john it will print <code>Name: John but when it's called from the instance named mary it will print <code>Name: Mary Id: 001.

Παράρτημα C - Προγραμματιστικές Ασκήσεις

Σε αυτό το παράρτημα περιλαμβάνονται οι προγραμματιστικές ασκήσεις της εφαρμογής και τα τεστ που χρησιμοποιούνται για την ορθότητα της λύσης.

Τίτλος Κεφαλαίου : Basics

Τίτλος Άσκησης : Hello World

The print function is used to display messages to the console. Use the print function to show the message "Hello World!"

```
import io
import sys

import builtins

def print(*args, **kwargs):
    capturedOutput = io.StringIO()  # Create StringIO object
    sys.stdout = capturedOutput  # and redirect stdout.
    builtins.print(*args, **kwargs)
    sys.stdout = sys.__stdout__
    builtins.print(capturedOutput.getvalue().strip(), **kwargs)
    if capturedOutput.getvalue().strip() == "Hello World!":
        builtins.print("Correct", **kwargs)
    else:
        builtins.print("Try again", **kwargs)
```

Τίτλος Κεφαλαίου : Basics

Τίτλος Άσκησης : Hello World

Variables are reserved memory used to store data. The standard type of data in Python are numbers, strings, lists, tuples and dictionaries. To assign a value to a variable the equals operator is used. Name a variable 'my_var' and give it any value you want.

```
def var_test():
    try:
        my_var
    except NameError:
        print("You haven't declared a variable named my_var")
        return False
    return True
```

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Άσκησης : Lists

Lists contain multiple items separated by commas and they may contain items of different type.

Create a list named "my_list" and put 3 items in it.

example:

<pre class="python">

a_list=[1,"john",5.7,"8"]

```
def test_my_list():
    try:
        my list
        if type(my_list) == list:
            if len(my_list) == 3:
                print("Correct")
            else:
                print("Your list doesn't contain 3 items")
        else:
            print("my_list doesn't appear to be a list")
    except NameError:
        print("You haven't named a list my_list")
```

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Άσκησης : Lists(2)

You can access a lists specific element using it's index. The first element has index 0. Given a list named "colors" print it's second element.

example:

<pre class="python">

my_var= my_list[7]

```
import io
import sys

import builtins

colors = ["green", "yellow", "red"]
```

```
def print(*args, **kwargs):
    capturedOutput = io.StringIO()  # Create StringIO object
    sys.stdout = capturedOutput  # and redirect stdout.
    builtins.print(*args, **kwargs)
    sys.stdout = sys.__stdout__
        builtins.print(capturedOutput.getvalue().strip(), **kwargs)
    if capturedOutput.getvalue().strip() == "yellow":
        builtins.print("Correct", **kwargs)
    else:
        builtins.print("Try again", **kwargs)
```

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Άσκησης : List length

You can get the size of a list using "len".Given a list named colors print it's size.

example:

<pre class="python">

size= len(my_list)

```
import io
import sys

import builtins

colors = ["green", "yellow", "red"]


def print(*args, **kwargs):
        capturedOutput = io.StringIO()  # Create StringIO object
    sys.stdout = capturedOutput  # and redirect stdout.
    builtins.print(*args, **kwargs)
    sys.stdout = sys.__stdout__
        builtins.print(capturedOutput.getvalue().strip(), **kwargs)
    if capturedOutput.getvalue().strip() == "3":
        builtins.print("Correct", **kwargs)
    else:
        builtins.print("Try again", **kwargs)
```

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Άσκησης : Adding to a list

You can add more items to an existing list by using append. Add the color "blue" to the list colors.

example:

<pre class="python">

my_list.append(new_item)

```
colors = ["green", "yellow", "red"]

def test list append():
    if len(colors) >= 4:
        print("Correct")
    else:
        print("You haven't appended one more item to the list colors ")
```

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Άσκησης : Tuples

Tuples are sequences just like lists with the main difference being that tuples are immutable. Tuples use parentheses

instead of square brackets. Create a tuple named "my_tuple" and give it some values.

example:

<pre class="python">

one_item_tuple=(1,)

a_tuple=("a",1)

```
def test_my_tuple():
    try:
        my_tuple
        if type(my_tuple) == tuple:
            print("Correct")
        else:
```

```
            print("my_tuple is not a tuple")

    except NameError:
        print("You haven't named a tuple my_tuple")
```

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Άσκησης : Dictionaries

Dictionaries consist of key-value pairs. Keys are usually numbers or strings and the values can be any type of Python object.

