# Μπαρμπούτι

## Ενδεικτικές Απαντήσεις Φύλλου Εργασίας

2

22 Ιουλίου 2016

13:51

## Τυχαιότητα

Στο παιχνίδι που θα υλοποιήσουμε, ο παίκτης ρίχνει δύο ζάρια. Το αποτέλεσμα του παιχνιδιού εξαρτάται από το *άθροισμα* των ενδείξεων των δύο ζαριών: για κάποιες τιμές του αθροίσματος ο παίκτης κερδίζει, για κάποιες άλλες χάνει, ενώ υπάρχουν και περιπτώσεις που το αποτέλεσμα δεν κρίνεται από την πρώτη ζαριά.

Στην πραγματικότητα, κάθε φορά που ρίχνουμε ένα ζάρι παράγουμε έναν *τυχαίο* αριθμό από το `1` μέχρι και το `6`. Οι βασικές εντολές της Python δεν μας παρέχουν κάποιο μηχανισμό για να μιμηθούμε αυτή τη διαδικασία. Για το σκοπό αυτό θα χρειαστεί να εισάγουμε στο πρόγραμμά μας μια από τις πολλές διαθέσιμες *βιβλιοθήκες*, τη `random`, η οποία παρέχει τη λειτουργικότητα που μας χρειάζεται.

Οι βιβλιοθήκες είναι συλλογές από έτοιμα μικρά προγράμματα που μπορούμε να χρησιμοποιήσουμε στα προγράμματά μας.

1. *Ξεκινήστε* το πρόγραμμα σας με την εντολή που ακολουθεί, για να *εισάγετε* τη βιβλιοθήκη `random`:

```
import random
```

*Προσθέστε* τώρα την εντολή:

```
dice1 = random.randint(1,6)
```

Στην εντολή αυτή χρησιμοποιείται η *συνάρτηση* `randint`, από τη βιβλιοθήκη `random`. Όταν εκτελεστεί το πρόγραμμα, θα παραχθεί μια τυχαία τιμή από το `1` μέχρι και το `6` και αυτή θα είναι η τιμή της μεταβλητής `dice1`.

2. Τώρα *προσθέστε* ακόμα μια εντολή που παράγει με τον ίδιο τρόπο την ένδειξη του δεύτερου ζαριού και την ονομάζει `dice2`.

*Θα χρησιμοποιήσουμε ξανά τη `randint()`, όπως προηγουμένως και θα αναθέσουμε το αποτέλεσμα που επιστρέφει στη μεταβλητή `dice2`:*

```
dice2 = random.randint(1,6)
```

3. Στο παιχνίδι αυτό δεν έχουν σημασία οι επιμέρους ενδείξεις των ζαριών, αλλά το *άθροισμά* τους. *Προσθέστε* μια εντολή που υπολογίζει το άθροισμα των `dice1` και `dice2` και ονομάζει το αποτέλεσμα `roll`.

*Η εντολή που θα προσθέσουμε στο πρόγραμμα είναι:*

```
roll = dice1 + dice2
```

4. *Προσθέστε* μια εντολή που εμφανίζει στον παίκτη τη ζαριά που έφερε, δηλαδή τις τιμές των μεταβλητών `dice1`, `dice2` και `roll`.

   Για παράδειγμα:

   ```
   Έριξες 3 και 6 = 9
   ```

   *Η εντολή που θα προσθέσουμε στο πρόγραμμα είναι:*

   ```
   print("Έριξες",dice1,"και",dice2,"=",roll)
   ```

   Εκτελέστε το πρόγραμμα μερικές φορές και σημειώστε τις ζαριές σας, όπως εμφανίζονται στην οθόνη.

   ```
   Έριξες  1  και  5  =  6
   Έριξες  2  και  1  =  3
   Έριξες  4  και  6  =  10
   ```

   Κάθε φορά που εκτελείτε το πρόγραμμα, η ζαριά σας είναι ίδια ή διαφορετική; Υπάρχει τρόπος να γνωρίζετε εκ των προτέρων τη ζαριά που θα φέρετε;

   *Η ζαριά μπορεί να διαφέρει σε κάθε εκτέλεση του προγράμματος ή μπορεί να είναι η ίδια! Ακριβώς επειδή οι αριθμοί των ζαριών είναι τυχαίοι δεν μπορούμε να γνωρίζουμε εκ των προτέρων το αποτέλεσμα.*

5. Είναι το ίδιο *πιθανό* να φέρουμε μια ζαριά με άθροισμα 7 και μια ζαριά με άθροισμα 2;

   Σκεφτείτε πόσοι πιθανοί συνδυασμοί ζαριών αντιστοιχούν στο ένα άθροισμα και πόσοι στο άλλο.

