



Νέες έννοιες στο φύλλο:
ακολουθίες, αλφαριθμη-
τικά, τεμαχισμός, λίστες,
σύνολα, for
Ακόμη περισσότερα για
if, while, συναρτήσεις



Όταν γράφουμε στο κέ-
λυφος απευθείας ονό-
ματα μεταβλητών, στα-
θερές, κλήσεις συναρ-
τήσεων, εκφράσεις, το
κέλυφος αποκρίνεται με
την εμφάνιση της τιμής
τους. Δεν πρόκειται για
εντολές.

Στα προγράμματά μας
ελέγχουμε τι εμφανίζε-
ται στην έξοδο, και πώς,
αποκλειστικά με τη συ-
νάρτηση print().



Βλέπεις ότι θάζεις **δύο**
δείκτες [από:έως]. Ο
πρώτος μας δείχνει **από**
ποια θέση θα ξεκινήσεις
και ο δεύτερος μας δεί-
χνει **έως** ποια θα φτά-
σεις (**χωρίς αυτήν**).

Η θετική αρίθμηση ξεκι-
νάει από το 0 για τον
πρώτο χαρακτήρα, η αρ-
νητική από το -1 για τον
τελευταίο χαρακτήρα.

Η φορά, ωστόσο, για το
[από, έως], είναι πά-
ντοτε προς τα δεξιά).

Αντί εισαγωγής

Σήμερα θα σε κρεμάσουμε! Στο παιχνίδι μας ένας παίκτης θα δίνει στον υπολογι-
στή τη **μυστική λέξη**, και μετά ο άλλος παίκτης θα **προσπαθεί να τη μαντέψει**,
δίνοντας γράμματα. Η γνωστή σου κρεμάλα.

Για να φτιάξεις το παιχνίδι, θα χρειαστεί να μάθεις μερικά **καινούρια** κόλπα της
Python. Πάμε πρώτα στο κέλυφος.

>>> 1. Γράψε, και μετά, δώσε το όνομά σου:

```
fname = input("Δώσε μου το όνομά σου: ")
```

Δώσε μου το όνομά σου: π.χ. Αλέξανδρος

Τί περιέχει τώρα η μεταβλητή fname ;

2. Τώρα γράψε στο κέλυφος την εντολή:

```
fname[1]
```

Ποιο γράμμα του ονόματός σου εμφανίστηκε; (το 1ο; το 2ο; το 3ο; το τελευταίο;)

3. Κάνε δοκιμές και συμπλήρωσε κατάλληλα τον παρακάτω πίνακα:

Για να πάρω το...	Γράφω στην python...
Πρώτο γράμμα	
Δεύτερο γράμμα	fname[1]
Τρίτο γράμμα	
Τέταρτο γράμμα	
Το τελευταίο γράμμα	

Τι συμπέρασμα βγάζεις για τον **δείκτη**; (τον αριθμό που βάζουμε στις αγκύλες)

4. Για δοκίμασε τώρα να βάλεις... αρνητικό αριθμό! Για παράδειγμα, ποιον χαρα- κτήρα θα δεις αν γράψεις:

```
fname[-1]
```

Μπορείς με αρνητικό δείκτη να εμφανίσεις τον πρώτο χαρακτήρα;

Τον προτελευταίο;

Τον τρίτο από το τέλος;

5. Γράψε τώρα:

```
fname[2:4]
```

Ποιο τμήμα εμφανίστηκε; από ποια θέση ξεκίνησε, και σε ποια τελείωσε;

Ας κάνουμε τώρα μία ακόμα μικρή άσκηση! Πρώτα δώσε την εντολή:

```
s = "Η καλή μέρα από το πρωί φαίνεται!"
```

Κάνε τώρα δοκιμές, και συμπλήρωσε τον πίνακα παρακάτω:

Για να πάρω το...	Γράφω στην python...	Υπόδειξη - σχόλιο
"Η" (δηλ τον 1ο χαρακτήρα)		αυτό είναι πολύ εύκολο!
"!" (δηλαδή τον τελευταίο χαρακτήρα)		και αυτό είναι πολύ εύκολο! δεν χρειάζεσαι μέτρημα!
"καλή"		χμ, εδώ θες δύο δείκτες...
"φαίνεται!"		και εδώ θες δύο δείκτες. Όμως, είπαμε ότι το «έως» δεν περιλαμβάνει τη θέση που α-ριθμεί, έτσι, δεν μπορείς να χρησιμοποιήσεις αρνητικές τι-μές)
"το πρωί φαίνεται"		μοιάζει με το προηγούμενο. Για δοκίμασε αρνητικούς δείκτες!
"το"		ευκολάκι... με δυο τρόπους!



[από: έως: βήμα]

Ο τρίτος δείκτης αποτε-
λεί το βήμα. Θα ξεκινάει
από το **από**, θα συνεχίζει
βήμα-βήμα θέσεις, και
θα τελειώσει όσο πιο κο-
ντά πριν το **έως**.

