Μάντεψε τον Αριθμό

Ενδεικτικές Απαντήσεις Φύλλου Εργασίας

3

10 Σεπτεμβρίου 2016
10:22

Μάντεψε!

1. Το πρόγραμμα θα ξεκινά επιλέγοντας τον μυστικό αριθμό. Η τιμή που επιλέγει το πρόγραμμά μας θα αποθηκεύεται σε μια μεταβλητή με το όνομα `secret`.

   Έστω ότι το πρόγραμμα επιλέγει το τυχερό 13. *Προσθέστε* στο πρόγραμμα μια εντολή η οποία ορίζει ότι η τιμή της `secret` είναι 13.

   Θα ξεκινήσουμε το πρόγραμμα με την εντολή:

   ```
   secret = 13
   ```

   Τώρα μπορούμε ν' αναφερόμαστε στο μυστικό αριθμό με το όνομα `secret`, χωρίς να έχει σημασία ποια είναι η τιμή του.

2. *Συμπληρώστε* το πρόγραμμα έτσι ώστε να ζητά από το χρήστη να μαντέψει τον μυστικό αριθμό, εμφανίζοντας κατάλληλη προτροπή. Για παράδειγμα:

   ```
   Μάντεψε τον αριθμό:
   24
   ```

   Αποθηκεύστε την τιμή που πληκτρολογεί ο χρήστης σε μια μεταβλητή `number`.

   Οι εντολές που θα προσθέσουμε στο πρόγραμμα είναι:

   ```
   print("Δώσε αριθμό:")
   number = int(input())
   ```

   Εκτελέστε το πρόγραμμα. Τα καταφέρατε;

   Ναι, το πρόγραμμα εμφανίζει το μήνυμα προτροπής στον χρήστη και στη συνέχεια ζητά τον αριθμό της επιλογής του.

3. Συμπληρώστε το πρόγραμμα, ώστε να εμφανίζει στον παίκτη τον αριθμό που πληκτρολόγησε, για παράδειγμα:

   ```
   Έδωσες τον αριθμό 24
   ```

   Η εντολή που θα προσθέσουμε στο πρόγραμμα είναι:

   ```
   print("Έδωσες τον αριθμό", number)
   ```

4. *Συμπληρώστε* το πρόγραμμά σας με μια **if**–**else**, έτσι ώστε να ελέγχει αν ο παίκτης βρήκε τον μυστικό αριθμό ή όχι και να του εμφανίζει ανάλογο μήνυμα. Για παράδειγμα, τώρα που ο μυστικός αριθμός είναι ο 13, υπάρχουν δύο πιθανές περιπτώσεις:

   ```
   Μάντεψε τον αριθμό:
   13
   Σωστά!
   ```

   ```
   Μάντεψε τον αριθμό:
   24
   Λάθος...
   ```

   Οι εντολές που θα προσθέσουμε στο πρόγραμμα είναι:

   ```
   if number == secret:
       print("Σωστά!")
   else:
       print("Λάθος...")
   ```

   Στη συνθήκη της **if** χρησιμοποιήσατε τη μεταβλητή `secret` ή συγκρίνατε την τιμή της `number` απευθείας με το 13;

   Παρόλο που μπορούμε να χρησιμοποιήσουμε απευθείας τον αριθμό 13 *στη συνθήκη της* **if**, *αποτελεί καλύτερη τακτική να χρησιμοποιήσουμε τη μεταβλητή* `secret`. *Όταν η συνθήκη της* **if** *είναι η* `number == 13` *τότε ελέγχουμε πάντα αν ο χρήστης έδωσε* συγκεκριμένα *τον αριθμό* 13. *Αντιθέτως, όταν η συνθήκη είναι* `number == secret`, *τότε ελέγχουμε αν ο χρήστης μάντεψε τον μυστικό αριθμό,* όποιος κι αν είναι αυτός.

   Στη δεύτερη περίπτωση, *τροποποιήστε* τη συνθήκη της **if**, έτσι ώστε η μεταβλητή `number` να συγκρίνεται με την `secret` και όχι με το 13.

   Ποιο πλεονέκτημα πιστεύετε ότι έχει η χρήση της μεταβλητής `secret`, αντί της σταθεράς 13;

   Με τον τρόπο αυτό, αν αλλάξουμε το μυστικό αριθμό, δηλαδή την τιμή της `secret` *δεν θα χρειαστεί καμία αλλαγή στη συνθήκη της* **if**.

5. Εκτελέστε το πρόγραμμα δύο φορές και παίξτε το ρόλο του χρήστη. Την πρώτη φορά δώστε σωστά τον μυστικό αριθμό, ενώ την επόμενη δώστε έναν διαφορετικό.

   Μήπως το πρόγραμμά σας εμφανίζει το μήνυμα `"Λάθος..."`, ακόμα κι όταν ο χρήστης μαντέψει σωστά τον μυστικό αριθμό; Σε αυτή την περίπτωση, δοκιμάστε να κάνετε την παρακάτω τροποποίηση:

   ```
   number = int( input() )
   ```

   Γιατί είναι απαραίτητη η χρήση της `int`; Τί πιστεύετε ότι συμβαίνει όταν δεν χρησιμοποιείται η `int` και το πρόγραμμα δεν λειτουργεί;

Με το == ελέγχεται αν δύο τιμές είναι ίσες. Μην το συγχέετε με το = που χρησιμοποιείται για να δώσουμε τιμή σε μια μεταβλητή.

Με το != ελέγχεται αν δύο τιμές είναι διαφορετικές.

Η `input()` *επιστρέφει στο πρόγραμμα μια αλφαριθμητική τιμή, δηλαδή* κείμενο. *Η χρήση της* `int()` *είναι απαραίτητη προκειμένου να μετατραπεί αυτό το κείμενο σε ακέραιο αριθμό και να έχει νόημα η σύγκριση με την τιμή της* `secret`.

Αν δεν έχει προηγηθεί η μετατροπή της `number` *σε ακέραιο με την* `int()`, *η συνθήκη* `number == secret` *θα ελέγχει αν ένα κείμενο είναι ίσο με έναν αριθμό. Αυτό δεν μπορεί ποτέ να είναι αληθές. Ακόμα κι αν ο χρήστης πληκτρολογήσει* `"13"`, *και ο μυστικός αριθμός είναι* `13`, *αυτές οι δύο τιμές δεν θεωρούνται ίσες, αφού η μία είναι αλφαριθμητική και η άλλη ακέραια. Εφόσον η συνθήκη που ελέγχεται με την* **`if`** *δεν μπορεί ποτέ να είναι αληθής, το μήνυμα* `"Σωστά"` *δεν θα εμφανιστεί ποτέ.*

6. *Προσθέστε* στο πρόγραμμα μια εντολή που θα εμφανίζει τον μυστικό αριθμό στην περίπτωση που ο παίκτης απαντήσει λάθος, όπως παρακάτω:

```
Ο μυστικός αριθμός ήταν ο 13.
```

Βεβαιωθείτε κι εδώ ότι αποκαλύπτετε στο χρήστη την τιμή της `secret`, και όχι την σταθερά `13`.

Η εντολή που θα προσθέσουμε στο πρόγραμμα είναι:

```
    print("Ο μυστικός αριθμός ήταν ο", secret)
```

Εκτελέστε το πρόγραμμα δύο φορές. Στην πρώτη δώστε σωστά τον μυστικό αριθμό, ενώ στην επόμενη δώστε έναν διαφορετικό. Εμφανίζει το μυστικό αριθμό *μόνο* στην περίπτωση που ο παίκτης δεν τον μαντέψει; Αν όχι, γιατί πιστεύετε ότι συμβαίνει αυτό;

Η εντολή πρέπει να προστεθεί μέσα στην **`else`**, *επομένως πρέπει να προηγηθούν τέσσερα κενά, ώστε να δημιουργηθεί η κατάλληλη εσοχή. Σε διαφορετική περίπτωση θα εκτελείται ανεξάρτητα από τον αν ο παίκτης μαντέψει το μυστικό αριθμό.*

Αν ο μυστικός αριθμός εμφανίζεται είτε ο χρήστης τον μαντέψει, είτε όχι, τότε πιθανότατα δεν τοποθετήσατε τη νέα εντολή *μέσα* στην **`else`**, αλλά μετά από αυτή. Προσθέστε 4 κενά πριν την εντολή, για να υποδηλώσετε ότι κι αυτή ανήκει στην **`else`**.