   *Ο μόνος συνδυασμός ζαριών που αντιστοιχεί στο άθροισμα 2 είναι το 1 και 1. Αντίθετα, το άθροισμα 7 προκύπτει από έξι διαφορετικούς συνδυασμούς ζαριών: 1 και 6, 2 και 5, 3 και 4 και τα αντίστροφά τους. Επομένως, η πιθανότητα να φέρουμε ζαριά με άθροισμα 2 είναι 2 στις 36 (5.56%), σαφώς μικρότερη από την πιθανότητα να φέρουμε ζαριά με άθροισμα 7, που είναι 6 στις 36 (16.67%).*

   Αντί να παράγουμε δύο ζαριές και να υπολογίζουμε το άθροισμά τους, θα μπορούσαμε να παράγουμε απευθείας ένα τυχαίο άθροισμα, με την εντολή `roll = random.randint(2,12)`. Είναι το ίδιο;

   Με τη `random.randint(2,12)` όλα τα πιθανά αθροίσματα είναι *ισοπίθανα.*

   *Όχι, γιατί σε αυτή την περίπτωση κάθε ζαριά θα είχε ακριβώς την ίδια πιθανότητα να εμφανιστεί. Δηλαδή στο προηγούμενο παράδειγμα η πιθανότητα να φέρουμε 2 ή 7 θα ήταν 1 στις 11.*

## Ρίξτα!

Όπως έχει τώρα το πρόγραμμα, τα ζάρια ρίχνονται αμέσως μόλις ξεκινήσει το παιχνίδι. Είναι όμως καλύτερο ν' αποφασίζει ο παίκτης

πότε θα ριχτεί η ζαριά, για να του δώσουμε την αίσθηση ότι παίζει.

6. *Προσθέστε*, στο σημείο που θεωρείτε κατάλληλο, μια εντολή που θα εμφανίζει στον παίκτη ένα μήνυμα, μια *προτροπή* να ρίξει τα ζάρια πατώντας το πλήκτρο `ENTER` .

```
Ρίξε τα ζάρια πατώντας το ENTER...
```

*Η εντολή που θα προσθέσουμε στο πρόγραμμα είναι:*

```
print("Ρίξε τα ζάρια πατώντας το ENTER...")
```

*Ένα πιθανό σημείο για να προσθέσουμε την παραπάνω εντολή είναι μετά από όλες τις εντολές που υπάρχουν ήδη στο πρόγραμμα.*

*Ένα άλλο πιθανό σημείο είναι πριν την εμφάνιση της προτροπής στον παίκτη να ρίξει τα ζάρια.*

Αμέσως μετά *προσθέστε*:

```
input()
```

Εκτελέστε το πρόγραμμα. Για ποιο λόγο χρησιμοποιούμε εδώ την `input()`;

*Η `input()` χρησιμοποιείται εδώ, προκειμένου το πρόγραμμα να "περιμένει" το πάτημα του ENTER από τον παίκτη, ώστε να προχωρήσει.*

Γιατί δεν αποδίδεται η τιμή που επιστρέφει η `input()` σε κάποια μεταβλητή, όπως γίνεται συνήθως;

*Διότι η τιμή που πληκτρολογεί στο σημείο αυτό ο χρήστης δεν έχει κάποιο νόημα, πέραν της "αναμονής" του προγράμματος. Επομένως, δεν χρειάζεται να την αποδώσουμε σε κάποια μεταβλητή.*

Έχει σημασία αν οι εντολές που μόλις προσθέσατε τοποθετηθούν *πριν* ή *μετά* από τις εντολές που δίνουν τιμή στις `dice1` και `dice2`; Αν ναι, ποια είναι η διαφορά;

*Δεν υπάρχει διαφορά ως προς τις ζαριές ή τα αποτελέσματα που εμφανίζονται στον παίκτη. Απλά αξίζει να σημειωθεί πως αν η `input()` μπει μετά από τις εντολές που δίνουν τιμή στις `dice1` και `dice2`, τότε ουσιαστικά η ζαριά παράγεται πριν το ζητήσει ο παίκτης, το ENTER απλά την εμφανίζει.*

## Πρώτη Εκδοχή

Τώρα που οι κύβοι ερρίφθησαν, ας εξετάσουμε αν ο χρήστης κέρδισε ή έχασε. Θα ξεκινήσουμε με μια απλή εκδοχή του παιχνιδιού.