7. Χμμ... Πειραματίσου λίγο ακόμα. Γράψε τώρα:

```
s [0 : 10 : 2]
```

Ποιους χαρακτήρες περιέχει το τμήμα που εμφανίστηκε; Πώς λειτουργεί ο τρίτος δείκτης που πρόσθεσες;

8. Άλλη μια μικρή άσκηση τώρα! Πρώτα δώσε την εντολή:

```
ab = "αβγδεζηθικλμνξοπρστυφχψω"
```

Κάνε τις απαραίτητες δοκιμές για να συμπληρώσεις τον παρακάτω πίνακα:

Για να πάρω το...	Γράφω ...	Υπόδειξη - σχόλιο
"αβγδεζηθικ" (Τα γράμματα από το α μέχρι το κ)		δυο δείκτες φτάνουν
"αγεη" (Τα γράμματα από το α μέ- χρι το κ, θέση παρά θέση.		εδώ θες τρεις
"αδηκ" (από το α ως κ, παραλείπο- ντας δύο ενδιάμεσα, κάθε φορά)		κι εδώ τρεις
"κιθηξεδγβ"		εδώ, δύο ή τρεις; Άραγε παίζουν αρνητικά βήματα;

Στο [από:έως:βήμα] και
στο [από:έως], μπο-
ρούμε να παραλείψουμε
το **από** ή/και το **έως** ε-
φόσον αυτά είναι αντί-
στοιχα η αρχή και το τέ-
λος του αλφαριθμητι-
κού. Δεν παραλείπουμε
όμως την άνω και κάτω
τελεία.

9. Δοκίμασε τώρα αυτά:

Γράφω στην python...	Για να πάρω το...
ab [: 12]	
ab [12 :]	
ab [: 12 : 2]	
ab [: 12 : -2]	
Τι θα γράψω για να πάρω <u>αντίστροφα όλο</u> το αλφάβητο;	

10. Κουράστηκε κανείς από τώρα; Συνεχίζεις να πειραματίζεσαι. Γράψε την εντολή, και μετά δώσε το επώνυμο:

```
lname = input("Δώσε μου το επώνυμό σου: ")
```

Δώσε μου το επώνυμό σου: π.χ. Παπαδόπουλος

11. Μπορείς να εμφανίσεις τώρα και το όνομα και το επώνυμό σου; Πώς;

12. Δοκίμασε να γράψεις:

```
fname+lname
```

Τι παρατηρείς; Τι πρέπει να διορθώσεις;

13. Δώσε την εντολή:

```
fullname = fname + " " + lname
```

Τι τιμή έχει τώρα η μεταβλητή fullname; Εμφάνισέ την στο κέλυφος.

14. Επομένως, με το + μπορείς να ενώνεις αλφαριθμητικά. Δες τώρα τι μπορείς να κάνεις σε ένα αλφαριθμητικό, αν το πολλαπλασιάσεις με ένα αριθμό. Δοκίμασε:

```
multi_fname = fname*3
```

Τι παρατηρείς; Τι θα γινόταν αν αντί για 3 βάζαμε 4 ή 5 ; Δοκίμασέ το!

15. Πολλές φορές χρειαζόμαστε κι άλλες πληροφορίες για τα αλφαριθμητικά. Δοκίμασε τα ακόλουθα και σημειώσε τα αποτελέσματά τους:

Καλώ τη συνάρτηση...	Αποτέλεσμα
len(fname)	
len(lname)	
len(fullname)	
len("123456")	
len("1")	

Ποια είναι η λειτουργία της len();

16. Ουφ! Κουραστικά όλα αυτά, ε; Ευτυχώς ο υπολογιστής μπορεί να επαναλαμβάνει λειτουργίες για μας. Δοκίμασε τη σύνθετη εντολή στις 2 ακόλουθες γραμμές:

```
for i in (0, 1, 2, 3, 4):
    print(i)
```

Τι πήρες στην οθόνη; Τι πιστεύεις ότι κάνει η εντολή **for**;

17. Πειραμάτισου κι άλλο. Δώσε:

```
for i in (0, 1, 2, 3, 4):
    print(fullname[i])
```



Λίγη ορολογία, τώρα.

Τα **αλφαριθμητικά** (strings) ανήκουν στην κατηγορία δεδομένων που ονομάζουμε στην ρηθον ακολουθίες (sequences).

Πιο συγκεκριμένα τα αλφαριθμητικά είναι ακολουθίες από χαρακτήρες.

Όλες οι ακολουθίες στην ρηθον έχουν δύο χαρακτηριστικά:

- (1) αποτελούνται από επιμέρους στοιχεία-μέλη (όπως οι χαρακτήρες), και
- (2) έχει σημασία η θέση των μελών (π.χ. άλλο γραφή κι άλλο φραγή).

Η λειτουργία που επιτελούμε με τις αγκύλες [], ονομάζεται **τεμαχισμός** (slicing), αυτό ακριβώς που δηλώνει ο όρος: παίρνουμε μικρότερα ή μεγαλύτερα τεμάχια (φέτες) μιας ακολουθίας.

Θα δούμε και άλλες ακολουθίες στην ρηθον:

λίστες (lists), **πλειάδες** (tuples) και **διαστήματα** (ranges).



Τι πήρες τώρα στην οθόνη; Μπορείς να εξηγήσεις γιατί;

18. Κάνε κάτι παραπλήσιο, αλλά λιγάκι διαφορετικό:

```
for i in range(0, 5):
    print(fullname[i])
```

Τι πήρες τώρα στην οθόνη; Τι πιστεύεις ότι κάνει η συνάρτηση **range()**;



Χρησιμοποίησες με επιτυχία τη **συνάρτηση range()** για να ορίσεις αντίστοιχα **διαστήματα**, που χρειάστηκαν στην νέα επαναληπτική εντολή που μαθαίνεις, την **for**. Ως τώρα ήξερες μόνο την **while**.