Τί θα αλλάζατε στο πρόγραμμα όπως έχει μέχρι στιγμής; Πώς πιστεύετε ότι πρέπει να επεκταθεί για να γίνει πιο ενδιαφέρον;

Μετά από λίγες εκτελέσεις ο παίκτης θα αντιληφθεί ότι ο μυστικός αριθμός είναι πάντα το `13`. *Θα ήταν πιο ενδιαφέρον αν σε κάθε εκτέλεση ο μυστικός αριθμός έπαιρνε μια διαφορετική, τυχαία τιμή.*

Τυχαιότητα

Το παιχνίδι μας δεν αξίζει να το παίξεις πάνω από μία-δύο φορές, αν ο μυστικός αριθμός είναι πάντα το `13`. Αυτό που χρειαζόμαστε είναι να επιλέγεται κάθε φορά ένας διαφορετικός αριθμός. Για τον

σκοπό αυτό, θα χρειαστούμε τη βιβλιοθήκη `random`.

7. Στην αρχή του προγράμματος *προσθέστε* την εντολή εισαγωγής της βιβλιοθήκης `random`.

```
import random
```

Τροποποιήστε την εντολή του βήματος 1, όπου ορίζεται η τιμή της μεταβλητής `secret`, ως εξής:

```
secret = random.randint(1,32)
```

Εκτελέστε το πρόγραμμα αρκετές φορές. Τί είδους τιμές παρατηρείτε ότι παίρνει η μεταβλητή `secret`;

Η `secret` παίρνει τυχαίες τιμές.

Ποιος πιστεύετε ότι είναι ο ρόλος των παραμέτρων `1` και `32`; Τί συμβαίνει αν δοκιμάσετε άλλες τιμές στη θέση τους, για παράδειγμα `33` και `64`; Αν χρειαστεί εκτελέστε πάλι το πρόγραμμα αρκετές φορές προκειμένου να απαντήσετε στην ερώτηση.

Οι παράμετροι `1` και `32` καθορίζουν το διάστημα στο οποίο βρίσκεται η τυχαία τιμή. Αν χρησιμοποιήσουμε άλλους αριθμούς για παράδειγμα `33` και `64`, η τυχαία τιμή θα βρίσκεται μεταξύ των αριθμών `33` και `64`, συμπεριλαμβανομένων αυτών.

Πως θα χρησιμοποιούσατε τη `randint` για να προσομοιώσετε τη ρίψη ενός ζαριού, τη ρίψη ενός κέρματος ή την επιλογή ενός τυχαίου χαρτιού από μια τράπουλα;

ρίψη ζαριού:	`dice = random.randint(1, 6)`
ρίψη κέρματος:	`coin = random.randint(1, 2)`
επιλογή χαρτιού:	`card = random.randint(1, 52)`

Μπορείτε να σκεφτείτε άλλες περιπτώσεις στις οποίες θα χρειαζόσασταν τη `randint`;

Η `randint` μπορεί να χρησιμοποιηθεί σε οποιοδήποτε πρόγραμμα είναι απαραίτητη η παραγωγή τυχαίων αριθμών. Ενδεικτικά παραδείγματα είναι τα τυχερά παιχνίδια, όπως το Τζόκερ ή το Λόττο ή ένα πρόγραμμα που προσομοιώνει τη λειτουργία του "κουλοχέρη" (φρουτάκια).

8. Τί θα αλλάζατε στο πρόγραμμα όπως έχει μέχρι στιγμής; Πώς πιστεύετε ότι πρέπει να επεκταθεί για να γίνει πιο ενδιαφέρον;

Ο παίκτης έχει μόνο μια ευκαιρία να μαντέψει το μυστικό αριθμό. Θα θέλαμε το πρόγραμμα να εκτελείται επαναληπτικά, ώστε να έχει περισσότερες ευκαιρίες στη διάθεσή του.

Γύρω-Γύρω Όλοι

Θα θέλαμε να επεκτείνουμε το παιχνίδι έτσι ώστε ο παίκτης να έχει *επαναλαμβανόμενες* ευκαιρίες να βρει τον μυστικό αριθμό, ενώ ο μυστικός αριθμός *παραμένει ο ίδιος*.

9. *Προσθέστε* τη γραμμή που ακολουθεί *αμέσως μετά* από την εντολή του βήματος 7, που δίνει μια τυχαία τιμή στη μεταβλητή `secret`.

```
while True:
```

Προσθέστε τέσσερα κενά μπροστά από *όλες τις εντολές που ακολουθούν* τη **`while`**, σηματοδοτώντας έτσι ότι αυτές οι εντολές *εμφωλεύονται* στη **`while`**, δηλαδή περιέχονται σε αυτήν.

Εκτελέστε το πρόγραμμα. Ποια αλλαγή παρατηρείτε ότι επιφέρει η χρήση της **`while`**;

Μπορείτε να διακόψετε την εκτέλεση του προγράμματός σας με τον συνδυασμό πλήκτρων `Ctrl + C` .

*Οι εντολές που βρίσκονται μέσα στη **`while`** εκτελούνται ξανά και ξανά.*

Η εντολή του βήματος 7, που δίνει μια τυχαία τιμή στην μεταβλητή `secret`, βρίσκεται *πριν* από τη **`while`**. Ποια πιστεύετε ότι θα ήταν η διαφορά αν βρισκόνταν *μέσα* στη **`while`**;

Σε κάθε γύρο του παιχνιδιού ο μυστικός αριθμός θα ήταν διαφορετικός, αφού η εντολή `secret = random.randint(1,32)` θα εκτελούνταν σε κάθε επανάληψη, αποδίδοντας κάθε φορά μια νέα τυχαία τιμή στο μυστικό αριθμό.

Η εντολή του βήματος 6 εμφανίζει τον μυστικό αριθμό, στην περίπτωση που ο χρήστης δεν τον μαντέψει. Τώρα, αυτή η εντολή βρίσκεται μέσα στη **`while`** κι εκτελείται σε κάθε αποτυχημένη προσπάθεια του χρήστη. Γιατί αυτό είναι πρόβλημα;

Γιατί μετά την πρώτη αποτυχημένη προσπάθεια το πρόγραμμα εμφανίζει το μυστικό αριθμό στην οθόνη, επομένως στον επόμενο γύρο ο παίκτης θα γνωρίζει το μυστικό αριθμό και δεν θα χρειαστεί και πολλή προσπάθεια να τον μαντέψει!

Διαγράψτε την εντολή που εμφανίζει τον μυστικό αριθμό.

Παίξτε το παιχνίδι μέχρι να μαντέψετε τον μυστικό αριθμό. Υπάρχει κάτι που σας ενοχλεί; Κάτι που φαίνεται να μη δουλεύει σωστά;

Όταν ο παίκτης μαντέψει το μυστικό αριθμό το παιχνίδι συνεχίζεται, ενώ θα έπρεπε να σταματά.