Create a dictionary named "grades" with values:

<code>"Jonshon":78 ,

<code>"Kramer":52 ,

<code>"Mosby":89.

example:

<code>

a_dict={1:"green",2:"yellow"}

or

<pre class="python">

a_dict={}

a_dict[1]="green"

a_dict[2]="yellow"

```
def test_grades_dict():
    try:
        grades
        if type(grades) == dict:
            if grades["Jonshon"]==78 and grades["Kramer"]==52 and
grades["Mosby"]==89:
                print("Correct")
            else:
                print("You didn't use the correct key-value pairs")
        else:
            print("grades doesn't appear to be a dictionary")


    except NameError:
        print("You haven't named a dictionary grades")
```

Τίτλος Κεφαλαίου : Functions

Τίτλος Άσκησης : Functions

Functions are reusable blocks of code that provides modularity to your applications.

Define a function named "add" that takes two arguments and returns their sum

example:

<pre class="python">

def foo(a):

"""your code goes here"""

return ___

Τίτλος Κεφαλαίου : Functions

Τίτλος Άσκησης : Functions(2)

You will be using plenty of functions so lets make sure you got it.

Define a function named 'sub' that takes two arguments and returns their difference.

example:

<pre class="python">

def foo(a):

"""your code goes here"""

return ___

```
def subTest():
    try:
        sub
        if sub(4, 5) == -1:
            print("Correct")
        else:
            print("Your funtion doesn't appear to return the correct
result")
            print("Expected result for input (4,5) is -1  and " +
str(sub(4, 5)) + " was returned")
    except NameError:
        print("You haven't declared a function named sub that takes tow
arguments")
```

Τίτλος Κεφαλαίου : Control Flow

Τίτλος Άσκησης : If

In Python like in all programming languages you can execute different code based on conditions.

In Python any non-zero and non-null values are evaluated as True.

Create a function named "odd_even" that takes an integer as argument and returns "odd" if the number is odd and

"even" if the number is even.

example:

<pre class="python">

def foo():

if some_condition :

#do something

else:

#do something else

```
def test_odd_even():
    try:
        odd_even
        if odd_even(3) == "odd" and odd_even(16) == "even":
            print("Correct")
        else:
            print("Your funtion doesn't appear to return the correct
result")
            print("Expected result for input 3 is 'odd'  and " +
str(odd_even(3)) + " was returned")
            print("Expected result for input 16 is 'even'  and " +
str(odd_even(16)) + " was returned")
    except NameError:
        print("You haven't declared a function named odd_even that
takes one argument argument")
```

Τίτλος Κεφαλαίου : Control Flow

Τίτλος Άσκησης : If(2)

There are different ways to use an if structure.

An if statement witch consists of a boolean expression followed by one or more statements.

An if/else statement that is an if statement followed by an else statement that executes when the boolean expression is false.

An if/elif/else statement

Create a function named "fizzBuzz" that takes an integer as argument and returns "fizz" if the number can is evenly divisible by

2,"buzz" if the number is evenly divisible by 5 , "fizzbuzz" if it is evenly divisible by both 2 and 5 and 'fail' if it

is not evenly divisible by ether.

example:

<pre class="python">

def foo():

if some_condition :

#do something

elif:

#do something else

else:

#some other code

```
def test_fizzBuzz():
    try:
        fizzBuzz
        correct = fizzBuzz(2) == "fizz" and fizzBuzz(5) == "buzz" and
fizzBuzz(10) == "fizzbuzz" and fizzBuzz(33) == "fail"
        if correct:
            print("Correct")
        else:
            print("Your funtion doesn't appear to return the correct
result")
            print("Expected result for input 2 is 'fizz'  and " +
fizzBuzz(2) + " was returned")
            print("Expected result for input 5 is 'buzz'  and " +
fizzBuzz(5) + " was returned")
```

```
            print("Expected result for input 10 is 'fizzbuzz'  and " +
fizzBuzz(10) + " was returned")
            print("Expected result for input 33 is 'fail'  and " +
fizzBuzz(33) + " was returned")

    except NameError:
        print("You haven't declared a function fizzBuzz  that takes an
integer argument")
```

Τίτλος Κεφαλαίου : Loops

Τίτλος Άσκησης : Loops

Loops allows us to execute a block of code multiple times.In Python you can use for loops or while loops.

