Ε.Α.Π./ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

2η ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ

ΑΚΑΔΗΜΑΪΚΟΥ ΕΤΟΥΣ 2021-2022

22/11/2021

Ημερομηνία παράδοσης εργασίας: Κυριακή 16/01/2022
Καταληκτική ημερομηνία παραλαβής: Τετάρτη[1] 19/01/2022
Ημερομηνία ανάρτησης ενδεικτικών λύσεων: Σάββατο 22/01/2022
Καταληκτική ημερομηνία αποστολής σχολίων στον φοιτητή: Κυριακή 06/02/2022

ΥΠΟΕΡΓΑΣΙΑ 1. (βαθμοί 25)
Λίστες, Δομημένος Προγραμματισμός

ΥΠΟΕΡΓΑΣΙΑ 2. (βαθμοί 25)
Πλειάδες, Δομημένος Προγραμματισμός

ΥΠΟΕΡΓΑΣΙΑ 3. (βαθμοί 25)
Σύνολα, Δομημένος Προγραμματισμός

ΥΠΟΕΡΓΑΣΙΑ 4. (βαθμοί 25)
Λεξικά, Δομημένος Προγραμματισμός

ΣΥΝΟΛΟ **(βαθμοί 100)**

[1] Σύμφωνα με τον Κανονισμό Σπουδών, η καταληκτική ημερομηνία για την παραλαβή της Γ.Ε. από το μέλος ΣΕΠ είναι η επόμενη Τετάρτη από το τέλος της εβδομάδας παράδοσης Γ.Ε.

ΥΠΟΕΡΓΑΣΙΑ 1. (βαθμοί 25)

Να υλοποιηθεί κώδικας σε Python που δεχόμενος τις κατάλληλες παραμέτρους, δημιουργεί μία τυχαία λίστα και υπολογίζει τη μεγαλύτερη απόσταση μεταξύ των στοιχείων της. Για το σκοπό αυτό, ακολουθήστε τα παρακάτω βήματα:

α) Να ορίσετε τη συνάρτηση `generate_random_floats()` η οποία να δέχεται ως ορίσματα τρεις ακέραιες τιμές `n`, `m` και `seed` και να επιστρέφει μια λίστα με `n` τυχαίες πραγματικές τιμές, ομοιόμορφα επιλεγμένες στο εύρος τιμών από `-m` μέχρι `m`, έχοντας χρησιμοποιήσει ως «σπόρο» παραγωγής των τυχαίων τιμών την τιμή `seed`. Η παράμετρος `seed` να είναι προαιρετικό όρισμα. Σε περίπτωση που δεν δοθεί, τότε να χρησιμοποιείται ως `seed` η προκαθορισμένη τιμή που ορίζει η συνάρτηση παραγωγής τυχαίων τιμών της τυπικής βιβλιοθήκης της Python, `random`. Σημειώνεται ότι `seed` (σπόρος) είναι ένας αριθμός που αρχικοποιεί μια γεννήτρια ψευδοτυχαίων αριθμών. Αν χρησιμοποιηθεί το ίδιο `seed` για την παραγωγή δύο ίσου μήκους ακολουθιών τυχαίων αριθμών, τότε οι δύο αυτές ακολουθίες θα περιέχουν με την ίδια σειρά, τις ίδιες τιμές. Για παράδειγμα, στην Python, η εύρεση ενός τυχαίου πραγματικού αριθμού στο εύρος τιμών [-100,100], χρησιμοποιώντας ως `seed` την τιμή `1234`, βρίσκεται με τον ακόλουθο κώδικα:

```
>>> import random
>>> random.seed(1234)
>>> random.uniform(-100, 100)
93.29070713842776
```

Διαδοχικές κλήσεις της `random.uniform(-100, 100)` θα επιστρέψουν τυχαίες τιμές στο διάστημα `-100, 100`. Η ακολουθία των τιμών είναι πάντα η ίδια για τη συγκεκριμένη τιμή του seed.