10. Η εντολή **`break`** διακόπτει την επανάληψη μέσα στην οποία βρίσκεται *αμέσως* μόλις εκτελεστεί. *Προσθέστε* την **`break`** στο σημείο του προγράμματος που θεωρείτε κατάλληλο, έτσι ώστε το παιχνίδι να τερματίζεται *όταν ο παίκτης μαντέψει τον αριθμό.*

Πρέπει να ελέγξετε αν τοποθετήσατε την **`break`** στο σωστό σημείο. Εκτελέστε το πρόγραμμα και παίξτε το παιχνίδι δοκιμάζοντας αριθμούς. Διακόπτεται η επανάληψη όταν μαντέψετε τον μυστικό αριθμό;

*Η εντολή **`break`** πρέπει να προστεθεί μέσα στην **`if`**, στην περίπτωση που ο παίκτης μαντέψει σωστά το μυστικό αριθμό, αφού τότε πρέπει να τερματιστεί η επανάληψη.*

```
    if number == secret:
        print("Σωστά!")
        break
    else:
        print("Λάθος...")
```

11. Για δοκιμαστικούς λόγους, κάτω από την **break** *προσθέστε* την εντολή:

```
print("Ζντονκ!")
```

Εκτελέστε πάλι το πρόγραμμα μέχρι να μαντέψετε τον αριθμό. Εμφανίζεται το μήνυμα "Ζντονκ!"; Γιατί πιστεύετε ότι συμβαίνει αυτό;

Το μήνυμα δεν εμφανίζεται, αφού η εκτέλεση της εντολής **break** *προκαλεί την άμεση έξοδο από την επανάληψη. Επομένως, οι εντολές που την ακολουθούν δεν θα εκτελεστούν ποτέ.*

Αφαιρέστε τώρα την εντολή που προσθέσατε.

12. Τί θα αλλάζατε στο πρόγραμμα όπως έχει μέχρι στιγμής; Πώς πιστεύετε ότι πρέπει να επεκταθεί για να γίνει πιο ενδιαφέρον;

Προς το παρόν ο παίκτης μαντεύει στα τυφλά. Μια προσθήκη που θα τον βοηθούσε είναι να ενημερώνεται από το πρόγραμμα αν ο μυστικός αριθμός είναι μικρότερος ή μεγαλύτερος από αυτόν που έδωσε.

Επιλογές, Επιλογές

Θα επεκτείνουμε το πρόγραμμα έτσι ώστε να δίνει στο χρήστη περισσότερη πληροφορία: αντί να τον ενημερώνει απλά αν βρήκε τον μυστικό αριθμό ή όχι, θα τον κατευθύνει αν πρέπει να τον αναζητήσει ψηλότερα ή χαμηλότερα.

Για την επέκταση αυτή δεν αρκεί πια η απλή **if**–**else**, η οποία μπορεί να διακρίνει μόνο ανάμεσα σε δύο περιπτώσεις.

13. *Τροποποιήστε* την **if** που ελέγχει αν ο χρήστης βρήκε τον μυστικό αριθμό. Συμπληρώσετε την συνθήκη που λείπει:

```
    if secret == number:
        print("Σωστά!")
        break
    elif συνθήκη: # συμπληρώστε την συνθήκη
        print("Ο μυστικός αριθμός είναι μεγαλύτερος.")
    else:
        print("Ο μυστικός αριθμός είναι μικρότερος.")
```

Η συνθήκη που θα συμπληρώσουμε είναι:

```
elif secret > number:
```

ώστε το πρόγραμμα να ελέγχει αν ο μυστικός αριθμός είναι μεγαλύτερος από τον αριθμό που έδωσε ο χρήστης.

Εκτελέστε το πρόγραμμα και βεβαιωθείτε ότι λειτουργεί σωστά.

Σε ποια περίπτωση εκτελούνται οι εντολές της **else**; Γιατί πιστεύετε ότι δεν ελέγχουμε καμία συνθήκη σε αυτή την τρίτη περίπτωση;

Οι εντολές της **else** *εκτελούνται όταν ο μυστικός αριθμός είναι μικρότερος από τον αριθμό που έδωσε ο παίκτης. Δεν ελέγχουμε κάποια συνθήκη, αφού αυτή είναι η μοναδική, εναλλακτική περίπτωση.*

14. *Αναδιατάξτε* τις περιπτώσεις της **if** όπως φαίνεται παρακάτω. Και πάλι, θα πρέπει να συμπληρώσετε μια από τις συνθήκες που ελέγχονται. Είναι καλή εξάσκηση και θα σας κάνει να σκεφτείτε για την σειρά με την οποία ελέγχονται οι συνθήκες.

```
if συνθήκη: # συμπληρώστε την συνθήκη
    print("Ο μυστικός αριθμός είναι μικρότερος.")
elif secret > number:
    print("Ο μυστικός αριθμός είναι μεγαλύτερος.")
else:
    print("Σωστά!")
    break
```

Η συνθήκη που θα συμπληρώσουμε στην **if** *είναι:*

if `secret < number` :

Η συνθήκη που θα συμπληρώσουμε στην **elif** *είναι:*

elif `secret > number` :

Εκτελέστε το πρόγραμμα και βεβαιωθείτε ότι λειτουργεί σωστά.

Τί θα αλλάζατε στο πρόγραμμα όπως έχει μέχρι στιγμής; Πώς πιστεύετε ότι πρέπει να επεκταθεί για να γίνει πιο ενδιαφέρον;

Ο παίκτης έχει απεριόριστο αριθμό προσπαθειών, οπότε κάποια στιγμή θα μαντέψει το μυστικό αριθμό. Θα πρέπει να θέσουμε έναν μέγιστο αριθμό προσπαθειών για να τα καταφέρει.

Μέτρημα

Ο αριθμός των προσπαθειών που διαθέτει ο παίκτης δεν θα έπρεπε να είναι απεριόριστος. Θα επεκτείνουμε το παιχνίδι ώστε να τερματίζεται όταν εξαντληθούν οι προσπάθειες του παίκτη.

15. Αμέσως μετά τη **while**, δηλαδή στην αρχή της επανάληψης, *προσθέστε* την εντολή που ακολουθεί:

```
print("Απομένουν", tries, "προσπάθειες.")
```

Είναι εμφανές ότι η τιμή της μεταβλητής `tries` θα αντιστοιχεί στο πλήθος των προσπαθειών που απομένουν στον παίκτη.

Αν εκτελέσετε το πρόγραμμα όπως έχει θα εμφανιστεί μήνυμα λάθους, αφού επιχειρούμε να εμφανίσουμε την τιμή της `tries`, χωρίς πουθενά προηγουμένως να της έχουμε αποδώσει μια τιμή.

```
NameError: name 'tries' is not defined
```

16. Δώστε στην `tries` την αρχική τιμή 4. Αυτό θα είναι το πλήθος των προσπαθειών που διαθέτει ο χρήστης όταν ξεκινά το παιχνίδι.

    Η εντολή που θα προσθέσουμε στο πρόγραμμα είναι:

    ```
    tries = 4
    ```

    Τοποθετήσατε τις νέες εντολές *πριν* την επανάληψη ή *μέσα* σε αυτή; Για ποιο λόγο κάνατε αυτή την επιλογή;

    Η εντολή πρέπει να προστεθεί πριν από τη **`while`**, *ώστε η εκτέλεσή της να γίνει μόνο μια φορά. Διαφορετικά, οι προσπάθειες του παίκτη θα παίρνουν σε κάθε επανάληψη και πάλι την τιμή* `4` *αντί να μειώνονται.*

    Εκτελέστε το πρόγραμμα. Τί παρατηρείτε;

    Ο αριθμός των προσπαθειών δεν μειώνεται σε κάθε επανάληψη.

17. Η μεταβλητή `tries` θα πρέπει να μειώνεται σε κάθε γύρο του παιχνιδιού, δηλαδή *εντός της επανάληψης*. Μετά την `print` του βήματος 15 που εμφανίζει το πλήθος των προσπαθειών, *προσθέστε* τη γραμμή:

    ```
        tries = tries - 1
    ```

    Εκτελέστε το πρόγραμμα. Μειώνεται το πλήθος των προσπαθειών που απομένουν στον παίκτη;

    Ναι. Στην οθόνη εμφανίζεται σε κάθε γύρο ο νέος, μειωμένος αριθμός προσπαθειών.

