ΗΜΥ01Κ06 -Επιστημονικός Προγραμματισμός με την Python

Project 2 από 3

Φθινόπωρο 2025

Οδηγίες: Λύστε ***τη μία από τις ασκήσεις Α1 ή Α2*** και την ***άσκηση Β***. Αποθηκεύστε τον κώδικά σας σε ένα φάκελο χρησιμοποιώντας τα ονόματα των αρχείων που δίνονται στην αρχή κάθε άσκησης. Ονομάστε τον φάκελό σας PYTHON_PROJECT_02 ακολουθούμενο από το ονοματεπώνυμο και το ΑΜ σας Πχ `PYTHON_PROJECT_02_ANDREOY_NIKOLAOS_TH26999` και συμπιέστε τον σε ένα αρχείο .zip ή .rar με το ίδιο όνομα. Ανεβάστε το συμπιεσμένο αρχείο στο eclass Εργασίες > Project 02

Άσκηση Α1: *[4.0 μονάδες]* **`factorization.py`** *(με αυτό το όνομα θα την αποθηκεύσετε)*

Δημιουργήστε δυο συναρτήσεις με ονόματα `Elaxisto_Koino_Pollaplasio(a,b)` και `Megistos_Koinos_Dieretis(a,b)` που θα δέχονται σαν όρισμα δυο ακέραιους θετικούς αριθμούς και θα επιστρέφουν αντίστοιχα το Ελάχιστο Κοινό Πολλαπλάσιο (ΕΚΠ) και τον Μέγιστο Κοινό Διαιρέτη (ΜΚΔ) τους. Ένας τρόπος υπολογισμού των παραπάνω ποσοτήτων είναι να μετατρέψετε τους δυο αριθμούς σε γινόμενα πρώτων παραγόντων. Για να γίνει αυτό, δημιουργήστε μια συνάρτηση `paragontes(n)` που θα δέχεται σαν όρισμα έναν ακέραιο θετικό αριθμό και θα επιστρέφει το γινόμενο πρώτων παραγόντων του, υπό μορφή ενός λεξικού. Για παράδειγμα ο αριθμός `504` αναλύεται σε πρώτους παράγοντες ως εξής:

$504 = 2x2x2x3x3x7 = 2^3x3^2x7^1$ που κωδικοποιείται ως λεξικό `{2:3, 3:2, 7:1}`

Έχοντας μετατρέψει τους αριθμούς σε γινόμενο πρώτων παραγόντων

- το ΕΚΠ είναι ίσο με το γινόμενο των *κοινών* και μη *κοινών* παραγόντων με τους *μεγαλύτερους* εκθέτες
- ο ΜΚΔ είναι ίσος με το γινόμενο των *κοινών* παραγόντων με τους *μικρότερους* εκθέτες

Τέλος δημιουργήστε ένα πρόγραμμα που θα περιέχει τις παραπάνω συναρτήσεις και θα υπολογίζει το ΕΚΠ και τον ΜΚΔ δυο αριθμών που δίδονται από τον χρήστη εκτυπώνοντας ενδιάμεσα και τα δυο λεξικά που προκύπτουν από την ανάλυσή τους σε πρώτους παράγοντες

Παράδειγμα:
Αν $a = 504$ δηλαδή $2^3 \times 3^2 \times 7^1$
και $b = 140$ δηλαδή $2^2 \times 5^1 \times 7^1$
τότε το Ελάχιστο Κοινό Πολλαπλάσιο τους θα είναι $2^3 \times 3^2 \times 5^1 \times 7^1 = 2520$
Και ο Μέγιστος Κοινός Διαιρέτης τους θα είναι $2^2 \times 7^1 = 28$

Παράδειγμα εκτέλεσης του προγράμματος

```
Δώσε δυο θετικούς ακέραιους 504 140
504 = {2:3, 3:2, 7:1}
140 = {2:2, 5:1, 7:1}
ΕΚΠ(504,140) = 2520
ΜΚΔ(504,140) = 28
```

Άσκηση Α2: *[4.0 μονάδες]* **`max_chain.py`** *(με αυτό το όνομα θα την αποθηκεύσετε)*

Ορισμός: Λέμε ότι μια συμβολοσειρά περιέχει μια αλυσίδα n όμοιων χαρακτήρων (όπου n>1), όταν σε κάποιο σημείο της ένας χαρακτήρας επαναλαμβάνεται συνεχόμενα n *ακριβώς* φορές.