Write a program that prints the numbers from 1 to 5.

Hint:To generate a list from 1 to 5 use the builtin function range(1,6).

Hint2:Or you could use a while loop instead!

examples:

<pre class="python">

for i in some_list:

  #do something with i

while j<5:

  #execute this code

for i in range(50,75):

  #do something with i

```
import io
import sys

import builtins

output = []


def print(*args, **kwargs):

    capturedOutput = io.StringIO()  # Create StringIO object
    sys.stdout = capturedOutput  # and redirect stdout.
    builtins.print(*args, **kwargs)
    sys.stdout = sys.__stdout__  # Reset redirect.
```

```
    builtins.print(capturedOutput.getvalue().strip(), **kwargs)
    value = capturedOutput.getvalue().strip()
    output.append(value)


if output == ["1","2","3","4","5"]:
    builtins.print("Correct")
else:
    print("Wrong output,try again!")
```

Τίτλος Κεφαλαίου : Loops

Τίτλος Άσκησης : Loops(2)

Make a function named "div_by_5and7" that returns a list with the numbers between 1 and 100 that are evenly divisible by 5 and 7.

Hint:To generate a list from 1 to 100 use the builtin function range(1,101). Hint2:Or you could use a while loop instead!

```
def test_div_by_5and7():
    try:
        div_by_5and7
        if div_by_5and7() == [35, 70]:
            print("Correct")
        else:
            print("Your funtion doesn't appear to return the correct
result")
            print("Expected result is [7,35]  and " +
str(div_by_5and7()) + " was returned")
    except NameError:
        print("You haven't declared a function named div_by_5and7 ")
```

Τίτλος Κεφαλαίου : Loops

Τίτλος Άσκησης : Loops(3)

You can loop through dictionaries as well. Given a dictionary named "contacts" print its key-value pairs fromatted like

this: <code>key : value

example:

<pre class="python">

for k,v in some_dict.items():

  #do something

```
import io
import sys

import builtins

output = []


def print(*args, **kwargs):

    capturedOutput = io.StringIO()  # Create StringIO object
    sys.stdout = capturedOutput  # and redirect stdout.
    builtins.print(*args, **kwargs)
    sys.stdout = sys.__stdout__  # Reset redirect.

    builtins.print(capturedOutput.getvalue().strip(), **kwargs)
    value = capturedOutput.getvalue().strip()
    output.append(value)

if output == ["John : john@fakemail.com","Angela :
ang321@fakemail.com"]:
    builtins.print("Correct")
else:
    print("Wrong output,try again!Expected result was John :
john@fakemail.com","Angela : ang321@fakemail.com")
```

Τίτλος Κεφαλαίου : Loops

Τίτλος Άσκησης : List Overlap

Make a function named "list_overlap" that takes two parameters list_a and list_b and returns a list that contains their common

elements. Make sure the result list is sorted and doesn't contain duplicate items.

The easiest way to eliminate duplicate items is to turn a list into a set like this:

<pre class="python">

set_from_list = set(some_list)

You can likewise turn sets to lists like this:

<pre class="python">

list_from_set = list(some_set)

To sort a list you can use the function sort.

<pre class="python">

some_list.sort()

```
def test_list_overlap():
    try:
        list_overlap
        if list_overlap([],[1,2]) == [] and list_overlap([1,3,1],[1,2])
== [1] and list_overlap([1,3,2,1],[1,2]) == [1,2]:
            print("Correct")
        else:
            print("Your funtion doesn't appear to return the correct
result")
            print("Expected result for input [],[1,2]  is []  and " +
str(list_overlap([],[1,2])) + " was returned")
            print("Expected result for input [1,3,1],[1,2]  is [1]  and
" + str(list_overlap([1,3,1],[1,2])) + " was returned")
            print("Expected result for input([1,3,2,1],[1,2]  is [1,2]
and " + str(list_overlap([1,3,2,1],[1,2])) + " was returned")
    except NameError:
        print("You haven't declared a function named list_overlap")
```

Τίτλος Κεφαλαίου : Strings

Τίτλος Άσκησης : Swap

Make a function named "swap" that takes as an argument a string and returns the a new string with the first and last

characters of the original string swapped.