    Αν το πλήθος των προσπαθειών δεν μειώνεται, βεβαιωθείτε ότι η εντολή που προσθέσατε στο βήμα 16 και δίνει αρχική τιμή στην `tries`, βρίσκεται *πριν* από την επανάληψη και όχι μέσα σε αυτή.

    Περιγράψτε πως ακριβώς πιστεύετε ότι λειτουργεί η εντολή που προσθέσατε για να μειώνεται η `tries`.

    Σε κάθε γύρο του παιχνιδιού ο αριθμός των προσπαθειών μειώνεται κατά `1`*. Αυτό επιτυγχάνεται ως εξής: υπολογίζεται η τιμή της παράστασης* `tries - 1` *και το αποτέλεσμα ονομάζεται πάλι* `tries`*. Η νέα τιμή της μεταβλητής* `tries` *υπολογίζεται με βάση την τρέχουσα τιμή της, την οποία και αντικαθιστά. Επομένως, θα πάρει διαδοχικά τις τιμές* `3`*,* `2`*,* `1` *και* `0`*, δεδομένου ότι ο παίκτης δεν θα μαντέψει το μυστικό αριθμό.*

    Υπάρχει κάτι που σας ενοχλεί και φαίνεται να μη δουλεύει σωστά;

    Το παιχνίδι δεν τερματίζει όταν οι προσπάθειες μηδενιστούν. Αντίθετα, συνεχίζει δίνοντας αρνητικές τιμές στη μεταβλητή `tries`*.*

Τερματισμός

Το παιχνίδι δεν πρέπει να τερματίζεται μόνο όταν ο παίκτης μαντέψει τον αριθμό, αλλά και όταν τελειώσουν οι προσπάθειές του.

18. Προς το παρόν, *προσθέστε* τις εντολές που ακολουθούν στο σημείο που κρίνετε κατάλληλο ώστε η επανάληψη να τερματίζεται όταν εξαντληθούν οι προσπάθειες του παίκτη.

```
    if tries == 0:
        break
```

Οι εντολές θα προστεθούν μέσα στην **while**, *μετά από όλες τις υπόλοιπες εντολές της επανάληψης.*

Εκτελέστε το πρόγραμμά σας. Φροντίστε, ως παίκτες, να εξαντλήσετε τις προσπάθειές σας χωρίς να μαντέψετε τον αριθμό.

Ποιες τιμές επιλέξατε να δοκιμάσετε, για να είστε βέβαιοι ότι δεν θα πετύχετε τον μυστικό αριθμό και θα εξαντλήσετε τις προσπάθειες;

Δεδομένου ότι ο μυστικός αριθμός είναι ανάμεσα στο `1` *και στο* `32`*, μια απλή λύση για να είμαστε βέβαιοι ότι δεν θα τον πετύχουμε είναι να δώσουμε τιμές εκτός αυτών των ορίων, για παράδειγμα το* `40`*.*

Τερματίζεται το πρόγραμμα όταν εξαντληθούν οι προσπάθειες του παίκτη;

Ναι το πρόγραμμα τερματίζει κανονικά δίνοντας στον παίκτη όλες τις προσπάθειες που δικαιούται.

Είστε βέβαιοι ότι το πρόγραμμα επιτρέπει στον παίκτη να χρησιμοποιήσει *όλες* του τις προσπάθειες; Αν προσθέσατε τις εντολές αμέσως μετά την εντολή `tries = tries - 1`, τότε το πρόγραμμά σας θα στερεί από τον παίκτη μια προσπάθεια!

19. *Διαγράψτε* τις εντολές που προσθέσατε στο προηγούμενο βήμα. Αντί για την **break**, θα διερευνήσουμε έναν εναλλακτικό (και συχνά προτιμότερο) τρόπο για να τερματίζουμε την επανάληψη όταν εξαντληθούν οι προσπάθειες.

    Η **while** συνοδεύεται από μια *συνθήκη*. Στην *αρχή* κάθε κύκλου, η συνθήκη αυτή ελέγχεται εκ νέου. Αν η συνθήκη είναι αληθής τότε η επανάληψη συνεχίζεται για άλλον έναν κύκλο.

    Εμείς χρησιμοποιήσαμε μέχρι τώρα την τετριμμένη συνθήκη **True**, η οποία είναι πάντα αληθής, γι' αυτό και η επανάληψη δεν διακόπτονταν λόγω της συνθήκης.

    Αντικαταστήστε την συνθήκη **True** με την συνθήκη `tries > 0`, που είναι αληθής μόνο όταν απομένουν κι άλλες προσπάθειες στον παίκτη. Σε περίπτωση που αυτό δεν ισχύει, η επανάληψη θα διακοπεί.

    ```
    while tries > 0:
    ```

    Εκτελέστε και πάλι το πρόγραμμα και διερευνήστε την συμπεριφορά του. Λειτουργεί σωστά ή εντοπίζετε προβλήματα;

    Ναι, λειτουργεί σωστά. Παρατηρούμε ότι όταν ο παίκτης δεν μαντέψει τον αριθμό το πολύ σε τέσσερεις προσπάθειες, τότε το παιχνίδι σταματά.

20. Για δοκιμαστικούς λόγους, *προσθέστε* αμέσως κάτω από την εντολή `tries = tries - 1` τη γραμμή:

    ```
    print("Ζντονκ!", tries)
    ```

Εκτελέστε το πρόγραμμα μέχρι να εξαντληθούν οι προσπάθειες. Εμφανίζεται στο τέλος το μήνυμα `"Ζντονκ! 0"`;

Ναι, εμφανίζεται.

Ενώ η συνθήκη της `while` είναι `tries > 0`, από το μήνυμα φαίνεται ότι η εκτέλεση των εντολών της επανάληψης δεν διακόπτεται *άμεσα* όταν μηδενιστεί η μεταβλητή `tries` και η συνθήκη πάψει να ισχύει. Άρα η συνθήκη `tries > 0` της `while` δεν ελέγχεται συνεχώς αλλά μόνο στην αρχή κάθε νέου κύκλου της επανάληψης.

Αφαιρέστε τώρα την εντολή που προσθέσατε.

21. Αν ο παίκτης εξαντλήσει τις προσπάθειές του και δεν καταφέρει να βρει τον αριθμό τότε χάνει και το παιχνίδι σταματά. *Προσθέστε* τις κατάλληλες εντολές στο πρόγραμμα έτσι ώστε, στην περίπτωση αυτή, να εμφανίζει στον παίκτη τον αριθμό που αναζητούσε.

Θα χρειαστεί, μεταξύ άλλων, να επανεισάγετε την εντολή που αφαιρέσατε στο βήμα 9, η οποία εμφάνιζε τον μυστικό αριθμό.

```
Ο μυστικός αριθμός ήταν ο 13.
```

Φροντίστε να εμφανίζεται το μήνυμα *μόνο όταν είναι απαραίτητο*. Αν ο παίκτης βρει τον μυστικό αριθμό, τότε το μήνυμα δε χρειάζεται.

Στο σημείο αυτό μπορούμε να σκεφτούμε δύο πιθανές εκδοχές. Η μια είναι να εξετάσουμε ότι όταν τερματιστεί η επανάληψη οι προσπάθειες έχουν μηδενιστεί και η άλλη είναι να εξετάσουμε ότι μετά τον τερματισμό της επανάληψης ο αριθμός που έδωσε ο παίκτης δεν είναι ίδιος με το μυστικό. Παρόλο που και οι δύο μοιάζουν σωστές, μόνο η δεύτερη λειτουργεί με τον τρόπο που θέλουμε. Επομένως, μετά το τέλος των εντολών της επανάληψης, προσθέτουμε:

```
if number != secret:
    print("Ο μυστικός αριθμός ήταν ο", secret)
```

Τοποθετήσατε τις νέες εντολές *μέσα* στην επανάληψη ή *μετά* από αυτή; Για ποιο λόγο κάνατε αυτή την επιλογή;

Οι εντολές τοποθετήθηκαν μετά την επανάληψη, ώστε το πρόγραμμα να ελέγχει μόνο μια φορά, στο τέλος του παιχνιδιού αν θα "αποκαλύψει" το μυστικό αριθμό.