7. Χρησιμοποιήστε την **`if`** για να ελέγχει το πρόγραμμά σας αν η ζαριά του χρήστη (δηλαδή η τιμή της μεταβλητής `roll`) είναι ίση με 7. Στην περίπτωση αυτή, θα πρέπει να εμφανίζεται το μήνυμα:

```
Κέρδισες με την πρώτη!
```

Σε διαφορετική περίπτωση, θα πρέπει να εμφανίζεται το μήνυμα:

```
Έχασες με την πρώτη...
```

Χρησιμοποιήστε το `==` για να ελέγξετε αν δύο τιμές είναι ίσες, όχι το `=` που χρησιμοποιείται για να δώσουμε τιμή σε μια μεταβλητή.

*Οι εντολές που θα προσθέσουμε στο πρόγραμμα είναι:*

```
if roll == 7:
    print("Κέρδισες με την πρώτη!")
else:
    print("Έχασες με την πρώτη...")
```

Χρησιμοποιήσατε μια **`if`**–**`else`** για να διαχωρίσετε ανάμεσα στις δύο περιπτώσεις ή δύο διαφορετικές **`if`**; Ποια πιστεύετε ότι είναι η διαφορά;

*Εφόσον οι περιπτώσεις που εξετάζουμε είναι αλληλοαποκλειόμενες, είναι πιο δόκιμο να χρησιμοποιήσουμε την* **`if`** – **`else`**. *Με τον τρόπο αυτό όταν η συνθήκη που εξετάζει η* **`if`** *δεν ισχύει θα εκτελούνται απευθείας οι εντολές που βρίσκονται "μέσα" στην* **`else`**. *Αν χρησιμοποιήσουμε δύο ξεχωριστές* **`if`** *η κάθε μια θα εξετάζεται ανεξάρτητα από το αν ισχύει ή όχι η προηγούμενη.*

Αν έχετε χρησιμοποιήσει δύο **`if`**, τότε *τροποποιήστε* το πρόγραμμα ώστε να χρησιμοποιεί μια **`if`**–**`else`**. Αυτή είναι η κατάλληλη δομή όταν έχουμε δύο *αμοιβαία αποκλειόμενες* περιπτώσεις, όπως εδώ.

Εκτελέστε το πρόγραμμα όσες φορές χρειαστεί, μέχρι να φέρετε `7`. Η πιθανότητα να γίνει αυτό είναι μία στις έξι, οπότε συνήθως θα χάνετε. Ανακοινώνεται σωστά το αποτέλεσμα σε κάθε περίπτωση;

Για να ελέγξετε αν το πρόγραμμα λειτουργεί σωστά θα μπορούσατε επίσης να «στήσετε» τη ζαριά, δηλαδή να *παρεμβάλλετε* προσωρινά πριν την **`if`** μια εντολή που δίνει μια διαγνωστική τιμή στην μεταβλητή `roll`.

*Ναι, εμφανίζεται μήνυμα επιτυχίας όταν η ζαριά είναι* `7` *και μήνυμα αποτυχίας σε κάθε άλλη περίπτωση.*

8. *Συμπληρώστε* τη συνθήκη της **`if`** έτσι ώστε ο χρήστης να κερδίζει αν φέρει `7` *ή* αν φέρει `11`.

Προσέξτε, ο έλεγχος αυτός απαιτεί μια *επιπλέον* συνθήκη σε *διάζευξη* με την υπάρχουσα. Για τη διάζευξη χρησιμοποιήστε τον τελεστή **`or`**.

*Θα τροποποιήσουμε τη συνθήκη που εξετάζει η* **`if`** *ως εξής:*

```
if roll == 7 or roll == 11:
    print("Κέρδισες με την πρώτη!")
```

*Ο τελεστής* **`or`** *συνδέει δύο συνθήκες μεταξύ τους και το αποτέλεσμα είναι αληθές όταν ισχύει οποιαδήποτε από τις δύο (ή και οι δύο).*

## Δεύτερη Εκδοχή

Στην πραγματικότητα, τα πράγματα δεν είναι και τόσο άσχημα για τον παίκτη. Κερδίζει αν φέρει `7` ή `11`, αλλά χάνει αμέσως μόνο αν

φέρει 2, 3 ή 12. Σε οποιαδήποτε άλλη περίπτωση, το παιχνίδι, προς το παρόν, θα λήγει ισόπαλο.

Τώρα πια οι διαφορετικές πιθανές εκβάσεις για το παιχνίδι είναι τρεις. Έτσι, για την επέκταση αυτή δεν αρκεί πια η απλή **if–else**, η οποία μπορεί να διακρίνει μόνο ανάμεσα σε δύο περιπτώσεις.

