ΗΜΥ01Κ06 -Επιστημονικός Προγραμματισμός με την Python

Τρίτο Πρότζεκτ Φθινόπωρο 2025

Οδηγίες: Λύστε την άσκηση του πρώτου μέρους, και *μια από τις δυο* του δεύτερου μέρους. Αποθηκεύστε όλα τα προγράμματά σας σε ένα φάκελο χρησιμοποιώντας τα ονόματα των αρχείων που δίνονται στην αρχή κάθε άσκησης. Δώστε στο φάκελό σας το όνομα PROJECT_03_<ονοματεπώνυμο>_<ΑΜ> και συμπιέστε τον σε ένα αρχείο .zip ή .rar (πχ **PROJECT_03_ANDREOY_NIKOS_TH20999.zip**). Ανεβάστε το συμπιεσμένο αρχείο στο eclass Εργασίες > Project 03 ***μέχρι την Κυριακή 04/01/2026 23:55***

Προσοχή, Σημαντικό: Στον ίδιο φάκελο με τον κώδικά σας θα πρέπει επίσης να υπάρχουν και *όλα τα απαιτούμενα αρχεία εισόδου* ώστε καθένα από τα προγράμματά σας να μπορεί να τρέξει άμεσα (με ένα κλικ) όταν ο φάκελος αποσυμπιεστεί. Αν παραλείψετε να το κάνετε αυτό, το πρότζεκτ σας θα θεωρηθεί ελλιπές και θα βαθμολογηθεί ανάλογα.

Μέρος Α:

`txt_num_split.py` [3.5 μονάδες] + [0.5 μονάδα]

Υλοποιήστε ένα πρόγραμμα σε Python το οποίο θα ζητά από τον χρήστη ένα αρχείο κειμένου με κωδικοποίηση *cp1253* το οποίο περιέχει ελληνικό κείμενο και ακέραιους αριθμούς (χωρίς διαχωριστικό χιλιάδων) , και θα το χωρίζει σε δυο αρχεία με ονόματα `keimeno.txt` και `numbers.txt` όπου το μεν πρώτο να περιέχει *το αρχικό κείμενο χωρίς τους αριθμούς*, και το δεύτερο *μόνο τους αριθμούς έναν σε κάθε γραμμή και* ***ταξινομημένους*** *από τον μικρότερο ως τον μεγαλύτερο*.

Για δοκιμή μπορείτε να χρησιμοποιήσετε το αρχείο κειμένου `"test_cp1253.txt"` που βρίσκεται στον φάκελο του e-class *Έγγραφα > Αρχεία Τρίτου Project*

Για 0.5 επιπλέον μονάδα υλοποιήστε μια βελτιωμένη έκδοση (ονομάστε το πρόγραμμά σας **`txt_num_split_extra.py`**) που να μπορεί να χειρίζεται ελληνικά αρχεία κειμένου ανεξάρτητα από το είδος της κωδικοποίησής τους (δηλαδή *είτε* windows1253 *είτε* utf-8) την οποία θα αναγνωρίζει αυτόματα και θα προχωρά αναλόγως.

Για δοκιμή μπορείτε να χρησιμοποιήσετε τα αρχεία κειμένου `"test_cp1253.txt"` και `"test_utf8.txt"` που βρίσκονται στον φάκελο *Έγγραφα > Αρχεία Τρίτου Project* και περιέχουν ίδιο κείμενο με διαφορετικές κωδικοποιήσεις. Το πρόγραμμά σας πρέπει να δέχεται και να επεξεργάζεται κανονικά και τα δυο αυτά αρχεία κειμένου.

Υπόδειξη: Κάντε το πρόγραμμά σας να δοκιμάζει αρχικά να διαβάσει το κείμενο υποθέτοντας κωδικοποίηση windows1253 και σε περίπτωση σφάλματος ανάγνωσης (που θα χειρίζεται με try-except) να ξαναδοκιμάζει με κωδικοποίηση utf-8

Μέρος Β: (Λύστε *τη μία από τις δυο* ασκήσεις)

B1. `word_freq_plus_apax.py` [5.5 μονάδες]

Στον φάκελο του e-class *Έγγραφα > Αρχεία Τρίτου Project* βρίσκεται το αρχείο *"articles.zip"* Κάνοντας unzip αυτό το αρχείο στον υπολογιστή σας δημιουργείται ενας φάκελος με όνομα "articles" μέσα στον οποίο υπάρχουν 578 ειδησεογραφικά άρθρα σε αρχεία *αγγλικού* κειμένου με αντίστοιχα ονόματα *"article000.txt"* , *"article001.txt"* κ.ο.κ. μέχρι και *"article577.txt"* .