19. Κάνε μια πολύ μικρή αλλαγή:

```
for i in range(5):
    print(fullname[i])
```

Τι πήρες τώρα στην οθόνη; Τι άλλαξε στη συμπεριφορά της συνάρτησης **range()**;

20. Κάνε άλλη μια μικρή αλλαγή:

```
for i in range(3, 10, 2):
    print(fullname[i])
```

Τι πήρες τώρα; Σε τι αντιστοιχίστηκαν τώρα οι παράμετροι 3, 10 και 2 της συνάρτησης **range()**; Σου θυμίζει κάτι;



Κουράστηκες, ε; Τάιμ-άουτ, και μετά κάτι εντελώς νέο! (που λένε κι οι Μόντι Πάιθον)

Πριν ξεκινήσεις, έφτασε η ώρα να ξεκαθαριστεί κάτι. Ως τώρα, τα προγράμματα που γράφεις, τα εκτελείς στο κέλυφος του IDLE. Αυτό είναι ιδιαίτερα βολικό, αφού στο ίδιο κέλυφος μπορείς να κάνεις απευθείας και μεμονωμένες δοκιμές. Όμως τα προγράμματα σε Python, προορίζονται να μεταγλωττίζονται και να εκτελούνται από την Python, μέσα στο κέλυφος του Λειτουργικού Συστήματος (Λ.Σ.) του υπολογιστή του χρήστη. Στην περίπτωση των Windows το κέλυφος αυτό είναι η λεγόμενη **κονσόλα cmd** ή, αλλιώς, η **γραμμή εντολών**. Στο 99% των περιπτώσεων, η συμπεριφορά στο κέλυφος IDLE και σε αυτό του Λ.Σ. είναι πανομοιότυπη, όχι όμως πάντα. Ο προχωρημένος προγραμματιστής δοκιμάζει πάντοτε τις δημιουργίες του στα κελύφη πολλαπλών λειτουργικών συστημάτων.



Η συνάρτηση **getpass()** λειτουργεί ακριβώς όπως η **input()**, με τη διαφορά ότι αυτό που της εισάγει ο χρήστης δεν εμφανίζεται στην οθόνη.

21. Ξεκίνησε στο **συντάκτη** το πρόγραμμα **kremala.py**.

Το πρόγραμμά σου αρχικά θα ζητάει τη μυστική λέξη από τον ένα παίκτη. Προφανώς, τη λέξη αυτή δεν θα την βλέπει ο άλλος παίκτης που θα κληθεί να την βρει. Αυτό μπορεί να γίνει, αν την πληκτρολογεί χωρίς να εμφανίζεται στην οθόνη, όπως κάνουμε με τους κωδικούς, τα password. Η βιβλιοθήκη **getpass** περιλαμβάνει την κατάλληλη συνάρτηση, την **getpass()**. Πρώτα χρειάζεται να την εισαγάγεις:

```
import getpass
```

Τώρα, μπορείς να γράψεις τις κατάλληλες εντολές για να πάρεις από τον πρώτο παίκτη τη μυστική λέξη στη μεταβλητή **secret**.

```
secret = getpass.getpass("Δώσε τη μυστική λέξη: ")
```

F5 Τρέξε το πρόγραμμά σου ως εδώ. Βλέπεις ότι η `getpass` είναι από αυτές τις λίγες περιπτώσεις που στο κέλυφος `IDLE` δεν δουλεύει σωστά. Θα την αντικαταστήσεις με μια πανομοιότυπη `input()`; Πώς θα αποτρέψεις να δει ο άλλος παίκτης τη μυστική λέξη; Με μπόλικες κενές γραμμές που θα ακολουθήσουν.

```
secret = input("Δώσε τη μυστική λέξη: ")
print(... * "\n")
```

22. Τώρα που έχεις τη μυστική λέξη, το επόμενο βήμα είναι το πρόγραμμά σου να ζητάει από το δεύτερο, πλέον, παίκτη, **επαναληπτικά** από ένα γράμμα τη φορά. Με ποια εντολή εκτελείται επαναληπτικά κάτι ξανά και ξανά;



Ναι, ο άλλος παίκτης μπορεί να κλέψει αν θέλει. Η «προστασία» δεν είναι πολύ σπουδαία.

F5 Το δοκίμασες; Χρησιμοποίησες μια `while` με συνθήκη τη σταθερά `True`;

23. Ακολούθησε αυτήν την υλοποίηση στο εξής. Το γράμμα κάθε φορά θα το κρατάει η μεταβλητή **letter**:

```
secret = input("Δώσε τη μυστική λέξη: ")
print(50 * "\n")
while True:
    letter = input("Δώσε ένα γράμμα: ")
```



Διακόπτεις την εκτέλεση ενός προγράμματος με το `Ctrl+C` (στα αγγλικά, όμως).

24. Όπως και στη χειρόγραφη κρεμάλα, καλό είναι πριν το μήνυμα "Δώσε ένα γράμμα", να φαίνεται η μυστική λέξη με παύλες: π.χ., **π_ _ _ ο** για τη μυστική λέξη **παγωτό**. Αλλά με ποιον τρόπο μπορείς να φτιάξεις το "π_ _ _ ο"; Χρειάζεσαι:

- Το **πρώτο γράμμα** από τη **secret**; (χμ, πώς το επιλέγεις;)
- Το **τελευταίο γράμμα** της **secret**; (χμ, πώς το επιλέγεις;)
- **Αρκετές (κάτω) παύλες** ενδιάμεσα
 - Πόσες παύλες; Έχουν σχέση με **το μήκος** της **secret**; Αν το πλήθος τους το κρατά η μεταβλητή **underlines**, γράψε την εντολή που το υπολογίζει:
 - `underlines =`
 - Φτιάξε τώρα ένα αλφαριθμητικό με, **underlines** το πλήθος, κάτω παύλες:

.....