β) Να ορίσετε τη συνάρτηση `find_max_gap` η οποία θα δέχεται ως παράμετρο μια λίστα τιμών, να την ταξινομεί σε αύξουσα σειρά και να επιστρέφει τη μεγαλύτερη διαφορά ανάμεσα σε δύο διαδοχικές τιμές της λίστας. Θεωρήστε ότι η λίστα τιμών έχει τουλάχιστον δύο τιμές και ότι δεν απαιτείται έλεγχος στη συνάρτηση για αυτό.

γ) Να ορίσετε τη συνάρτηση `present_list` που δέχεται ως όρισμα μια λίστα τιμών, και την τυπώνει όπως στο παράδειγμα:

Η τυχαία λίστα είναι: `466.45, -59.27, -492.51, 410.98, 439.27`

δ) Το πρόγραμμα να ζητά από τον χρήστη να εισαγάγει τον αριθμό `n` των τυχαίων τιμών που θα δημιουργηθούν, το εύρος των τιμών (ο χρήστης θα δίνει το `m`), και το `seed`. Να εφαρμοστεί

αμυντικός προγραμματισμός έτσι ώστε το `n` να είναι ακέραιος αριθμός μεγαλύτερος του `1`, το `m` να είναι θετικός ακέραιος αριθμός και το `seed` να είναι ακέραιος αριθμός. Το `seed` σε περίπτωση που λάβει την τιμή μηδέν δεν θα χρησιμοποιείται για την αρχικοποίηση της ακολουθίας των τυχαίων τιμών.

ε) Να καλεί τη συνάρτηση `generate_random_floats()` με παραμέτρους τις τιμές που θα έχει εισαγάγει ο χρήστης στο ερώτημα (δ). Η λίστα τυχαίων τιμών που παράγονται να χρησιμοποιείται ως όρισμα στην κλήση της συνάρτησης `find_max_gap`. Η τιμή που επιστρέφει η `find_max_gap` να εμφανίζεται στην οθόνη.

Υπόδειξη: Να χρησιμοποιηθεί το αρχείο **1_code_template.py** ως οδηγός επίλυσης.

ΥΠΟΕΡΓΑΣΙΑ 2. (βαθμοί 25)

Μια νέα αεροπορική εταιρία χαμηλού κόστους αποφασίζει να κάνει μια μελέτη για σύνδεση όλων των διεθνών αεροδρομίων της χώρας κατά τη θερινή περίοδο.

Η κατάσταση με τα διεθνή αεροδρόμια της χώρας και τις γεωγραφικές συντεταγμένες τους είναι η εξής:

```
Alexandroupoli     40.855869°N 25.956264°E
Athens             37.936389°N 23.947222°E
Chania             35.531667°N 24.149722°E
Chios              38.343056°N 26.140556°E
Corfu              39.601944°N 19.911667°E
Heraklion          35.339722°N 25.180278°E
Kalamata           37.068333°N 22.025556°E
Kavala             40.913333°N 24.619167°E
Kefalonia          38.12°N 20.500278°E
Kos                36.793336°N 27.091667°E
Lemnos             39.917072°N 25.236308°E
Mytilene           39.0567°N 26.5994°E
Paros              37.020833°N 25.113056°E
Rhodes             36.405419°N 28.086192°E
Samos              37.6891°N 26.9116°E
Thessaloniki       40.519722°N 22.970833°E
Zakynthos          37.750833°N 20.884167°E
```

Στοιχεία από την wikipedia, https://en.wikipedia.org/wiki/List_of_airports_in_Greece

Ζητείται να κατασκευαστεί εφαρμογή η οποία θα εμφανίζει μενού με τις ακόλουθες επιλογές:

1) Επιλογή δύο αεροδρομίων για τον υπολογισμό της απόστασής τους σε χιλιόμετρα (ο χρήστης εισάγει τους αριθμούς δύο αεροδρομίων χωρισμένους με κόμμα)

2) Εύρεση ζεύγους αεροδρομίων με την ελάχιστη απόσταση, εμφάνιση αεροδρομίων και της ελάχιστης απόστασης (ο χρήστης εισάγει την λέξη min)

Η εφαρμογή θα τερματίζει αν ο χρήστης απλά πατήσει <enter>.