Για παράδειγμα η συμβολοσειρά `"adhfgdd5%6ccccc ggg2!334"` περιέχει μια αλυσίδα μήκους 5 (`"ccccc"`), μια μικρότερη αλυσίδα μήκους 3 (`"ggg"`) και δυο ακόμα μικρότερες αλυσίδες μήκους 2 (`"dd"` και `"33"`)

Δημιουργήστε ένα πρόγραμμα σε Python που να ζητάει από τον χρήστη μια συμβολοσειρά και να επιστρέφει *το **μήκος** και την **θέση** της αλυσίδας **μέγιστου μήκους*** που τυχόν αυτή περιέχει (αν η συμβολοσειρά περιέχει περισσότερες από μια αλυσίδες μέγιστου μήκους, τότε το πρόγραμμά σας θα επιστρέφει την πρώτη που συνάντησε).

Παραδείγματα εκτέλεσης του προγράμματος

```
Δώσε μια συμβολοσειρά > adhfgdd5%6ccccc ggg2!334
Περιέχει αλυσίδα μεγίστου μήκους 5 (ccccc) που ξεκινάει από την θέση 10

Δώσε μια συμβολοσειρά > 5249021111115500000074
Περιέχει αλυσίδα μεγίστου μήκους 6 (111111) που ξεκινάει από την θέση 6
(εδώ υπάρχει και δεύτερη αλυσίδα μεγίστου μήκους, όμως επιστρέφεται η πρώτη)

Δώσε μια συμβολοσειρά > zzzzzzzzz
Περιέχει αλυσίδα μεγίστου μήκους 9 (zzzzzzzzz) που ξεκινάει από την θέση 0

Δώσε μια συμβολοσειρά > wofi313xald
Δεν περιέχει αλυσίδες όμοιων χαρακτήρων
```

Πιθανή υπόδειξη: Δημιουργήστε πρώτα μια συνάρτηση get_chain(s, chain_len) η οποία να επιστρέφει την θέση *(τον δείκτη του πρώτου στοιχείου)* της *πρώτης* αλυσίδας μήκους chain_len στην συμβολοσειρά s, ή None αν η συμβολοσειρά δεν περιέχει αλυσίδες του *δοσμένου* μήκους.

Πχ για τη συμβολοσειρά `"adhfgdd5%6ccccc ggg2!334"` και για chain_len ίσο με 2, η get_chain θα πρέπει να επιστρέφει τον ακέραιο 5 (θέση απ' όπου ξεκινάει η αλυσίδα `"dd"`), ενώ για chain_len ίσο με 4, θα πρέπει να επιστρέφει None.

Άσκηση Β: *[6.0 μονάδες]* **`kryptogramma.py`** *(με αυτό το όνομα θα την αποθηκεύσετε)*

Δημιουργήστε ένα πρόγραμμα που να υλοποιεί μια ενισχυμένη παραλλαγή ενός από τους πιο απλούς αλγόριθμους κρυπτογράφησης (κρυπτόγραμμα Vigenère) η οποία περιγράφεται ως εξής:

Δίνεται ένα *μήνυμα (συνήθως μια πρόταση)* προς κρυπτογράφηση, και μια μικρότερη λέξη ή φράση που ονομάζεται *κλειδί* της κρυπτογράφησης:

Παράδειγμα: Μήνυμα: `"Θα επιτεθούμε τη νύχτα"` Κλειδί κρυπτογράφησης: `"πέτρα"`

Για την ***κρυπτογράφηση*** του μηνύματος ακολουθούμε την εξής διαδικασία:

Βήμα Κ1: Μετατρέπουμε τη συμβολοσειρά που περιέχει το μήνυμα, σε *λίστα χαρακτήρων*.

```
['Θ', 'α', ' ', 'ε', 'π', 'ι', 'τ', 'ε', 'θ', 'ο', 'ύ', 'μ', 'ε', ' ', 'τ',
'η', ' ', 'ν', 'ύ', 'χ', 'τ', 'α']
```

Στη συνέχεια μετατρέπουμε την *λίστα χαρακτήρων* σε σε *λίστα ακεραίων*, αντικαθιστώντας κάθε χαρακτήρα με τον δείκτη της θέσης του στον πίνακα κωδικοποίησης UNICODE. Αυτό γίνεται χρησιμοποιώντας την ενσωματωμένη συνάρτηση `ord(c)` Έτσι πχ το `ord('Θ')` θα επιστρέψει `920`, το `ord('α')` θα επιστρέψει `945`, το `ord(' ')` θα επιστρέψει `32` κλπ. και τελικά θα πάρουμε την εξής λίστα ακεραίων:

```
[920, 945, 32, 949, 960, 953, 964, 949, 952, 959, 973, 956, 949, 32, 964,
951, 32, 957, 973, 967, 964, 945]
```