Hint:You can take a substring of a string using indices ex: s[3:5] will get the 3rd and 4rth characters of string s

Hint2:You can get the last character of a string two ways 1) s[-1] 2) s[len(s)]

```
def test_swap():
    try:
        swap
        if swap("word") == "dorw":
            print("Correct")
        else:
            print("Your funtion doesn't appear to return the correct
```

```
result")
            print("Expected result for input 'word'  is 'dorw'  and " +
swap("word") + " was returned")
    except NameError:
        print("You haven't declared a function named swap")
```

Τίτλος Κεφαλαίου : Strings

Τίτλος Άσκησης : Reverse string

Create a function named "reverse_string" that takes as an argument a string and returns it's reverse.

```
def test_reverse_string():
    try:
        reverse_string
        if reverse_string("word") == "drow" and reverse_string("hello")
== "olleh":
            print("Correct")
        else:
            print("Your function doesn't appear to return the correct
result")
            print("Expected result for input 'word'  is 'drow'  and " +
reverse_string("word") + " was returned")
            print("Expected result for input 'hello'  is 'drow'  and "
+ reverse_string("olleh") + " was returned")
    except NameError:
        print("You haven't declared a function named reverse_string")
```

Τίτλος Κεφαλαίου : Strings

Τίτλος Άσκησης : Longest word

Make a function named "longest_word" that takes as an argument a list of words and returns the length of the longest one.

```
def test_longest_world():
    try:
        longest_word
        if longest_word(["a", "aa", "v"]) == 2:
            print("Correct")
        else:
            print("Your funtion doesn't appear to return the correct
result")
            print("Expected result for input ['a','aa','v']  is 2  and
" + str(
                longest_word(["a", "aa", "v"])) + " was returned")
    except NameError:
        print("You haven't declared a function named longest_word ")
```

Τίτλος Κεφαλαίου : Strings

Τίτλος Άσκησης : Split a string

Make a function named "split_string" that takes as an argument a string and a character, splits it at the specific character and returns
the first part of the string.

```
def test_split_string():
    try:
        split_string
        if split_string("song1.mp3", ".") == "song1" and
split_string("www.myblog.com/news", "/") == "www.myblog.com":
            print("Correct")
        else:
            print("Your function doesn't appear to return the correct
result")
            print("Expected result for input 'song1.mp3','.'  is
'song1'  and " + split_string("song1.mp3",

".") + " was returned")
            print("Expected result for input 'www.myblog.com/news','/'
is ''www.myblog.com'  and " + split_string(
                "www.myblog.com/news", "/") + " was returned")
    except NameError:
        print("You haven't declared a function named split_string")
```

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Άσκησης : Classes

Python is an object-oriented programming language. Objects are data structures that combine data and functionality in the
form of methods, methods are called the functions of an object.

By making classes you can create new types of objects.

A simple class definition looks like this:

<pre class="python">
class MyFirstClass:
    #class specific code

Classes can also inherit from other classes (more on that later):

<pre class="python">
class MySecondClass(MyFirstClass):
    #class specific code

Define a class named "Car" and put the keyword pass as a placeholder in the body of the class so your program can run.

```
def test_car_class_exists():
    try:
        Car
        if  type(Car)==type:
            print("Correct")
        else:
            print("You haven't defined a class named Car")
```

```
    except NameError:
        print("You haven't defined a class named Car")
```

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Άσκησης : Classes:The __init__ method

The init method is a special method a class may contain that acts as the constructor when you create a new instance
of the class

A simple class definition looks like this:

<pre class="python">
class MyFirstClass:
    def __init__(self,value_a,value_b):
        self.a = value_a
        self.b = value_b

a_class_object = MyFirstClass("something","something_else")