Μια τυπική αλληλεπίδραση με την εφαρμογή είναι η εξής:

```
Επιλέξτε δύο αεροδρόμια i,j για υπολογισμό της απόστασής τους ή min για ελάχιστη απόσταση
1 Alexandroupoli, 2 Athens, 3 Chania, 4 Chios, 5 Corfu, 6 Heraklion, 7
```

```
Kalamata, 8 Kavala, 9 Kefalonia, 10 Kos, 11 Lemnos, 12 Mytilene, 13 Paros, 14
Rhodes, 15 Samos, 16 Thessaloniki, 17 Zakynthos
Επιλογή:min
Η ελάχιστη απόσταση είναι μεταξύ των αεροδρομίων Kefalonia-Zakynthos (53.1km)

Επιλέξτε δύο αεροδρόμια i,j για υπολογισμό της απόστασής τους ή min για
ελάχιστη απόσταση
1 Alexandroupoli, 2 Athens, 3 Chania, 4 Chios, 5 Corfu, 6 Heraklion, 7
Kalamata, 8 Kavala, 9 Kefalonia, 10 Kos, 11 Lemnos, 12 Mytilene, 13 Paros, 14
Rhodes, 15 Samos, 16 Thessaloniki, 17 Zakynthos
Επιλογή:2,16
Η απόσταση μεταξύ αεροδρομίων Athens και Thessaloniki είναι 299.40km

Επιλέξτε δύο αεροδρόμια i,j για υπολογισμό της απόστασής τους ή min για
ελάχιστη απόσταση
1 Alexandroupoli, 2 Athens, 3 Chania, 4 Chios, 5 Corfu, 6 Heraklion, 7
Kalamata, 8 Kavala, 9 Kefalonia, 10 Kos, 11 Lemnos, 12 Mytilene, 13 Paros, 14
Rhodes, 15 Samos, 16 Thessaloniki, 17 Zakynthos
Επιλογή:
>>>
```

Δίδεται η συνάρτηση `distance(lat1, lon1, lat2, lon2)` η οποία δέχεται ως ορίσματα 4 τιμές, το γεωγραφικό πλάτος `lat1` και το γεωγραφικό μήκος `lon1` ενός τόπου καθώς και το γεωγραφικό πλάτος `lat2` και το γεωγραφικό μήκος `lon2` ενός άλλου τόπου και επιστρέφει την απόσταση μεταξύ των δύο τόπων με βάση τον τύπο Haversine. Εάν επιθυμείτε να μάθετε περισσότερα για τον υπολογισμό αποστάσεων πάνω σε σφαίρα και τον τύπο Haversine, ανατρέξτε στη σελίδα https://en.wikipedia.org/wiki/Haversine_formula.

```
def distance(lat1, lon1, lat2, lon2):
    R = 6373.0  # ακτίνα της γης σε km
    lat1, lon1 = radians(lat1), radians(lon1)
    lat2, lon2 = radians(lat2), radians(lon2)
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat / 2) ** 2 + cos(lat1) * cos(lat2) * sin(dlon / 2) ** 2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    d = R * c
    return d
```

Οδηγίες για ανάπτυξη της εφαρμογής

(α) Να ορίσετε μια συμβολοσειρά `airport_data = '''...'''`, η οποία περιέχει τα παραπάνω στοιχεία των διεθνών αεροδρομίων της χώρας.

Στη συνέχεια, να δημιουργήσετε μια συνάρτηση `process_airports()` η οποία δημιουργεί μια λίστα `airports`, που περιέχει πλειάδες κάθε μια από τις οποίες έχει ως πρώτο στοιχείο το όνομα του αεροδρομίου και ως δεύτερο και τρίτο τις γεωγραφικές συντεταγμένες τους ως πραγματικούς αριθμούς.

(β) Να δημιουργήσετε μια συνάρτηση `menu()` η οποία θα τυπώνει την υπόδειξη προς τον χρήστη για επιλογή δύο αεροδρομίων ώστε να υπολογιστεί η απόστασή τους, ή τη λέξη `min` για υπολογισμό της ελάχιστης απόστασης αεροδρομίων, ή <enter> για έξοδο από το πρόγραμμα.