Ονομάζουμε `L_msg` την λίστα ακεραίων που δημιουργήσαμε. Προφανώς η `L_msg` θα έχει το ίδιο μήκος με την συμβολοσειρά του μηνύματος από το οποίο προήλθε (στην περίπτωσή μας 22)

Βήμα Κ2: Χρησιμοποιώντας τη συνάρτηση `string_hash(key)` *η οποία δίνεται* μετατρέπουμε το κλειδί της κρυπτογράφησης σε μια νέα συμβολοσειρά που αποτελείται μόνο από δεκαδικά ψηφία και έχει το ίδιο μήκος με την αρχική. Έτσι πχ αν το κλειδί μας είναι `"πέτρα"`, τότε παραπάνω συνάρτηση θα επιστρέψει τη συμβολοσειρά `"70631"`, που αποτελείται μόνο από δεκαδικά ψηφία και έχει μήκος 5, όσο και το αρχικό μας κλειδί.

Βήμα Κ3: Χρησιμοποιώντας το αποτέλεσμα από το *Βήμα 2*, δημιουργούμε μια *λίστα μονοψήφιων ακεραίων* που έχει ίδιο μήκος με το αρχικό μήνυμα και αποτελείται από τα ψηφία του αριθμητικού κλειδιού `"70631"`, που επαναλαμβάνονται συνεχώς μέχρι να γεμίσει αυτή η λίστα. Αν το μήκος της λίστας δεν είναι ακέραιο πολλαπλάσιο του μήκους του κλειδιού, τότε στο τέλος συμπληρώνουμε μόνο όσα ψηφία είναι απαραίτητα.

Έτσι πχ καθώς το παραπάνω μήνυμα έχει μήκος 22, και το κλειδί είναι 5-ψήφιο η λίστα μας θα είναι:

```
[7, 0, 6, 3, 1, 7, 0, 6, 3, 1, 7, 0, 6, 3, 1, 7, 0, 6, 3, 1, 7, 0]
```

Ονομάζουμε `L_key` την λίστα ακεραίων που δημιουργήσαμε.

Βήμα Κ4: Συνδυάζοντας τις λίστες ακεραίων `L_msg` και `L_key` *(οι οποίες προφανώς έχουν το ίδιο μήκος, στο παράδειγμά μας 22),* δημιουργούμε μια νέα λίστα `L_encrypted` σύμφωνα με τον ακόλουθο τύπο:

```
L_encrypted[i] = 2 * L_msg[i] + 5 *  L_key[i]
```
(τύπος 1)

Σύμφωνα με την παραπάνω λογική, το κρυπτογραφημένο μήνυμα παράγεται ως εξής:

Αρχικό μήνυμα	θ	α		ε	π	ι	τ	ε	θ	ο	ύ	μ	ε	…
L_msg	920	945	32	949	960	953	964	949	952	959	973	956	949	…
L_key	7	0	6	3	1	7	0	6	3	1	7	0	6	…
L_encrypted	1875	1890	94	1913	1925	1941	1928	1928	1919	1923	1981	1912	1928	…

Εδώ για παράδειγμα, το `L_encrypted[0]` που είναι `1875` προέκυψε από την πράξη `2*920 + 5*7`

Η λίστα `L_encrypted` περιέχει το κρυπτογραφημένο μήνυμα που μπορεί τώρα να μεταδοθεί σε κοινή θέα, χωρίς κανείς να κατανοεί το περιεχόμενό της, ακόμα και αν είναι γνωστός ο αλγόριθμος κρυπτογράφησής της. Ο μόνος τρόπος να αποκρυπτογραφηθεί είναι αν μας δοθεί η λέξη-κλειδί.

Για την ***αποκρυπτογράφηση*** του μηνύματος χρειαζόμαστε την `L_encrypted` και φυσικά τη λέξη-κλειδί που χρησιμοποιήθηκε για κρυπτογράφηση και εργαζόμαστε ως εξής:

Βήμα Α1: Χρησιμοποιούμε τη λέξη-κλειδί (εδώ το "πέτρα") για να δημιουργήσουμε την λίστα ακεραίων ψηφίων `L_key` όπως ακριβώς και στα βήματα Κ2 και Κ3 της κρυπτογράφησης. Το μήκος της `L_key` προκύπτει από το μήκος της `L_encrypted` το οποίο γνωρίζουμε.

