Το Παιχνίδι της Αφαίρεσης

Ενδεικτικές Απαντήσεις Φύλλου Εργασίας

4

10 Σεπτεμβρίου 2016
10:01

Το Στήσιμο

Ένα πλήθος από αντικείμενα (π.χ. σπίρτα, ξυλάκια) τοποθετούνται στη σειρά και ο κάθε ένας από τους δύο παίκτες αφαιρεί με τη σειρά του από ένα μέχρι και τρία αντικείμενα, μέχρι να μη μείνει κανένα. Ο παίκτης που θα πάρει το τελευταίο αντικείμενο *χάνει* το παιχνίδι.

1. Ας υποθέσουμε ότι τα αντικείμενα που χρησιμοποιούν οι παίκτες είναι σπίρτα. Οι κανόνες του παιχνιδιού δεν προσδιορίζουν το αρχικό πλήθος σπίρτων, μπορούμε λοιπόν να το ορίσουμε μόνοι μας.

   Γίνεται το παιχνίδι να ξεκινάει κάθε φορά μ' ένα διαφορετικό αριθμό σπίρτων; Αν ναι, ποιος είναι ο τρόπος;

   Θα μπορούσε το παιχνίδι να ξεκινά επιλέγοντας μια τυχαία τιμή ως αρχικό αριθμό σπίρτων ή να ερωτάται ο πρώτος παίκτης με πόσα σπίρτα θα ήθελε να ξεκινήσει.

   Ας ονομάσουμε το πλήθος των σπίρτων `matches`. Μπορούμε να αρχικοποιήσουμε τη μεταβλητή `matches` με μια τυχαία τιμή (ας πούμε από το 7 μέχρι και το 21).

   Προσθέστε τις κατάλληλες εντολές στο πρόγραμμά, ώστε να εισάγει τη βιβλιοθήκη `random` και να παράγει μια τυχαία τιμή ανάμεσα στο `7` και στο `21`, την οποία και θ' αποθηκεύει στη μεταβλητή `matches`.

   Θα ξεκινήσουμε το πρόγραμμα με τις εντολές:

   ```
   import random
   matches = random.randint(7,21)
   ```

2. *Προσθέστε* την κατάλληλη εντολή στο πρόγραμμά, προκειμένου να εμφανίζει τον αρχικό αριθμό σπίρτων στον παίκτη. Για παράδειγμα:

   ```
   Αρχικό πλήθος σπίρτων: 13
   ```

   Θα προσθέσουμε την εντολή:

   ```
   print("Αρχικό πλήθος σπίρτων:", matches)
   ```

   Εκτελέστε μερικές φορές το πρόγραμμά σας. Εμφανίζεται διαφορετικός αριθμός σπίρτων σε κάποιες εκτελέσεις;

Σημείωση: Δεν αποκλείεται κάποιες από τις τυχαίες τιμές να είναι οι ίδιες.

Εκτελώντας το πρόγραμμα πέντε φορές, εμφανίστηκε ως αρχικό πλήθος σπίρτων το `20`*, το* `18`*, το* `20`*, το* `15` *και το* `9`*.*

Δεν είναι παράξενο που εκτελέσαμε πέντε φορές το πρόγραμμα και το `20` *εμφανίστηκε δύο φορές, όπως δεν θα ήταν παράξενο αν ρίχναμε πέντε φορές ένα ζάρι και φέρναμε δύο φορές την ίδια ζαριά (ή και περισσότερες). Αν όμως ρίχναμε 10000 φορές ένα ζάρι και φέρναμε 4000 φορές την ίδια ζαριά, τότε θα είχαμε υποψίες ότι τα αποτελέσματα του συγκεκριμένου ζαριού δεν είναι και τόσο "τυχαία". Υπάρχουν συγκεκριμένοι έλεγχοι που μπορούμε να κάνουμε για να ισχυριστούμε τεκμηριωμένα ότι μια διαδικασία δεν είναι πραγματικά και τόσο τυχαία, αλλά για να είναι αξιόπιστοι θα πρέπει η διαδικασία να επαναληφθεί πάρα πολλές φορές.*

3. *Συμπληρώστε* το πρόγραμμα ώστε να ζητά από τον παίκτη τον αριθμό των σπίρτων που θα αφαιρέσει, εμφανίζοντας κατάλληλη προτροπή. Για παράδειγμα:

```
Πόσα σπίρτα θέλεις;
```

Αποθηκεύστε την απάντηση του παίκτη στη μεταβλητή `removed`.