9. *Τροποποιήστε* την **if** που ελέγχει αν ο παίκτης κέρδισε ή όχι. *Συμπληρώστε* τη συνθήκη που λείπει, έτσι ώστε ο παίκτης να χάνει με την πρώτη αν φέρει 2, 3 ή 12:

```
if roll == 7 or roll == 11:
    print("Κέρδισες με την πρώτη!")
elif συνθήκη:   # συμπληρώστε τη συνθήκη
    print("Έχασες με την πρώτη...")
else:
    print("Ισοπαλία.")
```

Η **elif** είναι συντομογραφία της **else if**. Ελέγχει τη συνθήκη που την συνοδεύει (όπως και η **if**) μόνο εφόσον οι συνθήκες που προηγούνται είναι ψευδείς (**False**).

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

```
elif roll <= 3 or roll == 12:
```

Εκτελέστε μερικές φορές το πρόγραμμά σας. Ανακοινώνει σωστά το αποτέλεσμα του παιχνιδιού;

*Ναι, ανακοινώνει το κατάλληλο μήνυμα ανάλογα με τη ζαριά.*

Η συνθήκη που συμπληρώσατε στην **elif** πιστεύετε ότι ελέγχεται κάθε φορά που εκτελείται το πρόγραμμα ή μόνο σε μερικές περιπτώσεις (και πότε);

*Η συνθήκη της* **elif** *ελέγχεται* μόνο *όταν η συνθήκη που ελέγχει η* **if** *είναι ψευδής (*`False`*).*

*Αντικαταστήστε προσωρινά* την **elif** με μια **if**. Εκτελέστε μερικές φορές το πρόγραμμα μέχρι να φέρετε 7 ή 11. Τί έχει αλλάξει στην συμπεριφορά του προγράμματος και που πιστεύετε ότι οφείλεται αυτό;

*Πλέον, ακόμα και στην περίπτωση που ισχύει η συνθήκη της πρώτης* **if** *το πρόγραμμα εξετάζει* και *την περίπτωση της* **if** *που ακολουθεί. Επειδή προφανώς αυτή δεν ισχύει, δηλαδή η ζαριά δεν είναι* 2,3 *ή* 12*, εκτελούνται οι εντολές της* **else** *και εμφανίζεται* και *το μήνυμα* `Ισοπαλία` *μαζί με το μήνυμα* `Κέρδισες με την πρώτη!`

Μην ξεχάσετε να *επαναφέρετε* την **elif**, ώστε το πρόγραμμα να λειτουργεί και πάλι σωστά.

## Τρίτη Εκδοχή

Ας κάνουμε το παιχνίδι μας πιο ενδιαφέρον και λίγο πιο κοντά στους πραγματικούς κανόνες. Αν ο παίκτης δεν κερδίσει ούτε χάσει με την

πρώτη, τότε θα πρέπει να *ξαναρίξει*. Στην περίπτωση αυτή κερδίζει αν η δεύτερη ζαριά του έχει το ίδιο άθροισμα με την πρώτη, ενώ σε διαφορετική περίπτωση χάνει.

**10.** Στο ξεκίνημα της τρίτης περίπτωσης, μετά το **`else`**, *τροποποιήστε* την υπάρχουσα `print`, ώστε να ενημερώνει το χρήστη ποια είναι η τιμής της (δεύτερης) ζαριάς που πρέπει να φέρει για να κερδίσει.

Για παράδειγμα, αν το άθροισμα των ζαριών στην πρώτη προσπάθεια ήταν ίσο με `9`, τότε αυτό θα πρέπει να είναι το άθροισμα και στη δεύτερη ζαριά για να κερδίσει ο παίκτης.

```
Ξαναρίξε. Πρέπει να φέρεις 9
```

*Η εντολή `print` θα τροποποιηθεί όπως παρακάτω:*

```
print("Ξαναρίξε. Πρέπει να φέρεις", roll)
```

**11.** Το πρόγραμμα περιέχει *ήδη* εντολές που "ρίχνουν" τα δύο ζάρια.

```
print("Ρίξε τα ζάρια πατώντας το ENTER...")
input()
dice1 = random.randint(1,6)
dice2 = random.randint(1,6)
roll = dice1 + dice2
print("Έριξες", dice1, "και", dice2, "=", roll)
```

Στο δικό σας πρόγραμμα είναι πιθανό οι εντολές αυτές να έχουν λίγο διαφορετική σειρά, αλλά αυτό δεν έχει σημασία. Για τη δεύτερη ζαριά χρειαζόμαστε ακριβώς τις ίδιες εντολές.

*Αντιγράψτε* αυτές τις εντολές που αντιστοιχούν στην πρώτη ζαριά του παίκτη και *προσθέστε* τις στο σημείο όπου ο παίκτης πρέπει να ρίξει τη δεύτερη ζαριά του, δηλαδή στην τρίτη περίπτωση όπου ο παίκτης ούτε κερδίζει, ούτε χάνει με την πρώτη.