Βήμα Α2: Συνδυάζουμε τις λίστες `L_encrypted` και `L_key` για να δημιουργήσουμε την λίστα `L_decrypted` αντιστρέφοντας τον τύπο που χρησιμοποιήσαμε στην κρυπτογράφηση

```
L_decrypted[i] = (L_encrypted[i] - 5 * L_key[i]) / 2
```
(τύπος 2)

Η λίστα `L_decrypted` περιέχει τώρα τους ακεραίους που αντιστοιχούν στους χαρακτήρες UNICODE του αρχικού μηνύματος (ταυτίζεται με την `L_msg` της αρχικής κωδικοποίησης)

Βήμα Α3: Χρησιμοποιώντας την εσωτερική συνάρτηση `chr(i)` - αντίστροφη της `ord(c)` - μετατρέπουμε τους ακεραίους UNICODE της `L_decrypted` σε χαρακτήρες, αποκαλύπτοντας έτσι το αρχικό μήνυμα.

Σύμφωνα με την παραπάνω λογική, το αρχικό μήνυμα αποκαλύπτεται ως εξής:

L_encrypted	1875	1890	94	1913	1925	1941	1928	1928	1919	1923	1981	1912	1928	…
L_key	7	0	6	3	1	7	0	6	3	1	7	0	6	…
L_decrypted	920	945	32	949	960	953	964	949	952	959	973	956	949	…
Αρχικό μήνυμα	θ	α		ε	π	ι	τ	ε	θ	ο	ύ	μ	ε	…

Εδώ για παράδειγμα, το `L_decrypted[0]` που είναι `920` προέκυψε από την πράξη `(1875 – 5*7)/2`

Επομένως, για να υλοποιήσετε αυτήν την άσκηση:

α) Δημιουργήστε τη συνάρτηση `encrypt(txt_message, key)` η οποία θα δέχεται σαν ορίσματα εισόδου τις *συμβολοσειρές* του μηνύματος και του κλειδιού αντίστοιχα, και θα επιστρέφει μια *λίστα* ακεραίων που θα αντιστοιχούν στο κωδικοποιημένο μήνυμα.

β) Δημιουργήστε τη συνάρτηση `decrypt(L_encrypted, key)` η οποία θα δέχεται σαν ορίσματα εισόδου την *λίστα* που αντιστοιχεί στο κωδικοποιημένο μήνυμα και την *συμβολοσειρά* του κλειδιού αντίστοιχα, και θα επιστρέφει μια συμβολοσειρά που θα αντιστοιχεί στο αποκωδικοποιημένο μήνυμα.

γ) Τέλος δημιουργήστε το πρόγραμμα **kryptogramma.py** που θα περιέχει τις παραπάνω συναρτήσεις και θα ζητάει από τον χρήστη ένα μήνυμα με το αντίστοιχο κλειδί του, θα το κωδικοποιεί χρησιμοποιώντας την συνάρτηση `encrypt` και θα εκτυπώνει την κωδικοποίησή του σαν μια λίστα ακεραίων. Στη συνέχεια θα χρησιμοποιεί την συνάρτηση `decrypt` για να αποκωδικοποιήσει και να εκτυπώσει το αρχικό μήνυμα.

> Η συνάρτηση `string_hash(key)` που χρειάζεστε στα βήματα Κ2 και Α1 βρίσκεται στο eclass > Έγγραφα > Αρχεία Δεύτερου Πρότζεκτ

Παράδειγμα εκτέλεσης του προγράμματος

```
Δώσε το μήνυμα προς κρυπτογράφηση: Θα επιτεθούμε τη νύχτα
Δώσε το κλειδί της κρυπτογράφησης: πέτρα

Κρυπτογραφημένο μήνυμα (λίστα ακεραίων):
[1875, 1890, 94, 1913, 1925, 1941, 1928, 1928, 1919, 1923, 1981, 1912, 1928, 79,
1933, 1937, 64, 1944, 1961, 1939, 1963, 1890]

Αποκρυπτογραφημένο μήνυμα (κλειδί = πέτρα):
Θα επιτεθούμε τη νύχτα
```