Τι διαφορά θα υπήρχε αν είχατε κάνει την αντίθετη επιλογή;

Μετά την πρώτη αποτυχημένη προσπάθεια του παίκτη να μαντέψει το μυστικό αριθμό, το πρόγραμμα θα του τον εμφάνιζε στην οθόνη. Επομένως, στη δεύτερη προσπάθεια δεν θα δυσκολευόταν και πολύ να "μαντέψει" σωστά!

Εκτελέστε το πρόγραμμα. Λειτουργεί σωστά στην περίπτωση που ο παίκτης μαντέψει τον αριθμό; Μήπως στο τέλος του παιχνιδιού του εμφανίζει τον μυστικό αριθμό, παρόλο που τον έχει βρει;

Ναι, λειτουργεί σωστά. Ο μυστικός αριθμός εμφανίζεται μόνο στην περίπτωση που ο παίκτης αποτύχει.

Σε αυτή την περίπτωση, *διορθώστε* το πρόγραμμα.

Το πρόγραμμα λειτουργεί σωστά στην περίπτωση που ο παίκτης μαντεύει τον αριθμό *στην τελευταία του προσπάθεια*; Μήπως του εμφανίζει και πάλι τον μυστικό αριθμό, παρόλο που τον έχει βρει;

Ναι, λειτουργεί, αφού οι εντολές της `if` *δεν μπορούν να εκτελεστούν όταν ο παίκτης έχει μαντέψει το μυστικό αριθμό.*

Αν η απάντηση ήταν καταφατική, μάλλον προσπαθείτε να διαπιστώσετε αν ο παίκτης έχασε ελέγχοντας την συνθήκη `tries == 0`. Σκεφτείτε όμως: αν ο παίκτης μαντέψει τον αριθμό στην τελευταία του προσπάθεια τότε θα ισχύει ότι `tries == 0` όμως ο παίκτης δεν θα έχει χάσει. Χρειάζεται να διορθώσετε το πρόγραμμα ελέγχοντας με *διαφορετική συνθήκη* αν ο παίκτης απέτυχε να μαντέψει τον αριθμό.

Περισσότερη Βοήθεια

22. Εκτελέστε το πρόγραμμα. Στην πρώτη σας προσπάθεια δοκιμάστε τον αριθμό `13`. Ο μυστικός αριθμός είναι μικρότερος ή μεγαλύτερος;

Αν ο μυστικός αριθμός είναι το `13`, απλά εκτελέστε και πάλι το πρόγραμμα.

Δοκιμάζουμε τον αριθμό `13` *και το πρόγραμμα εμφανίζει μήνυμα ότι ο μυστικός αριθμός είναι μεγαλύτερος από* `13`.

Σε ποιο διάστημα θ' αναζητήσετε τώρα τον μυστικό αριθμό, δηλαδή ποια είναι η ελάχιστη και ποια η μέγιστη δυνατή τιμή που γνωρίζετε τώρα ότι μπορεί να έχει ο μυστικός αριθμός;

Η μικρότερη δυνατή τιμή του μυστικού αριθμού είναι το `14`*, αφού ο μυστικός αριθμός είναι μεγαλύτερος του* `13`*, σύμφωνα με το προηγούμενο, ενώ η μεγαλύτερη δυνατή τιμή* παραμένει *το* `32`.

23. Στη δεύτερη προσπάθεια, δοκιμάστε έναν αριθμό που ανήκει στο διάστημα που απαντήσατε προηγουμένως.

Συνεχίστε μέχρι να τελειώσει το παιχνίδι, συμπληρώνοντας τον πίνακα που ακολουθεί. Σημειώστε σε κάθε βήμα τον αριθμό που δοκιμάσατε, την απάντηση του προγράμματος και το διάστημα μέσα στο οποίο "εγκλωβίσατε" κάθε φορά τον μυστικό αριθμό. Το διάστημα αυτό ορίζεται από την ελάχιστη (`low`) και τη μέγιστη (`high`) δυνατή τιμή που έχει νόημα να δοκιμάσετε μετά από κάθε προσπάθεια.

αριθμός `number`	ο μυστικός είναι (μικρότερος / μεγαλύτερος)	ελάχιστη `low`	μέγιστη `high`
`13`	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....

Έστω ότι ο μυστικός αριθμός είναι το 15. Ακολουθεί ένα παράδειγμα συμπλήρωσης του πίνακα:

αριθμός `number`	*ο μυστικός είναι* *(μικρότερος / μεγαλύτερος)*	*ελάχιστη* `low`	*μέγιστη* `high`
13	μεγαλύτερος	14	32
23	μικρότερος	14	22
18	μικρότερος	14	17
15	.....	.....	.....

Θεωρείτε "δίκαιο" να δίνονται στον παίκτη τέσσερις προσπάθειες και γιατί;

Δεδομένου ότι ο μυστικός αριθμός βρίσκεται ανάμεσα στο `1` *και στο* `32` *ο παίκτης μπορεί να τον βρει στις* `5` *προσπάθειες, εφόσον ακολουθήσει την κατάλληλη στρατηγική. Δίνοντάς του* `4` *ευκαιρίες τον "αναγκάζουμε" να επιλέξει τυχαία την τελευταία φορά.*

24. Τώρα θα επεκτείνουμε το πρόγραμμα έτσι ώστε να βοηθάει το χρήστη ακόμα περισσότερο. Θα χρησιμοποιήσουμε δύο μεταβλητές `low` και `high`, οι οποίες αντιστοιχούν στην ελάχιστη και τη μέγιστη δυνατή τιμή που γνωρίζουμε ότι μπορεί να έχει ο μυστικός αριθμός.

    Στην αρχή του προγράμματος, αποδώστε αρχικές τιμές σε αυτές τις μεταβλητές:

    ```
    low = 1
    high = 32
    ```

25. Αμέσως πριν από την `input()` με την οποία το πρόγραμμα διαβάζει από το χρήστη έναν αριθμό, *προσθέστε* μια εντολή που εμφανίζει στο χρήστη τα `low` και `high`, για να τον βοηθά στην επιλογή του. Για παράδειγμα, αν τα `low` και `high` είναι αντίστοιχα `14` και `23`, τότε να εμφανίζει:

    ```
    Δοκίμασε ανάμεσα στο 14 και το 23.
    ```

    Η εντολή που θα προσθέσουμε στο πρόγραμμα είναι:

    ```
    print("Δοκίμασε ανάμεσα στο", low, "και το", high)
    ```

    Εκτελέστε το πρόγραμμα και παρατηρήστε τις τιμές των `low` και `high`, καθώς προσπαθείτε να μαντέψετε τον μυστικό αριθμό. Υπάρχει κάτι που σας ενοχλεί και φαίνεται να μη δουλεύει σωστά;

    Ναι, το πρόγραμμα εμφανίζει σε κάθε επανάληψη το μήνυμα:

    ```
    Δοκίμασε ανάμεσα στο 1 και στο 32 ,
    ```

    δηλαδή δεν μεταβάλλει κατάλληλα τις τιμές των `low` *και* `high`.

26. Στο πρόγραμμα *υπάρχει ήδη* μια **`if`** που ελέγχει αν ο μυστικός αριθμός είναι μικρότερος από τον αριθμό του χρήστη:

```
    if secret < number:
        print("Ο μυστικός αριθμός είναι μικρότερος.")
```

Στην περίπτωση αυτή, όπως φαίνεται κι από τον πίνακα που συμπληρώσατε στο βήμα 23, *μόνο μία* από τις μεταβλητές `low` και `high` χρειάζεται ν' αλλάξει τιμή. Ποιά από τις δύο;

Στην περίπτωση που ο μυστικός αριθμός είναι μικρότερος πρέπει να αλλάξουμε την τιμή της μεταβλητής `high`.