Εκτελέστε το πρόγραμμα. Σας ζητά να ρίξετε μια δεύτερη ζαριά;

*Ναι, όταν ο παίκτης δεν κερδίσει ούτε χάσει με την πρώτη, τότε εκτελούνται οι εντολές της **`else`** και το πρόγραμμα ζητά την επανάληψη της ρίψης.*

Εκτελέστε το πρόγραμμα όσες φορές χρειαστεί, μέχρι να τύχει να χάσετε ή να κερδίσετε με την πρώτη ζαριά. Μήπως το πρόγραμμα σας ζητά να ξαναρίξετε, ακόμα και σ' αυτή την περίπτωση;

*Όχι, δεν συμβαίνει κάτι τέτοιο. Το πρόγραμμα θα εκτελέσει τις εντολές της **`else`** μόνο στην περίπτωση που ο παίκτης δεν κερδίσει ή χάσει με την πρώτη.*

Για ποιο λόγο πιστεύετε ότι μπορεί να εμφανιστεί αυτό το πρόβλημα; Προσπαθήστε ν' απαντήσετε, ακόμα κι αν δεν συνέβη σε σας.

*Για να παρουσιάσει το πρόγραμμα την παραπάνω "προβληματική" συμπεριφορά θα πρέπει οι εντολές που αντιγράψαμε να εκτελούνται σε κάθε περίπτωση, δηλαδή ανεξάρτητα από τη ζαριά του παίκτη. Αυτό θα συμ-*

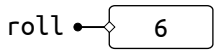
*βεί αν δεν προσθέσουμε τις κατάλληλες εσοχές μπροστά από τις εντολές, ώστε να μπουν "μεσα" στην* **else**.

Αν το πρόβλημα εμφανίστηκε και στο δικό σας πρόγραμμα, *διορθώστε* το τοποθετώντας τις κατάλληλες *εσοχές* μπροστά από τις εντολές που ξαναρίχνουν το ζάρι, υποδηλώνοντας έτσι ότι αυτές ανήκουν στην τρίτη περίπτωση.

12. Τώρα πρέπει να ελεγχθεί αν η δεύτερη ζαριά του παίκτη είναι ίση με την πρώτη, για να διαπιστωθεί αν κέρδισε ή έχασε. Σε ποια μεταβλητή είναι αποθηκευμένη η τιμή κάθε ζαριάς του παίκτη;

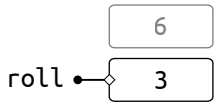
| | πρώτη ζαριά | δεύτερη ζαριά |
|---|---|---|
| μεταβλητή | ..... | `roll` |

*Στο αποτέλεσμα της πρώτης ζαριάς δίνεται το όνομα* `roll`. *Κατά τη δεύτερη ρίψη, χρησιμοποιείται το ίδιο όνομα για το αποτέλεσμα της δεύτερης ζαριάς, με συνέπεια το όνομα* `roll` *ν' αντιστοιχεί πλέον σε αυτή και να μην υπάρχει τρόπος αναφοράς στην πρώτη τιμή.*




Εικόνα: Η πρώτη ζαριά αποθηκεύεται στη μεταβλητή `roll`.

Με βάση τον παραπάνω πίνακα, μπορούμε να συγκρίνουμε μεταξύ τους τις δύο τιμές, για να διαπιστώσουμε αν είναι ίσες μεταξύ τους; Αν όχι, πως θα μπορούσαμε να αντιμετωπίσουμε αυτό το πρόβλημα;

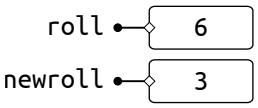



Εικόνα: Η νέα ζαριά αποθηκεύεται επίσης στη `roll`. Δεν υπάρχει πλέον τρόπος αναφοράς στην προηγούμενη τιμή της μεταβλητής.

*Όχι, δεν μπορούμε να τις συγκρίνουμε: η τιμή της πρώτης ζαριάς δεν είναι διαθέσιμη. Ούτε έχει νόημα να εξετάσουμε τη συνθήκη* `roll == roll` *μιας και είναι πάντα* Αληθής. *Για να διορθώσουμε το πρόβλημα πρέπει να αποθηκεύσουμε το αποτέλεσμα κάθε ζαριάς σε διαφορετική μεταβλητή.*

Στη δεύτερη ζαριά, *τροποποιήστε* τη `roll = dice1 + dice2` έτσι ώστε στο νέο άθροισμα των ζαριών να δίνεται διαφορετικό όνομα:

```
newroll = dice1 + dice2
```




Εικόνα: Η νέα ζαριά αποθηκεύεται σε διαφορετική μεταβλητή, τη `newroll`. Και οι δύο ζαριές είναι διαθέσιμες.