1. Κατασκευάστε την συνάρτηση `get_txtFile_contents(file_path)` η οποία δέχεται σαν όρισμα το μονοπάτι `file_path` ενός αρχείου αγγλικού κειμένου (πχ. `articles/article001.txt`) και επιστρέφει σε μορφή λίστας όλες τις λέξεις *τουλάχιστον 2 γραμμάτων* που περιέχονται σε αυτό το αρχείο (επομένως εδώ ορίζουμε σαν λέξη κάθε συνδυασμό *δυο ή περισσότερων* αγγλικών αλφαβητικών χαρακτήρων – κεφαλαίων ή/και πεζών. Όχι αριθμητικά ψηφία, όχι σημεία στίξης). [Υπόδειξη: χρησιμοποιείστε την μέθοδο `re.findall()` με την κατάλληλη κανονική έκφραση]. Σημειώστε ότι *η ίδια λέξη μπορεί να εμφανίζεται πολλές φορές* στη λίστα που επιστρέφεται.

2. Στη συνέχεια κατασκευάστε ένα πρόγραμμα με όνομα `word_freq_plus_apax.py` το οποίο με τη βοήθεια της παραπάνω συνάρτησης να διαβάζει ένα-ένα το περιεχόμενο καθενός από παραπάνω τα 578 αρχεία κειμένου και να προσθέτει τις λέξεις του σε ένα κεντρικό λεξικό που θα ονομάζεται `word_freq_dict`. Το λεξικό αυτό θα έχει *κλειδιά* τις *λέξεις από όλα τα άρθρα* και αντίστοιχες *τιμές* τον *αριθμό των εμφανίσεων (συχνότητα) κάθε λέξης*. [Υπόδειξη: Αν μια λέξη δεν υπάρχει στο λεξικό τότε προστίθεται ως νέο κλειδί, και παίρνει τιμή τον αριθμό 1 (πρώτη εμφάνιση). Αν η λέξη αυτή υπάρχει ήδη στο λεξικό, τότε η τιμή της αυξάνεται κατά 1 (μια ακόμα εμφάνιση) - κάτι παρόμοιο έχετε ήδη κάνει στην πρώτη από τις δυο ασκήσεις στην Εργασία 5η]

Αφού δημιουργήσει το λεξικό, το πρόγραμμά σας:

α) θα εμφανίζει στην οθόνη μια λίστα με τις 10 πιο συχνά εμφανιζόμενες λέξεις σε όλο το σώμα των άρθρων, μαζί με την συχνότητα εμφάνισης καθεμιάς από αυτές

β) θα δημιουργεί ένα νέο αρχείο με όνομα "`apax_legomena.txt`" στο οποίο θα περιέχονται (σε μια γραμμή η καθεμία) όλες οι λέξεις που εμφανίζονται μόνο μια φορά η καθεμιά στο συνολικό κείμενο όλων των άρθρων – δηλαδή τα "άπαξ λεγόμενα".

> Το "άπαξ λεγόμενον" είναι όρος της γλωσσολογίας για λέξεις που συναντώνται μονάχα μια φορά, είτε μέσα σε ένα κείμενο, είτε σε ένα σύνολο έργων, είτε σε μια ολόκληρη γλώσσα

Δείτε στην επόμενη σελίδα μια ενδεικτική μορφή εκτύπωσης του προγράμματός σας.

```
Οι 10 πιο συχνές λέξεις που βρέθηκαν είναι οι ακόλουθες:
 1)  5673: 'the'
 2)  3364: 'to'
 3)  2961: 'of'
  . . .
Το αρχείο 'apax_legomena.txt' που δημιουργήθηκε περιέχει 3871 μοναδικές
λέξεις.

------------------------------
```

Για δική σας επαλήθευση, βεβαιωθείτε ότι το αρχείο `'apax_legomena.txt'` που δημιουργήσατε ξεκινάει ως εξής:

```
REVISE
LONG
DOWNWARDS
revise
kilolitres
emergence
deliberations
Nuclear
. . .
```

Σημαντικό: Τόσο το πρόγραμμά σας όσο και τα αρχεία που χρειάζονται για να τρέξει πρέπει να παραδοθούν σε ενιαίο φάκελο κατά τρόπο ώστε να μπορεί *να τρέξει με ένα κλικ,* χωρίς να απαιτούνται άλλες διαδικασίες από την πλευρά του χρήστη. Σε διαφορετική περίπτωση η άσκησή σας θα θεωρηθεί ελλιπής και θα βαθμολογηθεί ανάλογα.