Να *προσθέσετε* σε αυτή την περίπτωση της `if` μια εντολή που μεταβάλλει κατάλληλα την τιμή της `low` ή της `high`. Αν δυσκολευτείτε, ανατρέξτε στον πίνακα που συμπληρώσατε στο βήμα 23.

Στην περίπτωση που ο μυστικός αριθμός είναι μικρότερος πρέπει να αλλάξουμε την τιμή της μεταβλητής `high`*, όπως παρακάτω:*

```
    if secret < number:
        print("Ο μυστικός αριθμός είναι μικρότερος.")
        high = number - 1
```

Αφού ο μυστικός αριθμός είναι μικρότερος από τον αριθμό που έδωσε ο παίκτης τότε η μεγαλύτερη δυνατή τιμή που μπορεί να έχει, δηλαδή η τιμή της `high`*, είναι τουλάχιστον μικρότερη κατά 1 από τον αριθμό αυτό.*

Στο πρόγραμμα υπάρχει επίσης μια `if` που ελέγχει αν ο μυστικός αριθμός είναι μεγαλύτερος από τον αριθμό του χρήστη:

```
    elif secret > number:
        print("Ο μυστικός αριθμός είναι μεγαλύτερος.")
```

Να *προσθέσετε* και σε αυτή την περίπτωση την εντολή που μεταβάλλει κατάλληλα την τιμή της `low` ή της `high`.

Στην περίπτωση που ο μυστικός αριθμός είναι μεγαλύτερος πρέπει να αλλάξουμε την τιμή της μεταβλητής `low`*, όπως παρακάτω:*

```
    elif secret > number:
        print("Ο μυστικός αριθμός είναι μεγαλύτερος.")
        low = number + 1
```

Αφού ο μυστικός αριθμός είναι μεγαλύτερος από τον αριθμό που έδωσε ο παίκτης τότε η μικρότερη δυνατή τιμή που μπορεί να έχει, δηλαδή η τιμή της `low`*, είναι τουλάχιστον μεγαλύτερη κατά 1 από τον αριθμό αυτό.*

27. Εκτελέστε το πρόγραμμά σας αρκετές φορές και ελέγξτε το διεξοδικά, για να διαπιστώσετε αν οι τιμές των μεταβλητών `low` και `high` μεταβάλλονται σωστά.

    Λειτουργεί σωστά το πρόγραμμα; Υπάρχει κάποια περίπτωση στην οποία η συμπεριφορά του είναι προβληματική;

Στην περίπτωση που ο παίκτης δώσει τιμές εκτός των ορίων `low` *και* `high`*, το πρόγραμμα δε λειτουργεί σωστά.*

28. Εκτελέστε ακόμα μερικές φορές το πρόγραμμα. Δοκιμάστε, ως παίκτης, να δίνετε τιμές που είναι *εκτός των ορίων* που προτείνει το πρόγραμμα. Τί παρατηρείτε; Δημιουργείται πρόβλημα;

Ναι, δημιουργείται. Το πρόγραμμα αναπροσαρμόζει τις τιμές των ορίων με λάθος τρόπο. Για παράδειγμα, αν στην πρώτη προσπάθεια ο παίκτης δώσει τον αριθμό `40` *τότε η τιμή της* `high` *θα γίνει* `39`*.*

Αν υπάρχει πρόβλημα όταν ο χρήστης δίνει τιμές εκτός του διαστήματος που ορίζουν οι `low` και `high`, προσθέστε τους κατάλληλους ελέγχους στο πρόγραμμα ώστε να διορθωθούν τυχόν σφάλματα. *Οι έλεγχοι που θα προσθέσουμε στο πρόγραμμα είναι:*

```
if secret < number:
    print("Ο μυστικός αριθμός είναι μικρότερος.")
    if number <= high:
        high = number - 1
elif secret > number:
    print("Ο μυστικός αριθμός είναι μεγαλύτερος.")
    if number >= low:
        low = number + 1
```

Gimme a Break

Η χρήση της **`break`** είναι μια πρακτική που δεν ακολουθείται από όλους. Ορισμένοι θεωρούν ότι ο κώδικας είναι πιο κατανοητός όταν υπάρχει ένα μοναδικό σημείο εξόδου από την επανάληψη: η συνθήκη συνέχειας.

29. Μπορούμε να χρησιμοποιήσουμε μια *λογική μεταβλητή*, την οποία θα ονομάσουμε `found`, για να "θυμάται" το πρόγραμμά μας αν ο χρήστης βρήκε τον μυστικό αριθμό.

Προσθέστε στην αρχή του προγράμματός σας την εντολή:

```
found = False    # η Found είναι ψευδής
```

Υπάρχουν μόνο δύο *λογικές* τιμές: **`True`** και **`False`**.

Για διαγνωστικούς λόγους, *προσθέστε προσωρινά* στο τέλος της επανάληψης (*μέσα* σε αυτή, όχι μετά) την εντολή:

```
print("Ζντονκ!", found)
```

Εκτελέστε το πρόγραμμά σας. Θα πρέπει *σε κάθε επανάληψη* να βλέπετε το μήνυμα `"Ζντονκ! False"`

Ναι εμφανίζεται το μήνυμα σε κάθε επανάληψη.

30. *Προσθέστε* την εντολή που ακολουθεί στο τμήμα του προγράμματος που εκτελείται *μόνο* όταν ο χρήστης εντοπίσει τον μυστικό αριθμό.

```
found = True    # η Found γίνεται αληθής
```

Θα προσθέσουμε την εντολή μέσα στην περίπτωση που ο παίκτης μαντέψει σωστά το μυστικό αριθμό, όπως παρακάτω:

```
    if secret < number:
        print("Ο μυστικός αριθμός είναι μικρότερος.")
    elif secret > number:
        print("Ο μυστικός αριθμός είναι μεγαλύτερος.")
    else:
        print("Σωστά!")
        found = True
        break
```

Εκτελέστε το πρόγραμμα. Πότε εμφανίζεται το `"Ζντονκ! True"` και γιατί;

Το μήνυμα δεν εμφανίζεται, διότι μόλις ο παίκτης μαντέψει σωστά το μυστικό αριθμό, η επανάληψη διακόπτεται εξαιτίας της εντολής **break** *που ακολουθεί.*

31. *Αφαιρέστε* την **break** από το πρόγραμμα. Ποιο αποτέλεσμα *πιστεύετε* ότι θα έχει αυτό;

Η επανάληψη δεν θα τερματίζεται όταν ο παίκτης μαντέψει σωστά το μυστικό αριθμό. Προκειμένου να τερματιστεί η επανάληψη θα πρέπει υποχρεωτικά να εξαντληθούν όλες οι προσπάθειες.

Προσθέστε στη συνθήκη της **while** μια επιπλέον συνθήκη που θα ελέγχει την τιμή της `found` για να διαπιστώνει αν η επανάληψη θα πρέπει να συνεχιστεί.

Για να συνδυάσετε αυτή την συνθήκη με την ήδη υπάρχουσα συνθήκη `tries > 0` χρησιμοποιήστε ανάμεσά τους το **and** που έχει σαν αποτέλεσμα την *σύζευξη* των συνθηκών.