Στη συνέχεια, να τυπωθούν τα αεροδρόμια όπως στο υπόδειγμα. Η συνάρτηση αυτή θα εφαρμόζει αμυντικό προγραμματισμό και επαναλαμβάνει την υπόδειξη αν ο χρήστης δεν δώσει επιτρεπτή επιλογή.

Η συνάρτηση επιστρέφει την είσοδο του χρήστη αν δώσει "min" ή πατήσει <enter>, ή μια πλειάδα δύο ακεραίων αν δώσει δύο αεροδρόμια.

(γ) Να δημιουργήσετε μια συνάρτηση `min_distance()` η οποία βρίσκει τα πλησιέστερα αεροδρόμια και την απόστασή τους, και τυπώνει κατάλληλο μήνυμα.

(δ) Να δημιουργήσετε το κυρίως πρόγραμμα το οποίο αρχικά καλεί τη συνάρτηση `process_airports` και στη συνέχεια καλεί τις συναρτήσεις `menu`, `distance` και `min_distance` ανάλογα με την επιλογή του χρήστη.

Υπόδειξη: Να χρησιμοποιηθεί το αρχείο **2_code_template.py** ως οδηγός επίλυσης.

ΥΠΟΕΡΓΑΣΙΑ 3. (βαθμοί 25)

Μια επιχείρηση διατηρεί τα παρακάτω στοιχεία για το προσωπικό της:

```
"""Γιώργος Γεωργίου,m,eng,project1 project3
 Μαρία Ρήγα,f,eng,project2
 Κατερίνα Σελή,f,secr,project1 project2
 Νίκος Πάλης,m,tech,project2
 Λίνα Πενταγιά,f,eng,project1
 Ρένα Ντορ,f,secr,project3 project2
 Τζον Κλης,m,tech,project1 project2
 Λάκης Λαζός,m,eng,project2
 Μαρίνα Μαρή,f,eng,project3
 Ζήσης Χελάς,m,tech,project1 project2"""
```

Η παραπάνω συμβολοσειρά πολλών γραμμών περιέχει τα εξής στοιχεία για κάθε εργαζόμενο της επιχείρησης:

(α) το ονοματεπώνυμό του,

(β) το φύλο του (τιμές `m` για άνδρας και `f` για γυναίκα),

(γ) την κατηγορία εργαζομένου (τιμές `eng`=μηχανικός, `secr`=διοικητικός υπάλληλος, `tech`=τεχνικός),

(δ) τα έργα (projects) της επιχείρησης στα οποία απασχολείται (τιμές: `project1`, `project2`, `project3`).

Ζητείται να οργανώσετε τις πληροφορίες της δοθείσας συμβολοσειράς σε σύνολα (sets). Στη συνέχεια, με χρήση των συνόλων αυτών να απαντήσετε σε τυπικά ερωτήματα.

α) Να δημιουργήσετε συνάρτηση `load_sets()`, η οποία επεξεργάζεται τα στοιχεία της συμβολοσειράς, και γεμίζει τα σύνολα `m`, `f`, `eng`, `tech`, `secr`, `p1`, `p2`, `p3` με τα ονόματα των εργαζομένων που έχουν την αντίστοιχη ιδιότητα, π.χ. το σύνολο `m` περιέχει τα ονόματα όλων των ανδρών εργαζόμενων της επιχείρησης. Με κλήση της βοηθητικής συνάρτησης `show_set()` που περιγράφεται στη συνέχεια, να τυπωθούν τα σύνολα αυτά.

β) Να δημιουργήσετε συνάρτηση `show_set(hint, s)` που παρουσιάζει τα στοιχεία ενός συνόλου `s` με την επεξηγηματική επικεφαλίδα που περιέχεται στη συμβολοσειρά `hint`.