Και όλα τα παραπάνω **θα τα ενώσεις** στη σειρά που θες σε ένα αλφαριθμητικό! Επομένως μάλλον έχεις όλα τα απαραίτητα συστατικά για να φτιάξεις τη μυστική λέξη με παύλες. Θα την αναθέσεις στη μεταβλητή **guess**. Δεν πρέπει να είναι ιδιαίτερα δύσκολο να συμπληρώσεις τον τροποποιημένο κώδικα:

```
secret = input("Δώσε τη μυστική λέξη: ")
print(50 * "\n")
underlines = .....
guess = ..... + ("_" * underlines) + .....
while True:
    print(guess)
    letter = input("Δώσε ένα γράμμα: ")
```

Οι παύλες είναι όσες το μήκος της **secret**, μείον 2 χαρακτήρες. Χρειάζεσαι τη `len()`. Η **guess** θα ξεκινάει με τον πρώτο χαρακτήρα της **secret**, και θα τελειώνει με τον τελευταίο της **secret**.

F5

Εκτέλεσε πάλι το πρόγραμμά σου.

25. Το πρόγραμμά σου ζητάει ένα-ένα τα γράμματα, αλλά δεν κάνει τίποτα με αυτά! Τι πρέπει να κάνει για κάθε γράμμα letter που διαβάζει;



26. Στο παραπάνω ερώτημα πρέπει να ξεχώρισης τι πρέπει να γίνει, ανάλογα με το αν το γράμμα **υπάρχει** μέσα στη **secret**, ή αν **δεν υπάρχει** μέσα στη **secret**.

Ξεκίνησε με το πιο απλό: Πώς θα αναγνωρίσεις σε **rython** αν ένα γράμμα υπάρχει μέσα σε μία λέξη; Θα ξαναγυρίσεις στο κέλυφος.



27. Ας υποθέσουμε ότι η μυστική λέξη είναι "σουσάμι". Γράψε στο κέλυφος:

```
secret = "σουσάμι"
```

και έστω ότι το γράμμα που έδωσε ο παίκτης είναι το βήτα. Γράψε στο κέλυφος:

```
letter = "β"
```

Δοκίμασε τώρα την εξής εντολή στο κέλυφος:

```
letter in secret
```

Τι απάντησε το κέλυφος;

28. Συνέχισε, αλλάζοντας τιμές στο letter και δοκιμάζοντας ξανά την εντολή:

```
letter in secret
```

Για κάθε γράμμα, δοκίμασε επίσης την παραλλαγή:

```
letter not in secret
```

Σε τι διέφερε το **in** από το **not in**;

Κάνε τις απαραίτητες δοκιμές για να συμπληρώσεις τον πίνακα:

Για το γράμμα...	το in secret επιστρέφει...	το not in secret επιστρέφει
"α"		
"ά" (τονισμένο πεζό άλφα)		
"Α"		
"Ι"		

Ξεχωρίζει η **rython** το μικρό άλφα από το κεφαλαίο;

Ξεχωρίζει η **rython** το τονισμένο από το άτονο φωνήεν;

29. Με βάση τα παραπάνω, ο παίκτης θα πρέπει να δοκιμάζει χωριστά το α, το ά και το Α, μα και το Ι! Αυτό είναι λίγο άδικο, δε νομίζεις; Να χάνει 4 προσπάθειες για 1 γράμμα; Δεν τα κάνεις όλα άτονα κεφαλαία να τελειώνεις;

Για να το κάνεις αυτό, θα χρησιμοποιήσεις μία συνάρτηση που θα σου τη δώσουμε έτοιμη. Παρόλο που την φτιάχνει ο προγραμματιστής (εμείς, δηλαδή) περιέχει λίγο προχωρημένα στοιχεία της γλώσσας, που θα τα μάθεις όταν προχωρήσεις στη γλώσσα! Να η συνάρτηση (θυμάσαι τη σύνταξη με την εντολή **def**):

```
def capitalized(word):
    return word.upper().translate(str.maketrans("ΑΕΗΙΪΟΥΨΩ", "ΑΕΗΙΙΟΥΥΩ"))
```

Η **rython** έχει 3 λογικούς τελεστές: **or**, **and**, **not**
Αντιστοιχούν στις λογικές πράξεις - γνωστές από τον Αριστοτέλη- της διάζευξης, της σύζευξης και της άρνησης.

Η **or** για να αληθεύει, ζητάει να ισχύει έστω μία από τις παραμέτρους της, ενώ η **and** ζητάει να ισχύουν όλες.

Η **not** αντιστρέφει την ισχύ της παραμέτρου της.

Το **in** είναι λογικός τελεστής για συλλογές και ακολουθίες. Επιστρέφει τιμή **True**, όταν η πρώτη παράμετρος αποτελεί στοιχείο της δεύτερης. Είναι γενίκευση του μαθηματικού «ανήκει» για στοιχεία και σύνολα.

Η συνάρτηση αυτή μετατρέπει και επιστρέφει σε άτονα κεφαλαία γράμματα την αλφαριθμητική παράμετρο που της δίνουμε (τη μυστική λέξη για μας). Δες στο κέλυφος το ακόλουθο παράδειγμα για να καταλάβεις τη χρήση της.



```
def capitalized(word):
    return word.upper().translate(str.maketrans("ΑΕΗΤΙΪΟΥΨΩ", "ΑΕΗΙΙΟΥΥΩ"))

a="δοκιμή"
capitalized(a)
αποτέλεσμα
letter="α"
capitalized(letter)
αποτέλεσμα
Καλό, ε;
```



30. Γύρισε στο πρόγραμμά σου στον συντάκτη. Με βάση τα προηγούμενα, θα κάνεις τώρα τις εξής αλλαγές:

1. Τοποθέτησε τη συνάρτηση capitalized στο πάνω μέρος του αρχείου σου, αντιγράφοντάς την με ακρίβεια (ίσως το έκανες ήδη)
2. Κάλεσέ την κατάλληλα, ώστε η secret να μετατραπεί σε άτονα κεφαλαία. Δοκίμασε να την τοποθετήσεις και να την καλέσεις. Τρέξε το πρόγραμμά σου για να δεις ότι λειτουργεί! Μετατρέπει σωστά την secret; Μήπως να βάλεις προσωρινά, και μόνο για το βήμα αυτό μία εντολή **print(secret)** στο κατάλληλο σημείο, ώστε να διαπιστώσεις αν η μετατροπή γίνεται σωστά; Πριν το επόμενο βήμα, θυμήσου να τη σβήσεις!
3. Έλεγξε μέσα στην while αν το γράμμα του παίκτη υπάρχει ή όχι στην κρυφή λέξη και ενέργησε αναλόγως.



