Μπαρμπούτι

2

Φύλλο Εργασίας

22 Ιουλίου 2016
13:47

Σε πολλές αμερικάνικες ταινίες οι πρωταγωνιστές γίνονται εκατομμυριούχοι ή χάνουν τα πάντα παίζοντας ένα παιχνίδι με ζάρια, άγνωστο στους περισσότερους από μας. Η πλησιέστερη ελληνική εκδοχή του παιχνιδιού είναι το μπαρμπούτι, πηγή έμπνευσης για πολλούς άτυχους ρεμπέτες.

Ακολουθώντας τα βήματα του φύλλου εργασίας θ' αναπτύξουμε ένα πρόγραμμα με το οποίο ο χρήστης θα παίζει αυτό το παιχνίδι. Η υλοποίηση του παιχνιδιού θα γίνει σε στάδια, ξεκινώντας από απλούστερες εκδοχές του, μέχρι να φτάσουμε στην τελική, ολοκληρωμένη του μορφή.

Έννοιες: δομή επιλογής, δομή επανάληψης, υποπρογράμματα

Για ν' ακολουθήσετε αυτό το φύλλο εργασίας, θα πρέπει *ήδη* να μπορείτε να κάνετε ένα πρόγραμμα να εμφανίζει μηνύματα, να διαβάζει τιμές και να επιλέγει την συμπεριφορά του ανάλογα με τις συνθήκες που επικρατούν κατά την εκτέλεσή του. Διαφορετικά, θα πρέπει πρώτα ν' ανατρέξετε στο *εισαγωγικό υλικό*.

Εισαγωγικό υλικό:
`pythonies.mysch.gr/chapters/answer.pdf`
`answer-worksheet.pdf`
`guess-standalone.pdf`

Σε αυτό το φύλλο θα εμβαθύνουμε στα παραπάνω και θα δούμε επίσης πως μπορούμε να κάνουμε τα προγράμματά μας να εκτελούν *επαναληπτικά*, δηλαδή ξανά και ξανά, τις ίδιες εντολές. Θα δούμε επίσης πως μπορούμε να φτιάχνουμε *υποπρογράμματα*, δηλαδή ομάδες εντολών που υλοποιούν μια συγκεκριμένη λειτουργία και *καλούνται* στα σημεία όπου θέλουμε να εκτελεστούν.

Διαβάστε το αντίστοιχο κεφάλαιο:
`pythonies.mysch.gr/chapters/craps.pdf`

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`.

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` κι ονομάζει το αποτέλεσμα `roll`.

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
```

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

```
Έριξες ..... και ..... = .....

Έριξες ..... και ..... = .....

Έριξες ..... και ..... = .....
```

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;

.....

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.

.....

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*.

.....

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...
```

Αμέσως μετά *προσθέστε*:

```
input()
```

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

. . . . . .

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

. . . . . .

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;

. . . . . .

. . . . . .

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.

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

. . . . . .

. . . . . .

. . . . . .

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`.

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**.

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**).

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

. . . . . .

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

. . . . . .

. . . . . .

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;

. . . . . .

. . . . . .

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
```

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;

. . . . . .

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;

. . . . . .

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

. . . . . .

. . . . . .

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	. . . . .	. . . . .

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;

. . . . . .

. . . . . .

. . . . . .

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

```
newroll = dice1 + dice2
```

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

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;

. . . . . .

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;

. . . . . .

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;

. . . . . .

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;

. . . . . .

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`.

Εκτελέστε το πρόγραμμα. Λειτουργεί σωστά;

. . . . . .

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

. . . . . .

. . . . . .

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`.

. . . . . .

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...")
```

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

. . . . . .

. . . . . .

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

. . . . . .

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.

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;

. . . . . .

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.

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

. . . . . .

Δραστηριότητες για Εξάσκηση

Για περισσότερη εξάσκηση στις έννοιες που γνωρίσατε σ' αυτό το φύλλο εργασίας, μπορείτε ν' ανατρέξετε στις ασκήσεις του Κεφαλαίου "Μπαρμπούτι".

pythonies.mysch.gr/complete