Η συνθήκη της **while** *θα είναι όπως παρακάτω:*

```
while tries > 0 and not found:
```

*Ο λογικός τελεστής της άρνησης (***not***) αντιστρέφει την τιμή μιας συνθήκης. Όταν η συνθήκη είναι ψευδής την κανει αληθή και αντίστροφα. Έτσι, όσο η* `found` *διατηρεί την τιμή* **False***, η πρόταση* **not** `found` *παραμένει αληθής και η επανάληψη συνεχίζεται (εφόσον οι προσπάθειες παραμένουν μεγαλύτερες από το* `0`*). Αν ο παίκτης βρει τον μυστικό αριθμό και η* `found` *γίνει* **True***, η πρόταση* **not** `found` *θα έχει την τιμή* **False** *και η επανάληψη θα τερματιστεί.*

Εκτελέστε το πρόγραμμα και διερευνήστε την συμπεριφορά του. Λειτουργεί σωστά ή εντοπίζετε προβλήματα;

Ναι, το πρόγραμμα λειτουργεί σωστά. Η επανάληψη τερματίζει πλέον και στην περίπτωση που ο παίκτης μαντέψει σωστά το μυστικό αριθμό.

Πότε εμφανίζεται το `"Ζντονκ! True"` και γιατί;

Το μήνυμα εμφανίζεται όταν ο παίκτης μαντέψει σωστά το μυστικό αριθμό.

Αφαιρέστε το διαγνωστικό μήνυμα από την επανάληψη.

32. *Αντικαταστήστε* τη συνθήκη `number != secret` στο τέλος του παιχνιδιού με μια συνθήκη που ελέγχει την τιμή της `found`.

Η συνθήκη της **`if`** *θα είναι:*

```
if not found:
```

33. Προηγουμένως, το παιχνίδι έληγε όταν ο παίκτης έβρισκε τον αριθμό με τη χρήση της **`break`**. Τώρα η **`break`** αφαιρέθηκε και ο τερματισμός της επανάληψης γίνεται αποκλειστικά όταν η συνθήκη συνέχειας της **`while`** είναι ψευδής. Ποια πιστεύετε ότι είναι η διαφορά ανάμεσα στις δύο περιπτώσεις;

Η εκτέλεση της εντολής **`break`** *έχει ως αποτέλεσμα την άμεση διακοπή της επανάληψης. Αντίθετα, στη δεύτερη περίπτωση, στην οποία εξετάζεται τη συνθήκη της* **`while`***, ολοκληρώνεται η εκτέλεση όλων των εντολών της επανάληψης και ο τερματισμός της επιτυγχάνεται μόνο όταν ελεγχθεί ξανά η συνθήκη της* **`while`** *και διαπιστωθεί ότι πλέον έχει γίνει ψευδής.*

Εξαρτήματα

34. Και τώρα, ας *ξεχάσουμε* για λίγο το πρόγραμμα που έχουμε αναπτύξει. Διαβάστε προσεκτικά τον κώδικα που ακολουθεί. Μην τον πληκτρολογήσετε, απλά εξετάστε τον.

```
print("Μάντεψε τον αριθμό.")
print("Δοκίμασε ανάμεσα στο", a, "και το", b)
c = int(input())
print("Έδωσες τον αριθμό", c)
```

Ποιες μεταβλητές χρειάζεται να έχουν *ήδη* τιμή για να λειτουργήσει αυτό το τμήμα κώδικα;

Για να λειτουργήσει ο κώδικας θα πρέπει να έχει δοθεί τιμή στις μεταβλητές `a` *και* `b` *που χρησιμοποιούνται στην εμφάνιση του μηνύματος.*

Σε ποιες μεταβλητές αποδίδεται τιμή σε αυτό το τμήμα κώδικα;

Στη μεταβλητή `c` *αποδίδεται η τιμή που επιστρέφεται από την* `input()`*. Αυτός είναι και ο λόγος που η μεταβλητή* `c` *δεν χρειάζεται να έχει τιμή εκ των προτέρων, όπως ισχύει για τις* `a` *και* `b`*.*

Περιγράψτε, όσο καλύτερα μπορείτε, τη λειτουργία αυτού του μικρού τμήματος κώδικα. Τί θα λέγατε ότι κάνει;

Ο κώδικας προτρέπει τον χρήστη να μαντέψει έναν αριθμό ανάμεσα στις τιμές των `a` *και* `b`*. Στη συνέχεια ζητάει την απάντησή του και την αποθηκεύει στη μεταβλητή* `c`*, την τιμή της οποίας εμφανίζει αμέσως μετά στην οθόνη.*

Υπάρχει κάποιο τμήμα κώδικα στο πρόγραμμά σας που να εκτελεί αυτή τη συγκεκριμένη λειτουργία;

Ναι, το τμήμα του προγράμματος που ζητάει από τον παίκτη να μαντέψει

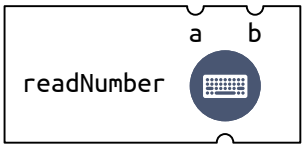
το μυστικό αριθμό, εμφανίζοντας του παράλληλα τα όρια μέσα στο οποία βρίσκεται.

35. Θα διαπιστώσατε ότι το πρόγραμμα περιλαμβάνει ένα τέτοιο τμήμα κώδικα, το οποίο χρησιμοποιείται για να εισαχθεί από το χρήστη μια πιθανή τιμή για τον μυστικό αριθμό.

    Θα περικλείσουμε αυτόν τον κώδικα σε μια *συνάρτηση*, ένα *υποπρόγραμμα* που επιτελεί μια συγκεκριμένη λειτουργία.

    Προσθέστε στην αρχή του προγράμματος τα εξής:

```
def readNumber(a,b):
    print("Μάντεψε τον αριθμό.")
    print("Δοκίμασε ανάμεσα στο", a, "και το", b)
    c = int(input())
    print("Έδωσες τον αριθμό", c)
    return c
```



Εικόνα: Η συνάρτηση `readNumber` είναι ένα ανεξάρτητο τμήμα κώδικα, ένα αυτόνομο "εξάρτημα" που επιτελεί μια συγκεκριμένη λειτουργία.

    Με τον τρόπο αυτό *ορίζεται* η συνάρτηση `readNumber`, η οποία λειτουργεί ως εξής: Δέχεται δύο παραμέτρους, δύο τιμές που τις ονομάζει `a` και `b` για να μπορεί να αναφέρεται σε αυτές. Προτρέπει το χρήστη να πληκτρολογήσει έναν αριθμό μεταξύ των `a` και `b` και *επιστρέφει* με την εντολή `return` την ακέραια τιμή που πληκτρολογεί ο χρήστης.

    Οι μεταβλητές `a`, `b` και `c` έχουν *τοπική εμβέλεια*, δηλαδή δεν είναι "ορατές" από το υπόλοιπο πρόγραμμα. Δημιουργούνται εκ νέου κάθε φορά που εκτελείται η συνάρτηση `readNumber` και μετά παύουν να υφίστανται.

    Γιατί φτιάξαμε αυτό το μικρό "εξάρτημα"; Πρώτον, για να το χρησιμοποιήσουμε στο πρόγραμμά μας. Δεύτερον, για να μπορέσουμε αργότερα να το αντικαταστήσουμε με παρόμοια εξαρτήματα που κάνουν την ίδια δουλειά (επιστρέφουν μια πιθανή τιμή για τον μυστικό αριθμό) με διαφορετικό τρόπο.

36. Κατασκευάσαμε ένα "εξάρτημα" που ονομάζεται `readNumber` και επιτελεί μια συγκεκριμένη λειτουργία, ωστόσο δεν το έχουμε ακόμα *χρησιμοποιήσει*.

    Εντοπίστε στο πρόγραμμά σας τις παρακάτω εντολές:

```
print("Μάντεψε τον αριθμό.")
print("Δοκίμασε ανάμεσα στο", low, "και το", high)
number = int(input())
print("Έδωσες τον αριθμό", number)
```

    Διαγράψτε τις και *αντικαταστήστε* με την παρακάτω γραμμή:

```
number = readNumber(low, high)
```

    Κατά την εκτέλεση του προγράμματος, αυτή η γραμμή θα *καλέσει* την `readNumber`, θα ενεργοποιήσει την εκτέλεση των εντολών της, παρέχοντας ως παραμέτρους τις τιμές των `low` και `high`. Την τιμή που επιστρέφεται από την συνάρτηση την ονομάζουμε `number`.