Όμως **δεν θα την ελέγχεις απ' άκρη σ' άκρη**, αλλά θα παραλείπεις τα άκρα της, το πρώτο και τον τελευταίο γράμμα της. *Γιατί;* γιατί αυτά είναι φανερά. Όταν ο παίκτης δίνει ένα γράμμα, εννοεί να είναι στα κρυφά γράμματα, τα ενδιάμεσα.



α. Χρειάζεσαι μία if. Δοκίμασε να την τοποθετήσεις μέσα στην while, αμέσως μόλις πάρεις το γράμμα του παίκτη, ώστε:

- i. όταν το γράμμα υπάρχει στη secret να εκτελεί κάποια **print("βρήκες γράμμα")**, ενώ
- ii. όταν δεν υπάρχει στη secret να εκτελεί μια **print("ωχ, δεν το πέτυχες...")**.

Τρέξε το πρόγραμμά σου για να δεις ότι λειτουργεί! Εμφανίζει τα μηνύματα σωστά; Κάνε αρκετές δοκιμές τόσο με γράμματα της λέξης όσο και με γράμματα που δεν υπάρχουν στη λέξη. *Μήπως ο παίκτης πρέπει να δίνει τα γράμματα σε μία συγκεκριμένη μορφή;*



- i. Πώς θα επιτρέψεις στον παίκτη να δίνει το κάθε γράμμα όπως αυτός θέλει, και το πρόγραμμα να το μετατρέπει σε άτονο κεφαλαίο;
- ii. Το βρήκες; Κάνε την αλλαγή, και μην ξεχάσεις!

Τρέξε το πρόγραμμά σου για να δεις ότι είναι εντάξει.



31. Αν όλα έχουν πάει καλά, το πρόγραμμά σου θα είναι τώρα κάπως έτσι:

```
def capitalized(word):
    return word.upper().translate(str.maketrans("ΑΕΗΙΪΟΥΨΩ", "ΑΕΗΙΙΟΥΥΩ"))
secret = input("Δώσε τη μυστική λέξη: ")
print(50 * "\n")
secret = ...βοήθεια: χρειάζεσαι την capitalized πάνω στην ίδια τη secret...
underlines = len(secret)-2
guess = secret[0] + ("_" * underlines) + secret[-1]
while True:
    print(guess)
    letter = input("Δώσε ένα γράμμα: ")
    letter = ...βοήθεια: χρειάζεσαι την capitalized πάνω στο letter...
    if ..... in .....:
        print("βρήκες ένα γράμμα")
    else:
        print("ωχ, πρόσεξε!")
```



32. Δες τώρα την περίπτωση που ο παίκτης έχει βρει ένα γράμμα. Δεν είναι προτιμότερο, μαζί με το μήνυμα "βρήκες ένα γράμμα", να αλλάζει το περιεχόμενο της guess, ώστε να μπει το γράμμα στη σωστή του θέση -όσες φορές υπάρχει- αντικαθιστώντας αντίστοιχες παύλες;

Για παράδειγμα, αν το πρώτο γράμμα που πληκτρολογεί ο παίκτης είναι το "Υ", και η μυστική λέξη είναι το "ΣΟΥΣΑΜΙ", δεν θα ήθελες η μεταβλητή guess:

από "Σ_ _ _ _ Ι" να γίνει "Σ_ Υ _ _ Ι";

Χμ, δύσκολο; Έχεις καμιά ιδέα;



.....
.....

33. Θα βοηθήσουμε λίγο. Δες τι έχεις ως τώρα. Καταρχήν το **letter** που έχει δώσει ο παίκτης. Επίσης τη μυστική λέξη **secret**, χωρίς τα άκρα της. Έστω ότι θες να συγκρίνεις το **letter** με το **τρίτο γράμμα της secret**· μπορείς; Για συμπλήρωσε την παρακάτω if που συγκρίνει το letter με το τρίτο γράμμα της λέξης secret;

if:

print("Το τρίτο γράμμα το βρήκες!")



Αφιέρωσε λίγο χρόνο εδώ για να το σκεφτείς! Δες και τα προηγούμενα!

34. Ναι, αλλά μόνο με το τρίτο γράμμα θα πρέπει να συγκρίνω, μπορεί ήδη να αναρωτιέσαι. Είναι πολύ ειδική περίπτωση! Μήπως θα μπορούσες να χρησιμοποιήσεις μία for για να συγκρίνεις το **κάθε γράμμα της λέξης secret** με το **letter**; Θα βάλεις την παραπάνω if μέσα σε μία for. Προσπάθησε να συμπληρώσεις κατάλληλα τα κενά.

for i in range(.....,.....)

if:

print("Βρήκες το", i+1, "ο γράμμα!")



Η εντολή **for** φτιάχτηκε ειδικά για να διευκολύνει τον προγραμματιστή με τις ακολουθίες. Ο **μετρητής** του **for** είναι δί-δυμος αδερφός με τους **δείκτες** στις ακολουθίες. Η **while** είναι αρκετή για όλες τις επαναλήψεις, με τη **for** όμως γλυτώνουμε τυπικό κώδικα στις ακολουθίες.