Once again define a class named "Car". This time instead of leaving the body create a constructor like it's described previously.
A Car object should contain information about the <code>color,model and <code>max_speed of the car.

```
def test_car_class_exists():
    try:
        Car
        if type(Car)==type:
            a_car=Car("black","ford",160)
            if hasattr(a_car,"color") and hasattr(a_car,"model") and
hasattr(a_car,"max_speed"):
                print("Correct")
            else:
                print("Your Car class doesn't have all required
attributes")
        else:
            print("You haven't defined a class named Car")


    except NameError:
        print("You haven't defined a class named Car")
```

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Άσκησης : Classes:Methods and the self argument

Whenever a method is called from an object the object adds its self as the first parameter. For that reason in

the definition of our methods we include the argument self first and we include it even in methods that don't take any
other arguments.

example:
<pre class="python">
```
class MyFirstClass:
    def __init__(self,value_a,value_b):
        self.a = value_a
        self.b = value_b

    def add_to_a(self,c):
        return self.a + c

a_class_object = MyFirstClass(1,2)
print(a_class_object.add_to_a(5))
```

Define a method for the class Car named "info" that takes no arguments and prints the information of the car formated like
the following example:

Color:Black,Model:Ford Fiesta,Max Speed:160 km/h

The code of the Car class so far:
<pre class="python">
```
class Car:
    def __init__(self,color,model,max_speed):
        self.color = color
        self.model = model
        self.max_speed = max_speed
```

```
import builtins
import sys
import io

def print(*args, **kwargs):
        capturedOutput = io.StringIO()# Create StringIO object
    sys.stdout = capturedOutput    #  and redirect stdout.
    builtins.print(*args ,**kwargs)
    sys.stdout = sys.__stdout__# Reset redirect.
    builtins.print(capturedOutput.getvalue().strip(),**kwargs)
    if capturedOutput.getvalue().strip() == "Color:Black,Model:Ford Fiesta,Max Speed:160 km/h":
        builtins.print("Correct",**kwargs)
    else:
        builtins.print("Try again",**kwargs)

def test_car_class_exists():
    try:
        Car
        if type(Car)==type:
            a_car=Car("Black","Ford Fiesta",160)
```

```
            if hasattr(a_car,"info"):
                a_car.info()
            else:
                print("Your Car class doesn't have a method named
info")
        else:
            print("You haven't defined a class named Car")


    except NameError:
        print("You haven't defined a class named Car")

test_car_class_exists()
```

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Άσκησης : Classes:Inheritance

A class can inherit from any other class. This means that an instance of the sub-class has all the data and method attributes
of the parent-class.The syntax is like that:

example:
<pre class="python">
class ParentClass:
   def __init__(self,valueA):
      self.a=valueA

   def parent_method(self):
      #some code

class SubClass(ParentClass):
   def __init__(self,valueA,valueB):
      super(ParentClass, self).__init__(valueA)
      self.b=valueB

The instances sub-class will also have the method attribute "parent_method".

Inheritance allows us to create a hierarchy of more general to more specific classes.

Assume the class Car is already implemented like described in previous exercises. Create a sub-class named "SuperCar". In
the case of a super car we also want to know in how many seconds it can reach 100 km/h (name the attribute <code>zero_to_100).

Reminder the class Car has three data attributes (color , model , max_speed) and one method attribute (info)

```
class Car:
    def __init__(self,color,model,max_speed):
        self.color = color
        self.model = model
        self.max_speed = max_speed

    def info(self):
        print("Color:{},Model:{},Max Speed:{}
km/h".format(self.color,self.model,self.max_speed))

def test_supercar_class_exists():
    try:
        SuperCar
        if type(Car) == type:
            if issubclass(SuperCar,Car):
                a_car = SuperCar("Black", "Ford Fiesta", 160,3.2)
                if hasattr(a_car, "zero_to_100"):
                   print("Correct")
                else:
                    print("Your SUperCar class doesn't have an
attribute named zero_to_100")
            else:
                print("There is no class named SuperCar that is a sub-
class of Car")
        else:
            print("You haven't defined a class named Car")


    except NameError:
        print("You haven't defined a class named SuperCar")
```

Παράρτημα D - Τεστ Πολλαπλής Επιλογής

Σε αυτό το παράρτημα περιλαμβάνονται τα τεστ ερωτήσεων πολλαπλής επιλογής της εφαρμογής.