Εκτελέστε το πρόγραμμα. Παρατηρείτε κάποια διαφορά στον τρόπο που λειτουργεί;

Όχι. Ως χρήστες δεν αντιλαμβανόμαστε καμία απολύτως διαφορά στα μηνύματα που βλέπουμε και στον τρόπο που αλληλεπιδρούμε με το πρόγραμμα. Ως προγραμματιστές όμως, παρατηρούμε ότι το κύριο πρόγραμμα είναι συντομότερο και φαίνεται απλούστερο στην κατανόηση της λειτουργίας του.

37. *Προσθέστε* στην αρχή του προγράμματός σας τον κώδικα που ακολουθεί. *Συμπληρώστε* κατάλληλα τις γραμμές που συνοδεύονται από σχόλιο, έτσι ώστε η συνάρτηση `randNumber` που ορίζεται εδώ *να επιστρέφει έναν τυχαίο αριθμό* ανάμεσα στα `a` και `b`.

```
import time
def randNumber(a,b):
    print("Μάντεψε τον αριθμό.")
    print("Δοκίμασε ανάμεσα στο", a, "και το", b)
    c = επιλογή τυχαίου αριθμού              # συμπληρώστε
    print("Το πρόγραμμα επιλέγει", c)
    time.sleep(3)
    return επιστροφή τιμής                   # συμπληρώστε
```

Το τμήμα κώδικα θα συμπληρωθεί όπως παρακάτω:

```
import time
def randNumber(a,b):
    print("Μάντεψε τον αριθμό.")
    print("Δοκίμασε ανάμεσα στο", a, "και το", b)
    c = random.randint(a,b)
    print("Το πρόγραμμα επιλέγει", c)
    time.sleep(3)
    return c
```

Στο κύριο πρόγραμμα, εντοπίστε την εντολή με την οποία η μεταβλητή `number` παίρνει τιμή από την συνάρτηση `readNumber` και *προσθέστε* μπροστά από την εντολή το σύμβολο #. Αυτό έχει σαν αποτέλεσμα η εντολή αυτή να θεωρείται *σχόλιο* και να αγνοείται κατά την εκτέλεση του προγράμματος.

Είναι συνηθισμένο να "σχολιάζουμε" μια εντολή όταν θέλουμε να παρακαμφθεί, χωρίς όμως να την αφαιρέσουμε πλήρως από το πρόγραμμα. Έτσι μπορούμε να την επαναφέρουμε αργότερα, αν αυτό είναι απαραίτητο.

Στο ίδιο σημείο, *προσθέστε* μια εντολή με την οποία η μεταβλητή `number` θα παίρνει τιμή από την συνάρτηση `randNumber`, με παραμέτρους τις τιμές των `low` και `high`. Αν δυσκολευτείτε, ανατρέξτε στο βήμα 37, όπου γίνεται κάτι ανάλογο.

Η εντολή που θα προσθέσουμε στο πρόγραμμα (με την προηγούμενη εντολή σχολιασμένη) είναι:

```
# number = readNumber(low,high)
number = randNumber(low,high)
```

Ουσιαστικά, εκεί που το πρόγραμμα χρησιμοποιούσε τη `readNumber` για να αποκτηθεί μια πιθανή τιμή για τον μυστικό αριθμό, τώρα αυτή αντικαταστάθηκε από την `randNumber`.

Χρησιμοποιείται πουθενά στο πρόγραμμα η `input`, για να παρέχει οποιαδήποτε είσοδο ο χρήστης; Αν όχι, ποιος θα επιλέγει τώρα αριθμούς, προσπαθώντας να μαντέψει τον τυχαίο αριθμό;

Όχι, ο χρήστης δεν δίνει πλέον τιμές. Στην ουσία το ρόλο του παίκτη τον έχει αναλάβει το ίδιο το πρόγραμμα, το οποίο παράγει τυχαίους αριθμούς προσπαθώντας να μαντέψει το μυστικό αριθμό.

Εκτελέστε το πρόγραμμα. Τι αλλαγή επιφέρει η χρήση της συνάρτησης `randNumber`, αντί για τη `readNumber`;

Ο ρόλος του παίκτη ανατίθεται πλέον στο ίδιο το πρόγραμμα.

38. Όταν ξεκινά το πρόγραμμα και γνωρίζετε ότι ο μυστικός αριθμός βρίσκεται κάπου ανάμεσα στο `1` και το `32`, ποια τιμή σας φαίνεται προτιμότερο να επιλέξετε;

Η καλύτερη τιμή είναι το `16`*. Χωρίζοντας το διάστημα των αριθμών στη μέση θα "διώξουμε" με μια κίνηση τους μισούς αριθμούς.*

Αν γνωρίζετε ότι ο μυστικός αριθμός βρίσκεται κάπου ανάμεσα στο `13` και το `23`, ποια τιμή σας φαίνεται προτιμότερο να επιλέξετε;

Ακολουθώντας την ίδια λογική θα επιλέξουμε την τιμή `18`*.*

Αν γνωρίζετε ότι ο μυστικός αριθμός βρίσκεται κάπου ανάμεσα στο `low` και το `high`, ποια τιμή σας φαίνεται προτιμότερο να επιλέξετε και γιατί;

Γενικεύοντας το σκεπτικό των προηγούμενων βημάτων θα επιλέγαμε την τιμή που βρίσκεται στη μέση του διαστήματος (`low`*,* `high`*), υπολογίζοντας την τιμή της παράστασης* `(low + high) // 2`*.*

39. Κατασκευάστε μια συνάρτηση `midNumber`. Η συνάρτηση θα δέχεται σαν παραμέτρους δύο τιμές, που θ' αντιστοιχούν στα όρια του διαστήματος μέσα στο οποίο γνωρίζουμε ότι βρίσκεται ο μυστικός αριθμός, και θα επιστρέφει την τιμή που βρίσκεται *στο μέσο αυτού του διαστήματος*.

Ο μυστικός αριθμός είναι ακέραιος, άρα η τιμή που επιλέγει η `midNumber` πρέπει να είναι επίσης ακέραια. Φροντίστε να χρησιμοποιήσετε τον τελεστή `//` της *ακέραιας* διαίρεσης για τον υπολογισμό του μέσου του διαστήματος.

Χρησιμοποιήστε ως πρότυπα τις `readNumber` και `randNumber`.

Η συνάρτηση `midNumber` *θα είναι όπως παρακάτω:*

```
def midNumber(a,b):
    print("Μάντεψε τον αριθμό:", a , "-", b)
    num = (a + b) // 2
    print("Ο υπολογιστής επιλέγει:", num)
    return num
```

Στο κύριο πρόγραμμα, εντοπίστε την εντολή με την οποία η μεταβλητή `number` παίρνει τιμή από την συνάρτηση `randNumber` και *προσθέστε* μπροστά από την εντολή το σύμβολο #.

Στο ίδιο σημείο, *προσθέστε* μια εντολή με την οποία η μεταβλητή `number` θα παίρνει τιμή από την συνάρτηση `midNumber`, με παραμέτρους τις τιμές των `low` και `high`.

Η εντολή που θα προσθέσουμε για να καλέσουμε την `midNumber` *είναι:*

```
number = midNumber(low,high)
```

Εκτελέστε το πρόγραμμα μερικές φορές. Καταφέρνει να εντοπίσει τον μυστικό αριθμό;

Ναι, το πρόγραμμα κατορθώνει κάποιες φορές να μαντέψει το μυστικό αριθμό. Τις υπόλοιπες πέφτει πολύ κοντά! Αν δοκιμάσουμε να του δώσουμε μια ακόμα προσπάθεια, δηλαδή `5` *συνολικά τότε θα καταφέρνει να τον μαντεύει σε όλες τις περιπτώσεις.*