35. Την παραπάνω for θα την βάλεις στο πρόγραμμά σου, για την περίπτωση που ο παίκτης έχει πετύχει κάποιο γράμμα:

```
[.....]
while True:
    print(guess)
    letter = input("Δώσε ένα γράμμα: ")
    letter = capitalized(letter)
    if letter in secret[1:-1]:          #secret χωρίς άκρα
        for i in range(....., .....):
            if .....:
                print("Βρήκες το", i+1, "ο γράμμα!")
        else:
            print("ωχ, πρόσεξε!")
```



Συμπλήρωσε σωστά τα κενά; Πώς πάει η εκτέλεση του προγράμματος τώρα;

36. Προχωράμε ωραία! Τίθεται τώρα το εξής ερώτημα:

Βρήκες ένα γράμμα (το letter) κατά την εκτέλεση του for και ξέρεις και πού βρίσκεται (στη θέση i –αλλά θυμάσαι ότι η αρίθμηση ξεκινά από 0). Πώς θα μεταβάλλεις την guess κατάλληλα, ώστε να το εμφανίζει; Θυμήσου από τα προηγούμενα ότι:

- Μπορείς να **ενώσεις** μεταξύ τους αλφαριθμητικά με το +, π.χ. guess + secret,
- **τμήματα αλφαριθμητικού** παίρνεις με τεμαχισμό [από:μέχρι], π.χ. guess[0:5],
- το γράμμα που βρήκες είναι το letter.

Προσπάθησε, βάσει των παραπάνω, να συναρμολογήσεις, σαν να ήταν τρενάκι Lego, το "Σ_Y__I" από το "Σ____I" και το "Y".



37. Έφτιαξες το νέο τρενάκι; Θες το τμήμα της guess πριν το γράμμα i (guess[:i]), το κομμάτι της guess μετά το i (guess[i+1:]) και ενδιάμεσα, ακριβώς στη θέση i, το letter. Βάλ' τα στην κατάλληλη σειρά και συμπλήρωσε τον κώδικα:

```
[.....]
while True:
    print(guess)
    letter = input("Δώσε ένα γράμμα: ")
    letter = capitalized(letter)
    if letter in secret[1:-1]:
        for i in range(1, len(secret)-1):
            if letter == secret[i]:
                print("Βρήκες το", i+1, "ο γράμμα!")
                guess = ..... + ..... + .....
        else:
            print("ωχ, πρόσεξε!")
```



Αν έκανες καλή δουλειά, θα το δεις στην εκτέλεση!

38. Όλα καλά, μέχρι τώρα, αλλά το πρόγραμμά σου έχει ένα βασικό μειονέκτημα. Μπορείτε να σκεφτείς ποιο; Τι χρειάζεται ακόμα απαραίτητα για να γίνει όπως η κανονική κρεμάλα; (και δεν εννοούμε τη ζωγραφιά της κρεμάλας –το αφήνουμε αυτό για πιο προχωρημένο όμιλο κώδικα...)



39. Πρόκειται να κρεμάσεις ποτέ κανέναν, με το πρόγραμμά σου ως τώρα; Η μόνη τιμωρία για τα λάθος γράμματα είναι το μήνυμα "ωχ, πρόσεξε". Στην περίπτωση που δεν υπάρχει το letter, τι πρέπει να κάνεις; (τι ρόλο παίζει η ζωγραφιά;)

40. Πρέπει να μετράς πόσα λάθος γράμματα έχει δώσει ο παίκτης. Καθόλου δύσκολο! Θες μια μεταβλητή (ας την πούμε attemptsLeft, στην οποία θα δώσεις αρχική τιμή 7 (6-7 μέρη έχει η ζωγραφιά), και σε κάθε λάθος γράμμα θα τη μειώνεις κατά ένα. Αν η attemptsLeft μηδενιστεί, έχουμε κρέμασμα, και θα κάνεις break από την while (το θυμάσαι το break, ε;). Συμπλήρωσε, λοιπόν, την προγραμματάρα σου:

```
def capitalized(word):
    return word.upper().translate(str.maketrans("ΑΕΗΙΪΟΥΨΩ", "ΑΕΗΙΙΟΥΥΩ"))
secret = input("Δώσε τη μυστική λέξη: ")
print(50 * "\n")
secret = capitalized(secret)
underlines = len(secret)-2
guess = secret[0] + ("_" * underlines) + secret[-1]
attemptsLeft = .....
while True:
    print(guess)
    letter = input("Δώσε ένα γράμμα: ")
    letter = capitalized(letter)
    if letter in secret[1:-1]:
        for i in range(1, len(secret)-1):
            if letter == secret[i]:
                print("Βρήκες το", i+1, "ο γράμμα!")
                guess = guess[:i] + letter + guess[i+1:]
    else:
        print("ωχ, πρόσεξε!")
        attemptsLeft .... 1 #μείωσε κατά 1
        if .....: #μήπως μηδενίστηκε κανείς;
            break
```



Αν έκανες καλή δουλειά, θα φανεί στο χειροκρότημα στην εκτέλεση. Πάμε, πολλές δοκιμές!

41. Κοντεύεις την τελειότητα, που λένε. Έχεις, όμως, ένα μικρό πρόβλημα. Το εντόπισες; Τι γίνεται αν ο παίκτης δώσει το ίδιο ανύπαρκτο στη λέξη γράμμα πολλές

φορές; Για παρατήρησέ το. Δώσε 7 φορές το ίδιο ανύπαρκτο γράμμα. Πόσες φορές τιμωρείς τον παίκτη; Και τον κρεμάς τελικά; Είναι δίκαιο αυτό;



42. Πρέπει με κάποιο τρόπο **να θυμάσαι τα γράμματα** που έχει δώσει ο παίκτης, και να μετράνε μόνο την πρώτη φορά. Πρέπει, συνεπώς, να τα κρατάς σε ένα σακί, για να ελέγχεις κάθε επόμενο. Το σακί από (μοναδικά) στοιχεία είναι στα μαθηματικά το σύνολο, και, ναι, σωστά μαντεύεις, το ίδιο και στην python: **σύνολο (set)**. Θα ανοίξουμε άλλη μια παρένθεση εδώ. Το έχεις καταλάβει ότι *παρένθεση σημαίνει επιστροφή στο κέλυφος*, ε; Πάμε εκεί να δούμε πώς φτιάχνεις σύνολα και πώς ελέγχεις αν κάτι ανήκει σ' αυτά.