B2. `decrypt_mystery_text.py` [5.5 μονάδες] + [0.5 μονάδα]

Στον φάκελο του e-class *Έγγραφα > Αρχεία Τρίτου Project* βρίσκεται το δυαδικό αρχείο *"secret_binary"* [(*)] το οποίο περιέχει κωδικοποιημένο ένα άγνωστο σε σας κείμενο σύμφωνα με τον αλγόριθμο κρυπτογράφησης που υλοποιήσατε στο δεύτερο project (ενισχυμένη παραλλαγή του κρυπτογράμματος Vigenère).

Έχετε τις εξής τρεις πληροφορίες:

1. Το αρχείο *"secret_binary"* περιέχει μια σειριακή ακολουθία από bytes που έχει προκύψει από την σειριοποίηση (pickling) μιας *πλειάδας ακεραίων* που κωδικοποιούν το άγνωστο σε σας κείμενο. *[για την διαδικασία pickling / unpickling ανατρέξτε στο τέλος της Διάλεξης 8: Αρχεία]*

2. Το άγνωστο κείμενο περιέχει τη λέξη `"βιώνουμε"`

3. Το κλειδί της κωδικοποίησης είναι μια λέξη από το ποίημα του Γιώργου Σεφέρη "Άρνηση"

Αξιοποιώντας τις παραπάνω πληροφορίες, κατασκευάστε ένα πρόγραμμα σε Python (ονομάστε το **`decrypt_mystery_text.py`**) που να αποκωδικοποιεί και να εκτυπώνει το αρχικό μυστικό κείμενο μαζί με το κλειδί που χρησιμοποιήθηκε για την κωδικοποίησή του. Μπορείτε προφανώς να χρησιμοποιήσετε τμήματα του κώδικά σας από την αντίστοιχη άσκηση του δεύτερου πρότζεκτ.

Για 0.5 επιπλέον μονάδα υλοποιήστε μια βελτιωμένη έκδοση (ονομάστε το αρχείο σας **`super_decrypt_mystery_text.py`**) που να αποκωδικοποιεί και να εκτυπώνει το αρχικό μυστικό κείμενο *χωρίς να λαμβάνει υπόψη του την πληροφορία* ότι σε αυτό περιέχεται η λέξη "*βιώνουμε*".

Υπόδειξη: Πως μπορούμε άραγε να αποφανθούμε (με σχετική σιγουριά) ότι το περιεχόμενο ενός αρχείου κειμένου είναι πραγματικά κείμενο και όχι "ασυναρτησίες" χωρίς να δούμε αυτό το αρχείο στην οθόνη (οπότε φυσικά θα το καταλαβαίναμε αμέσως); Κάτι που διαφοροποιεί ένα γλωσσικό κείμενο από μια τυχαία ακολουθία χαρακτήρων είναι η δομή του σε επίπεδο "λέξεων", δηλαδή ακολουθιών χαρακτήρων που χωρίζονται με κενά. Τι συμπεραίνετε πχ. αν ένα αρχείο κειμένου περιέχει "λέξεις" μήκους 20, 30 ή 40 χαρακτήρων; Περιέχει κάποιο γλωσσικό κείμενο ή μήπως ασυναρτησίες;

(*) Το δυαδικό αρχείο *"secret_binary"* δεν έχει όπως θα αναμενόταν την κατάληξη '.bin' επειδή το eclass αρνείται να διαχειριστεί αρχεία με αυτή την κατάληξη. Όμως μπορείτε να το κατεβάσετε χωρίς πρόβλημα, και στη συνέχεια να το ανοίξετε κανονικά στην Python με την εντολή `open("secret_binary", "rb")`