*Τροποποιήστε* επίσης την `print` που ακολουθεί, έτσι ώστε μετά τη δεύτερη ζαριά να εμφανίζεται στο χρήστη το νέο άθροισμα, δηλαδή η τιμή της `newroll`.

*Η εντολή* `print` *θα τροποποιηθεί όπως παρακάτω:*

```
print("Ξαναρίξε. Πρέπει να φέρεις", newroll)
```

Τώρα η τιμή της κάθε ζαριάς αποθηκεύεται σε διαφορετική μεταβλητή κι έτσι οι δύο τιμές μπορούν να συγκριθούν μεταξύ τους.

13. *Προσθέστε* μια **if** που θα ελέγχει αν ο παίκτης κέρδισε ή έχασε με τη δεύτερη ζαριά του. Στην περίπτωση που η δεύτερη ζαριά του παίκτη είναι ίση με την πρώτη, θα πρέπει να εμφανίζεται το μήνυμα:

```
Έφερες τη ζαριά-στόχο. Κέρδισες!
```

Σε διαφορετική περίπτωση, θα πρέπει να εμφανίζεται το μήνυμα:

```
Έφερες διαφορετική ζαριά. Έχασες...
```

*Οι εντολές που θα προσθέσουμε στο πρόγραμμα είναι:*

```
if roll == newroll:
    print("Έφερες τη ζαριά-στόχο. Κέρδισες!")
else:
    print("Έφερες διαφορετική ζαριά. Έχασες...")
```

Εκτελέστε το πρόγραμμα όσες φορές χρειαστεί, ώστε να κερδίσετε και να χάσετε τουλάχιστον μια φορά *ρίχνοντας και δεύτερη ζαριά*. Ανακοινώνεται σωστά το αποτέλεσμα σε κάθε περίπτωση;

*Ναι, εμφανίζεται το κατάλληλο μήνυμα σε κάθε περίπτωση.*

## Εξαρτήματα

14. Ας *εξετάσουμε* για λίγο ακόμα τις εντολές που σχετίζονται με τη ρίψη των ζαριών:

```
print("Ρίξε τα ζάρια πατώντας το ENTER...")
input()
dice1 = random.randint(1,6)
dice2 = random.randint(1,6)
roll = dice1 + dice2
print("Έριξες", dice1, "και", dice2, "=", roll)
```

Οι εντολές αυτές επαναλαμβάνονται σχεδόν αυτούσιες σε δύο σημεία του προγράμματος. Ουσιαστικά αποτελούν μια *ενότητα εντολών που επιτελούν μια ορισμένη λειτουργία*.

Ποια μεταβλητή θεωρείτε ότι κρατά το "αποτέλεσμα" αυτού του τμήματος κώδικα; Με άλλα λόγια, ποια είναι η τιμή που προκύπτει από αυτόν τον κώδικα και χρησιμοποιείται στο υπόλοιπο πρόγραμμα;

*Το αποτέλεσμα αποθηκεύεται στη μεταβλητή* `roll`.

15. Τώρα θα τοποθετήσουμε αυτές τις εντολές μέσα σε μια *συνάρτηση* και στη συνέχεια θα τις ενεργοποιούμε στα σημεία όπου τις χρειαζόμαστε, *καλώντας* την συνάρτηση. Μια συνάρτηση είναι ένα *υποπρόγραμμα* που επιτελεί μια συγκεκριμένη λειτουργία.

*Προσθέστε* στην αρχή του προγράμματος, κάτω από την **`import`**, τις εντολές που ακολουθούν. Εκτός από την πρώτη και την τελευταία γραμμή, οι υπόλοιπες εντολές υπάρχουν ήδη στο πρόγραμμα, οπότε μπορείτε απλά να τις κάνετε copy–paste:

```
def rollDice():
    print("Ρίξε τα ζάρια πατώντας το ENTER...")
    input()
    dice1 = random.randint(1,6)
    dice2 = random.randint(1,6)
    roll = dice1 + dice2
    print("Έριξες", dice1, "και", dice2, "=", roll)
    return roll
```


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

Με τον τρόπο αυτό *ορίζεται* η συνάρτηση `rollDice()`, οι εντολές τις οποίας υλοποιούν το "ρίξιμο" των ζαριών. Σημειώστε πως το αποτέλεσμα της συνάρτησης, δηλαδή η τιμή της μεταβλητής `roll`, *επιστρέφεται* από την συνάρτηση με την εντολή **`return`**.

16. Η συνάρτηση `rollDice()` είναι σαν ένα μικρό "εξάρτημα" που επιτελεί μια συγκεκριμένη λειτουργία. Ωστόσο, αν και την κατασκευάσαμε, δεν την χρησιμοποιούμε πουθενά προς το παρόν.