43. Γράψε:

```
passed = set()
```

Συγχαρητήρια, μόλις έφτιαξες ένα κενό σύνολο! (θυμάσαι στα μαθηματικά; \emptyset). Χρειάστηκε τη συνάρτηση **set()**. Το ονόμασες passed.

44. Βάλε μερικά στοιχεία στο passed. Θα χρειαστείς το σύμβολο της ένωσης συνόλων, την κατακόρυφη γραμμή |. Θα πληκτρολογήσεις Shift + \ (συνήθως, πάνω από το Enter είναι το πλήκτρο:



```
passed = passed | set("a")
```

```
# ή passed = passed | {"a"}
```

```
# ή passed |= {"a"}
```

Η πράξη της ένωσης γίνεται ανάμεσα σε σύνολα μόνο, γι' αυτό έβαλες το χαρακτήρα "a" μέσα σε ένα σύνολο. Ποιες άλλες πράξεις συνόλων θυμάσαι; Όπως και στις άλλες μεταβλητές, ένα σύνολο το βλέπεις στο κέλυφος απλά με το όνομά του:



Η python υποστηρίζει και τις 4 πράξεις συνόλων, με αντίστοιχους τελεστές:

45. Δοκίμασε στο κέλυφος να βάλεις στο passed τα παρακάτω γράμματα:

Για να βάλω στο σύνολο το	Γράφω στην python ...
"a"	
"b"	
"c", "d"	
"a" (ξανά)	
"d" (ξανά)	

s1 | s2 ένωση (union)

s1 & s2 τομή (intersection)

s1 - s2 διαφορά (difference)

s1 ^ s2 συμμετρική διαφορά (symmetric difference)

Δες τώρα το σύνολο passed. Τι παρατηρείς; Τι συμβαίνει με τα γράμματα που έβαλες πολλές φορές; Η python είναι πιστή στα μαθηματικά!

46. Ωραία, είδες πώς φτιάχνεις σύνολα, είδες και πώς βάζεις μέσα στα σύνολα νέα στοιχεία. Μαντεύεις πώς ελέγχεις αν κάτι ανήκει σε ένα σύνολο; Τον ξέρεις τον τελεστή! Πώς θα ελέγξεις αν το "b" είναι μέσα ;-) στο passed; Πάμε:

```
if .....
```

```
    print("Μέσα είναι")
```

```
# δοκίμασε να γράψεις και μόνη της τη συνθήκη. Τι απάντηση περιμένεις;
```



47. Βρήκες τον τελεστή **in**, έτσι; δεν κλέβεις... Επιστροφή στον συντάκτη και στο πρόγραμμα, για τις τελευταίες αλλαγές -το πιστεύεις;



1. Θες ένα σύνολο `passed`,
2. στο οποίο θα βάζεις κάθε γράμμα που δίνει ο παίκτης,
3. και το οποίο θα συμβουλευέσαι μόλις ο παίκτης δίνει γράμμα· αν αυτό είναι ήδη στο σύνολο, θα το παραβλέψεις και θα ζητάς το επόμενο...

Ας δούμε μαζί τις τέσσερις αλλαγές μόλις διαβάξεις το γράμμα του παίκτη:

```
passed = set()
while True:
    print(guess)
    letter = input("Δώσε ένα γράμμα: ")
    letter = capitalized(letter)
    if passed .....
    if letter in secret[1:-1] and letter not in passed:
        for i in range(1, len(secret)-1)
            if letter == secret[i]:
                print("Βρήκες το", i+1, "ο γράμμα!")
                guess = guess[:i] + letter + guess[i+1:]
    elif letter not in passed:
        print("ωχ, πρόσεξε!")
        attemptsLeft -= 1
        if attemptsLeft == 0:
            break
    passedLetters |= {letter}
```

Εξηγήσαμε τον
τελεστή **and**
στη σελίδα 6.



F5

Κοντεύουμε, αν κάνεις τις απαραίτητες δοκιμές!

F5

48. Δουλεύουν όλα καλά; Σχεδόν... Έλεγξε δύο πράγματα ακόμη: Τερματίζει το πρόγραμμά σου και στις δύο περιπτώσεις; Και όταν βρεθεί η λέξη και όταν κρεμαστεί ο παίκτης; Για δες!



Κάτι πάει στραβά όταν η λέξη έχει βρεθεί, ε; Ακόμη, δεν θα έπρεπε να βλέπουμε ένα μήνυμα για την κατάληξη του παιχνιδιού;

Θα αντικαταστήσεις τις τρεις τελευταίες γραμμές μέσα στο `while` με κάτι πιο σύνθετο –αλλά όχι περίπλοκο. Ας το σκεφτούμε:

- Μόλις έπραξες ό,τι έπρεπε με το γράμμα του παίκτη, εφόσον δεν είχε ξαναπεράσει: αντικατέστησες παύλες αν ήταν σωστό, ή μείωσες κατά μία τις προσπάθειες του παίκτη.
- Τώρα έχεις τρεις περιπτώσεις, για τις οποίες θα χρειαστείς ένα `if...elif...else`:
 - να μην έχει άλλες προσπάθειες ο παίκτης, οπότε δίνεις κατάλληλο μήνυμα κρεμάσματος και `break`
 - να βρέθηκε η λέξη (πώς το ελέγχεις;), οπότε δίνεις κατάλληλο μήνυμα κρεμάσματος και `break`
 - ούτε το 'να ούτε τ' άλλο, οπότε προσθέτεις το γράμμα στα περασμένα, και καλό είναι να λες και (1) πόσες προσπάθειες μένουν (2) ποια γράμματα πέρασαν.