γ) Να υλοποιήσετε πρόγραμμα που με χρήση της συνάρτησης `show_set` εκτυπώνει τα αρχικά σύνολα `m`, `f`, `eng`, `tech`, `secr`, `p1`, `p2`, `p3`. Με χρήση του περιεχομένου των συνόλων αυτών και τη συνάρτηση `show_set` να απαντηθούν τα εξής ερωτήματα:

γ1) ποιοι άνδρες δουλεύουν στο `project1`;

γ2) ποιοι εργαζόμενοι δουλεύουν στο `project1` αλλά όχι στο `project2` ή `project3`;

γ3) ποιες γυναίκες είναι μηχανικοί;

γ4) ποιοι τεχνικοί δουλεύουν είτε στο `project1` ή στο `project2`;

γ5) ποιοι άνδρες μηχανικοί δεν δουλεύουν στο `project2`;

Υπόδειξη: Να χρησιμοποιηθεί το αρχείο **3_code_template.py** ως οδηγός επίλυσης.

ΥΠΟΕΡΓΑΣΙΑ 4. (βαθμοί 25)

Ζητείται να υλοποιηθεί πρόγραμμα που επεξεργάζεται ένα κείμενο που δίνεται ως συμβολοσειρά πολλαπλών γραμμών, και στη συνέχεια, επιστρέφει στατιστικά στοιχεία για τη συχνότητα χαρακτήρων και χρήση λέξεων στο κείμενο με βάση την παρακάτω προδιαγραφή.

α) Να δημιουργήσετε τη βοηθητική συνάρτηση `capitalize_keep_only_en_chars(word)` η οποία δέχεται στην είσοδο μια λέξη και την επιστρέφει με κεφαλαία γράμματα αφού αγνοήσει χαρακτήρες εκτός Αγγλικού αλφαβήτου (σημεία στίξης, κ.λπ.). Για παράδειγμα, για είσοδο `"one--"` να επιστρέφει τη λέξη `"ONE"`.

β) Να δημιουργήσετε μια συνάρτηση `tokenize(txt)` που επεξεργάζεται το κείμενο `txt` και επιστρέφει μια λίστα με όλες τις λέξεις του κειμένου με κεφαλαία Αγγλικά γράμματα και χωρίς άλλα σύμβολα (χρησιμοποιήστε τη συνάρτηση `capitalize_keep_only_en_chars`). Να σημειωθεί ότι ο διαχωρισμός λέξεων γίνεται με βάση τα κενά που υπάρχουν στο κείμενο.

γ) Να δημιουργήσετε συνάρτηση `char_frequencies(words)`, η οποία δέχεται ως είσοδο μια λίστα λέξεων `words` και επιστρέφει ένα λεξικό με κλειδιά τους χαρακτήρες που συναντώνται στις λέξεις και τιμές το πλήθος εμφανίσεων του αντίστοιχου χαρακτήρα στο κείμενο.

δ) Να δημιουργήσετε συνάρτηση `word_frequencies(words)` που δέχεται ως είσοδο μια λίστα λέξεων `words` και επιστρέφει ένα λεξικό με κλειδιά τις λέξεις και τιμές το πλήθος επαναλήψεων της κάθε λέξης στη λίστα εισόδου.

ε) Εφαρμογή των παραπάνω συναρτήσεων:

Η εφαρμογή καλεί αρχικά τη συνάρτηση `tokenize` με όρισμα εισόδου το ακόλουθο κείμενο.

```
The Zen of Python, by Tim Peters
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
```

```
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

ε1) Στη συνέχεια, καλείται η συνάρτηση `char_frequencies()` για τη λίστα λέξεων του κειμένου και επιστρέφει το πλήθος εμφάνισης κάθε χαρακτήρα. Να τυπώνεται η αλφαβητική σειρά των γραμμάτων και το ποσοστό επί τοις εκατό εμφάνισής τους έναντι του συνολικού πλήθους εμφανίσεων όλων των γραμμάτων του αλφαβήτου στο κείμενο.

ε2) Στη συνέχεια, με χρήση της συνάρτησης `word_frequencies()` να εμφανίσετε κατάλογο με τις 10 λέξεις με περισσότερα από 3 γράμματα, που εμφανίζονται με μεγαλύτερη συχνότητα. Αν υπάρχουν λέξεις με την ίδια συχνότητα με τη 10η συχνότερη λέξη, να εμφανίζονται και αυτές στη λίστα των λέξεων.

Ένα δείγμα της αναμενόμενης εξόδου είναι:

```
Συχνότητα εμφάνισης γραμμάτων
A: 7.83%
B: 3.10%
C: 2.51%
...
Z: 0.15%
Λέξεις με πάνω από 3 γράμματα
1 BETTER 8
2 THAN 8
...
```

Υπόδειξη: Να χρησιμοποιηθεί το αρχείο **4_code_template.py** ως οδηγός επίλυσης.

Γενικές Υποδείξεις:

I) Για τις απαντήσεις της εργασίας μπορείτε να ανατρέξετε στη συμπληρωματική βιβλιογραφία που δίνεται και στα βοηθητικά κείμενα που υπάρχουν στον δικτυακό τόπο / portal της θεματικής ενότητας. Συνιστάται να προσθέσετε στο τέλος της εργασίας σας κατάλογο βιβλιογραφίας.

II) Οδηγίες σχετικές με τον κώδικα

- Το όνομα κάθε .py αρχείου να περιλαμβάνει το επώνυμό σας με λατινικούς χαρακτήρες, το χαρακτήρα της υπογράμμισης και τον αριθμό του συγκεκριμένου υποερωτήματος (π.χ. αν το επώνυμό σας είναι Γεωργίου, τότε ο κώδικας για την υποεργασία 1β θα έχει το όνομα Georgiou_1b.py). Κάθε αρχείο κώδικα που θα παραδοθεί θα πρέπει τουλάχιστον να περνάει τη φάση της διερμηνείας χωρίς συντακτικά σφάλματα.
- Τα αρχεία .py θα πρέπει να τα ανοίξετε και να τα επεξεργαστείτε με το ολοκληρωμένο περιβάλλον ανάπτυξης κώδικα IDLE. Ο κώδικας να είναι επαρκώς σχολιασμένος, σωστά στοιχισμένος και ενσωματωμένος μέσα στο έγγραφο Word, με τις απαντήσεις σας σε γραμματοσειρά courier.
- Στο έγγραφο της απάντησής σας και στο αρχείο του κώδικα θα πρέπει να δίνεται ολόκληρο το πρόγραμμα, επισημαίνοντας με σχόλια πού απαντάτε κάθε ερώτημα ώστε να θεωρούνται πλήρεις οι απαντήσεις.
- Όλα τα .py αρχεία με τον πηγαίο κώδικα και το .doc αρχείο κειμένου να υποβληθούν μέσω της πλατφόρμας **https://study.eap.gr**.

III) Τρόπος παράδοσης εργασίας:

α) Οι απαντήσεις πρέπει να είναι γραμμένες με χρήση **επεξεργαστή κειμένου** (π.χ. **Word**) σε σελίδες **διαστάσεων Α4 χωρίς χρώματα.** Το αρχείο να περιέχει ως **πρώτη σελίδα** το κείμενο του **Εντύπου Υποβολής – Αξιολόγησης** και ως δεύτερη σελίδα τον τίτλο «Σχόλια προς τον φοιτητή» (θα συμπληρωθεί από τον καθηγητή σας). Οι απαντήσεις στις υποεργασίες θα αρχίζουν από την τρίτη σελίδα, **χωρίς να επαναλαμβάνονται οι εκφωνήσεις**. Κάθε υποεργασία θα αρχίζει από νέα σελίδα. Για την απάντησή σας θα πρέπει να χρησιμοποιείτε υποχρεωτικά το **Πρότυπο Υποβολής Γραπτής Εργασίας**.

β) Το .doc αρχείο κειμένου να υποβληθεί στη διεύθυνση **https://study.eap.gr** με **όνομα αρχείου το επώνυμό σας με λατινικούς χαρακτήρες και τον Αριθμό Μητρώου σας**, π.χ. Ioannou_82345.

IV) Η καλή παρουσίαση της εργασίας λαμβάνεται υπόψη στην αξιολόγηση της εργασίας.

****