    Στο κύριο πρόγραμμα, στο σημείο όπου ρίχνονται για πρώτη φορά τα ζάρια, *διαγράψτε* τις εντολές του βήματος 14, που σχετίζονται με τη ρίψη των ζαριών και *αντικαταστήστε* τις με τη γραμμή:

    ```
    roll = rollDice()
    ```

    Με τη γραμμή αυτή ενεργοποιούμε ή *καλούμε* την `rollDice()`, πραγματοποιώντας έτσι τη ρίψη των ζαριών. Την τιμή που επιστρέφεται από την συνάρτηση, δηλαδή το άθροισμα των ζαριών, την ονομάζουμε `roll`.

    Εκτελέστε το πρόγραμμα. Λειτουργεί σωστά;

    *Ναι, το πρόγραμμα λειτουργεί σωστά.*

    Σε περίπτωση που κάτι πάει στραβά, βεβαιωθείτε ότι έχετε διαγράψει από το κύριο πρόγραμμα τις εντολές που αντιστοιχούν στο ρίξιμο της πρώτης ζαριάς, αφού αυτές εκτελούνται πλέον όταν καλείται το υποπρόγραμμα. Βεβαιωθείτε επίσης ότι η κλήση του υποπρογράμματος γίνεται στο κατάλληλο σημείο.

    Παρατηρεί ο χρήστης κάποια διαφορά στη λειτουργία του προγράμματος, μετά από αυτή την τροποποίηση;

    *Όχι, το πρόγραμμα δεν έχει καμία διαφορά για τον χρήστη, αφού η συνάρτηση κατά την κλήση της εκτελεί ακριβώς τις ίδιες εντολές με προηγουμένως.*

17. *Εντοπίστε* το σημείο του προγράμματος όπου ρίχνονται για δεύτερη φορά τα ζάρια. *Διαγράψτε* τις σχετικές εντολές που σχετίζονται με τη ρίψη των ζαριών και *αντικαταστήστε* τις με μια ακόμα κλήση της συνάρτησης `rollDice()`.

    Φροντίστε να αποθηκεύσετε την τιμή που επιστρέφεται από την συνάρτηση στη μεταβλητή `newroll`.

*Η εντολή που θα προσθέσουμε στο πρόγραμμα είναι:*

```
newroll = rollDice()
```

Εκτελέστε το πρόγραμμα. Λειτουργεί σωστά;

*Ναι, το πρόγραμμα όσον αφορά τον χρήστη εκτελεί ακριβώς τις ίδιες λειτουργίες με προηγουμένως.*

Τι πιστεύετε ότι κερδίζουμε με την αντικατάσταση των αρχικών εντολών από την κλήση της συνάρτησης `rollDice()`;

*Η χρήση των υποπρογραμμάτων έχει το πλεονέκτημα της "μείωσης" του κώδικα που γράφει ο προγραμματιστής, αφού χρειάζεται να γράψει τις εντολές μόνο σε ένα σημείο, δηλαδή στο σώμα της συνάρτησης. Στη συνέχεια μπορεί να τις ενεργοποιεί κατά βούληση σε οποιοδήποτε σημείο του προγράμματος, καλώντας απλά τη συνάρτηση. Με τον τρόπο αυτό ο κώδικας γίνεται πιο συμπαγής, ξεκάθαρος και μικρότερος σε έκταση.*

## Τελική Εκδοχή

Στη σημείο αυτό, ο κώδικας του κύριου προγράμματος, μετά τον ορισμό της `rollDice()`, πρέπει να μοιάζει κάπως έτσι:

```
roll = rollDice()
if roll == 7 or roll == 11:
    print("Κέρδισες με την πρώτη!")
elif roll <= 3 or roll == 12:
    print("Έχασες με την πρώτη...")
else:
    print("Ξαναρίξε. Πρέπει να φέρεις", roll)
    newroll = rollDice()
    if newroll == roll:
        print("Έφερες τη ζαριά-στόχο. Κέρδισες!")
    else:
        print("Έφερες διαφορετική ζαριά. Έχασες...")
```

Είμαστε κοντά στην τελική εκδοχή του παιχνιδιού. Στην πραγματικότητα, αν ο παίκτης δεν κερδίσει ούτε χάσει με την πρώτη ζαριά, τότε δεν ξαναρίχνει τα ζάρια μόνο μια φορά αλλά *επαναληπτικά*, μέχρι να φέρει μια ζαριά ίση με την πρώτη του, οπότε κερδίζει. Αν όμως στο μεταξύ, φέρει 7, τότε χάνει.