Πάμε! Δες ολόκληρο το πρόγραμμά σου, και συμπλήρωσε ξανά:

```
def capitalized(word):
    return word.upper().translate(str.maketrans("ΑΕΗΙΪΟΥΥΩ", "ΑΕΗΙΙΟΥΥΩ"))
secret = input("Δώσε τη μυστική λέξη: ")
print(50 * "\n")
secret = capitalized(secret)
underlines = len(secret)-2
guess = secret[0] + ("_" * underlines) + secret[-1]
attemptsLeft = 7
passed = set()
while True:
    print(guess)
    letter = input("Δώσε ένα γράμμα: ")
    letter = capitalized(letter)
    if letter in secret[1:-1] and letter not in passed:
        for i in range(1, len(secret)-1):
            if letter == secret[i]:
                print("Βρήκες το", i+1, "ο γράμμα!")
                guess = guess[:i] + letter + guess[i+1:]
    elif letter not in passed:
        print("ωχ, πρόσεξε!")
        attemptsLeft -= 1
    if .....:
        print("Σε κρέμασα! Η λέξη ήταν", ....)
        .....
        #αν τέλειωσαν οι προσπάθειες
        #βγες απ' το while
    elif secret==guess:
        print("Μπράβο, την βρήκες!")
        .....
        #αν βρήκε τη λέξη
        #βγες απ' το while
    else:
        .....
        #βάλε το γράμμα στα περασμένα
        print("Ως τώρα έδωσες :", ..... )
        print("Προσπάθειες ακόμη:", ..... )
```

F5

ΜΠΡΑΒΟ!



Αν τα έκανες όλα καλά, **τώρα** έχεις ένα ολοκληρωμένο πρόγραμμα! Εύγε!

49. Αντέχεις ακόμη; Να δούλευες λίγο ακόμα την εμφάνιση της λέξης, ώστε να μην κολλάνε μεταξύ τους τα γράμματα και τα κενά; Ουσιαστικά θες να αντικαταστήσεις την `print(guess)` παραπάνω, με μια δική σου εκτυπωτική συνάρτηση, πες την `sparseprint(word)`. Θα έχει μία παράμετρο, την `word`. Και τι θα κάνει με αυτήν;

.....

.....

50. Η `sparseprint(word)` θα τυπώνει την παράμετρο `word` γράμμα-γράμμα αφήνοντας ένα χαρακτήρα διαστήματος ανάμεσά τους. Ξέρεις πώς να ξεκινήσεις;

```
def sparseprint(word):
    for letter in word:
        print(letter, end=" ")
    print()
```





Να εξηγήσουμε την παράμετρο end στην print(). Είναι μια παράμετρος προαιρετική, γιατί έχει εξ ορισμού τιμή, αν την παραλείψουμε. Η end, λοιπόν, δηλώνει ποιος θα είναι η τελευταία τιμή που θα εκτυπώσει, μετά τις άλλες παραμέτρους της. Και έχει εξ ορισμού τιμή τον ειδικό χαρακτήρα που ορίζει να αλλάξει η γραμμή στο τέλος της εκτύπωσης (ο χαρακτήρας που δηλώνεται με \n). Γι' αυτό την έξοδο κάθε print την βλέπουμε σε διαφορετική γραμμή στην οθόνη.

Μαντεύεις τι κάνει η τιμή end = " "; Ορίζει αντί για αλλαγή γραμμής να εισάγεται ο χαρακτήρας του διαστήματος. Μπίνγκο! Στο τέλος αλλάξαμε γραμμή.



```
def sparseprint(word):
    for letter in word:
        print(letter, end=" ")
    print()

def capitalized(word):
    return word.upper().translate(str.maketrans("ΑΕΗΤΙΟΥΥΩ", "ΑΕΗΙΙΟΥΥΩ"))

secret = input("Δώσε τη μυστική λέξη: ")
print(50 * "\n")
secret = capitalized(secret)
underlines = len(secret)-2
guess = secret[0] + ("_" * underlines) + secret[-1]
attemptsLeft = 7
passed = set()
while True:
    print(guess)
    letter = input("Δώσε ένα γράμμα: ")
    letter = capitalized(letter)
    if letter in secret[1:-1] and letter not in passed:
        for i in range(1, len(secret)-1):
            if letter == secret[i]:
                print("Βρήκες το", i+1, "ο γράμμα!")
                guess = guess[:i] + letter + guess[i+1:]
    elif letter not in passed:
        print("ωχ, πρόσεξε!")
        attemptsLeft -= 1
    if attemptsLeft == 0:
        print("Σε κρέμασα! Η λέξη ήταν", secret)
        break
    elif secret==guess:
        print("Μπράβο, την βρήκες!")
        break
    else:
        passed |= {letter}
        print("Ως τώρα έδωσες :", passed)
print("Προσπάθειες ακόμη:", attemptsLeft)
```



ΜΠΡΑΒΟ!

Αφού κατάφερες να φτάσεις ως εδώ, μπορείς να ζητήσεις το
επόμενο, μεγαλύτερο φύλλο εργασίας!

Συγχαρητήρια λοιπόν!

Ελπίζουμε να το διασκέδασες όσο κι εμείς!