Τίτλος Κεφαλαίου : Basics

Τίτλος Τεστ : Basics

Question: How do you display text to the console in Python?

A1: printf('Hello')

A2: print('Hello')

A3: console.log('Hello')

CA: A2: print('Hello')

Question: Which is not a valid variable name in Python?

A1: _var2

A2: var_2

A3: 2var

CA: A3: 2var

Question: Which is a valid variable name in Python?

A1: _foo

A2: __foo__

A3: both

CA: A3: both

Question: What does variable name starting with an underscore indicate?

A1: It's a class' private variable

A2: It holds a string value

A3: It doesn't mean anything

CA: A1: It's a class' private variable

Question: Which is not a core data type?

A1: string

A2: float

A3: class

CA: A3: class

Τίτλος Κεφαλαίου : Data Structures

Τίτλος Τεστ : Lists

Question: Which command creates a new empty list?

A1: new_list = []

A2: new_list = list()

A3: both

CA: A3: both

Question: A list can contain ...

A1: one type of data (eg. only strings)

A2: up to two types of data (eg. strings and numbers)

A3: multiple types of data

CA: A3: multiple types of data

Question: Which is a list?

A1: ['a',2,False]

A2: {'a',2,False}

A3: ('a',2,False)

CA: A1: ['a',2,False]

Question: What is the value of l[2] given l=['a', 0, 5.8, False]?

A1: 0
A2: False
A3: 5.8
CA: A1: A3: 5.8

Question: What does len(['a','b','c']) return?

A1: 2
A2: 3
A3: len is not a valid function
CA: A2: 3

Question: Which is the slicing operator?

A1: :
A2: |
A3: ?
CA: A1: :

Τίτλος Κεφαλαίου : Data Structures
Τίτλος Τεστ : Tuples

Question: Which is a tuple?

A1: ['a',2,False]
A2: {}
A3: ('a',2,False)
CA: A3: ('a',2,False)

Question: What is the difference between tuples and lists?

A1: There is no defference

A2: Tuples can not be updated

A3: Tuples contain unique items

CA: A2: Tuples can not be updated

Question: What is the result of (1,4,2,3,5).sort() ?

A1: (1,4,2,3,5)

A2: (1,2,3,4,5)

A3: Tuples don't have a sort attribute

CA: A3: Tuples don't have a sort attribute

Question: What type is (5) ?

A1: integer

A2: tuple

A3: both integer and tuple

CA: A1: integer

Τίτλος Κεφαλαίου : Datastructures

Τίτλος Τεστ : Dictionaries

Question: Which statement(s) creates a dictionary ?

A1: new_dict = {}

A2: new_dict = {'name':'John','age':25}

A3: both

CA: A3: both

Question: Which are the keys in the dictionary d = {'name':'John','age':25} ?

A1: 'name','John','age' and 25
A2: 'name' and 'age'
A3: 'John' and 25
CA: A2: 'name' and 'age'

Question: Which of these about a dictionary is false?

A1: The values of a dictionary can be accessed using keys
A2: The keys of a dictionary can be accessed using values
A3: Dictionaries aren't ordered
CA: A2: The keys of a dictionary can be accessed using values

Question:Given d = {'name':'John','age':25} what will happen if you try to access d['job'] ?

A1: A KeyError exception will be raised
A2: A syntax error will be raised
A3: The statement will execute and None will be returned
CA: A1: A KeyError exception will be raised

Question: What is the output?

```
person={"first_name":"John",
        "last_name":"Doe",
        "age":38}
        person["age"] = 39
        person["job"] = "Programmer"
print('Age: {} Job: {}'.format(person["age"],person["job"]))
```

A1: 'Age: 38 Job: Programmer'

A2: 'Age: 39 Job: Programmer'

A3: An error will be thrown

CA: A2: 'Age: 39 Job: Programmer'

Τίτλος Κεφαλαίου : Functions

Τίτλος Τεστ : Functions

Question:A function definition contains ...

A1: It's name

A2: A list of parameters

A3: both

CA: A3: both

Question:A function ...

A1: can have no parameters

A2: must have one parameter

A3: must have at least one parameter

CA: A1: can have no parameters

Question:Which is a correct function call?