18. Στο ξεκίνημα της τρίτης περίπτωσης, όταν δηλαδή το παιχνίδι δεν έχει λήξει από την πρώτη ζαριά, *προσθέστε* "μέσα" στην **else**:

```
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.

Αν εκτελέσετε το πρόγραμμα και εμφανιστεί μήνυμα σφάλματος του τύπου `IndentationError: expected an indented block`, τότε ελέγξτε προσεκτικά τις *εσοχές*. Η **`while`** πρέπει να βρίσκεται δεξιότερα της **`else`** και οι εντολές που ακολουθούν ακόμα δεξιότερα.

Εκτελέστε το πρόγραμμά σας. Αν κερδίσετε ή χάσετε με την πρώτη ζαριά τότε εκτελέστε το και πάλι. 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 επαναληπτικά, δηλαδή ξανά και ξανά. Προς το παρόν η εκτέλεση των εντολών δεν τερματίζεται.*

**19.** *Τροποποιήστε* την **`if`** που βρίσκεται μέσα στην επανάληψη. Η πρώτη περίπτωση δε χρειάζεται αλλαγή: ο παίκτης κερδίζει όταν φέρει την ίδια ζαριά με την αρχική. Ωστόσο, δεν χάνει σε οποιαδήποτε άλλη περίπτωση, αλλά μόνο όταν φέρει 7.

```
        if newroll == roll:
            print("Έφερες τη ζαριά-στόχο. Κέρδισες!")
        elif συνθήκη:   # συμπληρώστε την συνθήκη
            print("Έφερες 7. Έχασες...")
```

*Η συνθήκη που θα προσθέσουμε στην* **`elif`** *είναι:*

```
elif newroll == 7:
```

Παρατηρήστε ότι δεν υπάρχει πια η **`else`**. Για ποιο λόγο πιστεύετε ότι συμβαίνει αυτό;

*Γιατί δεν χρειάζεται πλέον να περιγράψουμε τις ενέργειες που θα εκτελεστούν όταν ο παίκτης δεν φέρει τη ζαριά στόχο ή το 7. Σε αυτή την περίπτωση το πρόγραμμα απλά επαναλαμβάνει τις εντολές που βρίσκονται μέσα στη* **`while`**.

Εκτελέστε το πρόγραμμα. Υπάρχει κάτι που σας ενοχλεί; Κάτι που φαίνεται να μη δουλεύει σωστά;

*Το πρόγραμμα δεν τερματίζει τη λειτουργία του ακόμα και αν ο παίκτης φέρει τη ζαριά–στόχο ή το 7. Συνεχίζει να τον προτρέπει να ρίξει τα ζάρια.*

**20.** Η εντολή **`break`** διακόπτει την επανάληψη μέσα στην οποία βρίσκεται αμέσως μόλις εκτελεστεί. *Προσθέστε* την **`break`** στο σημείο που θεωρείτε κατάλληλο, έτσι ώστε η επαναληπτική ρίψη των ζαριών να τερματίζεται όταν ο παίκτης ξαναφέρει την αρχική ζαριά–στόχο.

*Η εντολή* **`break`** *θα προστεθεί μέσα στην* **`if`** *που εξετάζει ότι ο παίκτης έφερε τη ζαριά–στόχο, όπως παρακάτω:*

```
if newroll == roll:
    print("Έφερες τη ζαριά στόχο. Κέρδισες!")
    break
```

Εκτελέστε το πρόγραμμα. Αν κερδίσετε ή χάσετε με την πρώτη ζαριά τότε εκτελέστε το και πάλι. Σταματά η επανάληψη όταν ο παίκτης φέρει τη ζαριά–στόχο σε κάποια από τις επαναλαμβανόμενες ζαριές;

*Ναι, το πρόγραμμα σταματά.*

Αν απαντήσατε αρνητικά, τότε βεβαιωθείτε ότι έχετε τοποθετήσει την **break** μέσα στην αντίστοιχη περίπτωση της εμφωλευμένης **if**.

21. *Προσθέστε* την **break** στο σημείο που θεωρείτε κατάλληλο, έτσι ώστε η επαναληπτική ρίψη των ζαριών να τερματίζεται όταν ο παίκτης φέρει 7.

*Η εντολή* **break** *θα προστεθεί μέσα στην* **elif** *που εξετάζει ότι ο παίκτης έφερε* 7*, όπως παρακάτω:*

```
elif newroll == 7:
    print("Έφερες 7. Έχασες!")
    break
```

Εκτελέστε το πρόγραμμα. Σταματά η επανάληψη όταν ο παίκτης φέρει 7 σε κάποια από τις επαναλαμβανόμενες ζαριές;

*Ναι, το πρόγραμμα σταματά όταν ο παίκτης φέρει* 7*.*