A1: foo[a]

A2: foo(a)

A3: foo{a}

CA: A2: foo(a)

Question: What does this function return?

```
def foo():
    print("Hello")
```

A1: String

A2: None

A3: null

CA: A2: None

Question: Which function call is incorrect given a function foo(p1,p2,p3)?

A1: foo(1,2,3)

A2: foo(1,2,p3=3)

A3: foo(1,p2=2,3)

CA: A3: foo(1,p2=2,3)

Question: A variable defined inside a function is referred to as ...

A1: A global variable

A2: A local variable

A3: A fuction variable

CA: A2: A local variable

Τίτλος Κεφαλαίου : Control Flow

Τίτλος Τεστ : If

Question:What is the output ?

```
if True:
    print(1)
else:
    print(2)
```

A1: 1

A2: True

A3: 2

CA: A1: 1

Question:What is the output ?

```
if False:
```

```
        print(1)
elif True :
        print(2)
else:
        print(3)
```

A1: 1

A2: 3

A3: 2

CA: A3: 2

Question:What is the output ?

```
if 2>4:
        print(1)
```

A1: 1

A2: 2

A3: Nothing is printed

CA: A3: Nothing is printed

Question:What is the output ?

```
if 2>4:
        print(1)
else:
        print(2)
```

A1: 1

A2: 2

A3: Nothing is printed

CA: A2: 2

Τίτλος Κεφαλαίου : Loops

Τίτλος Τεστ : Loops

Question: How many times will the print statement be executed?

```
c=0
while c < 10:
        print(c)
        c += 1
```

A1: 9

A2: 10

A3: 11

CA: A2: 10

Question: How many times will the print statement be executed?

```
 for c in range(10):
        print(c)
```

A1: 9

A2: 10

A3: 11

CA: A2: 10

Question: Which numbers will be printed?

```
list_of_nums = [1,2,5,6,7]
for num in list_of_nums:
        print(num)
        if num == 5:
                break
```

A1: 1,2

A2: 1,2,5,6,7

A3: 6,7

CA: A1: 1,2

Question: Which numbers will be printed?

```
list_of_nums = [1,2,5,6,7]
for num in list_of_nums:
if num == 5:
continue
print(num)
```

A1: 1,2,5

A2: 5,6,7

A3: 1,2,6,7

CA: A3: 1,2,6,7

Τίτλος Κεφαλαίου : Strings

Τίτλος Τεστ : Strings

Question:Which is not a string literal?

A1: "False"

A2: 'False'

A3: False

CA: A3: False

Question:What is the output ?

```
 s='Hello'
print(s[1])
```

A1: 'H'

A2: 'e'

A3: 'He'

CA: A2: 'e'

Question:What is the output ?

```
s='Hello'
print(s.upper())
```

A1: 'hello'

A2: 'hELLO'

A3: 'HELLO'

CA: A3: 'HELLO'

Question:What is the output ?

s = 'Hello'

n = 5

print(s + 5)

A1: 'Hello 5'

A2: 'Hello5'

A3: This is not a legal statement

CA: A3: This is not a legal

Question:What function gives the length of a string?

A1: size(str)

A2: length(str)

A3: len(str)

CA: A3: len(str)

Τίτλος Κεφαλαίου : Object-Oriented Programming

Τίτλος Τεστ : Classes

Question:Whith which keyword does the definition of a class begin?

A1: def

A2: class

A3: return

CA: A2: class

Question:What is the name of the constructor method?

A1: __begin__

A2: __start__

A3: __init__

CA: A3: __init__

Question:An object is an instance of a ...

A1: class

A2: method

A3: variable

CA: A1: class

Question:What is conventionally the name of the first parameter of a method ?

A1: this

A2: self

A3: there is no conventional name for the first parameter of a method

CA: A2: self

Question:What is the output ?

```
class A:

        def __init__(self,x):
                self.x=x

        def print_info(self):
                print('x={}'.format(x))

class B(A):
        def __init__(self,x,y):
                A.__init__(self,x)
                        self.y = y
        def print_info(self):
```

```
            print('x={},y={}'.format(x,y))
            b = B(1,2)
    b.print_info()
```

```
A1: x=1
A2: y=2
A3: x=1,y=2
CA: A3: x=1,y=2
```