Θα προσθέσουμε τις εντολές:

```
print("Πόσα σπίρτα θέλεις;")
removed = int(input())
```

Είναι απαραίτητο να χρησιμοποιήσουμε την `int()` *για να μετατρέψουμε την τιμή που πληκτρολογεί ο παίκτης σε ακέραια, πριν την αποθηκεύσουμε στη μεταβλητή* `removed`*.*

4. *Προσθέστε* την παρακάτω εντολή προκειμένου να μειωθεί ο αριθμός των σπίρτων `matches` ανάλογα με το πλήθος των σπίρτων `removed` που ζήτησε ο παίκτης να αφαιρέσει.

```
    matches = matches - removed
```

Εκτελέστε το πρόγραμμά σας. Λειτουργεί σωστά ή εμφανίζει κάποιο μήνυμα σφάλματος;

Το πρόγραμμα λειτουργεί χωρίς να εμφανιστεί μήνυμα σφάλματος.

Σε περίπτωση που σας εμφανίζει το παρακάτω μήνυμα σφάλματος:

```
TypeError: unsupported operand type(s) for -
```

βεβαιωθείτε ότι έχετε χρησιμοποιήσει την `int()` όταν ζητάτε τον αριθμό των σπίρτων που θα αφαιρέσει ο παίκτης, προκειμένου να μετατρέψετε την τιμή αυτή σε ακέραιο αριθμό.

5. *Συμπληρώστε* το πρόγραμμα με την κατάλληλη εντολή ώστε να εμφανίζεται στους παίκτες ο αριθμός των σπίρτων που απομένουν. Έτσι θα μπορούν ν' αποφασίζουν για το πλήθος των σπίρτων που θα αφαιρέσουν στον επόμενο γύρο. Για παράδειγμα:

```
Σπίρτα που απομένουν: 15
```

Θα προσθέσουμε την εντολή:

```
print("Σπίρτα που απομένουν:", matches)
```

Εκτελέστε το πρόγραμμά σας. Εμφανίζει σωστά τον αριθμό των σπίρτων που απομένουν;

Ναι. Κατά την εκτέλεση του προγράμματος, το παιχνίδι ξεκίνησε με 14 *σπίρτα. Ζητήσαμε να αφαιρέσουμε* 3 *και εμφανίστηκε μήνυμα ότι απομένουν* 11.

Κάνε Ένα Γύρο

6. Η αφαίρεση σπίρτων από τους παίκτες δεν γίνεται μόνο μια φορά, αλλά πολλές. Θα χρησιμοποιήσουμε την εντολή **while** προκειμένου οι εντολές να εκτελούνται επαναληπτικά.

   Το συγκεκριμένο παιχνίδι συνεχίζεται όσο απομένει τουλάχιστον ένα σπίρτο. Τι τιμές πρέπει να έχει η μεταβλητή `matches`, προκειμένου να συνεχίζεται η επανάληψη;

   Όσο η `matches` *παίρνει θετικές τιμές, δηλαδή όσο υπάρχουν ακόμα σπίρτα στο τραπέζι, το παιχνίδι θα πρέπει να συνεχίζεται.*

   Προσθέστε την εντολή **while** μαζί με την κατάλληλη συνθήκη που εξετάζει την τιμή της `matches`, όπως την περιγράψατε προηγουμένως. *Εμφωλεύστε* τις εντολές που πιστεύετε ότι χρειάζεται να εκτελούνται επαναληπτικά, προσθέτοντας τις κατάλληλες εσοχές.

   Πρέπει να σκεφτούμε ποιες εντολές θα πρέπει να επαναλαμβάνονται, ώστε να τις εμφωλεύσουμε στη **while**.

   Οι εντολές που αφορούν στην τυχαία επιλογή και εμφάνιση του αρχικού πλήθους σπίρτων θα πρέπει να εκτελούνται μόνο μία φορά, στην αρχή του παιχνιδιού, επομένως δεν θα συμπεριληφθούν στην επανάληψη. Αντίθετα, όλες οι εντολές που ακολουθούν, η ανάγνωση του πλήθους των σπίρτων που θα αφαιρεθούν, η απομείωση του πλήθους των σπίρτων και η εμφάνιση των σπίρτων που απομένουν, θα πρέπει να εκτελούνται σε κάθε γύρο. Γι' αυτό θα τις τοποθετήσουμε μετά τη **while** *και θα τους προσθέσουμε εσοχές.*

   Με την προσθήκη της **while**, *το πρόγραμμα έχει πλέον ως εξής:*

```
import random
matches = random.randint(7,21)
print("Αρχικό πλήθος σπίρτων:", matches)
while matches > 0:
    print("Πόσα σπίρτα θέλεις;")
    removed = int(input())
    matches = matches - removed
    print("Σπίρτα που απομένουν:", matches)
```

Εκτελέστε το πρόγραμμά σας. Τερματίζει η επανάληψη και, αν ναι, πότε γίνεται αυτό;

Εκτελέσαμε το πρόγραμμα και σε κάθε κύκλο αφαιρούσαμε τυχαία μερικά σπίρτα. Η επανάληψη τερματίστηκε όταν τα σπίρτα τελείωσαν, δηλαδή όταν η `matches` *έγινε* `0`*. Δοκιμάσαμε να αφαιρέσουμε και περισσότερα σπίρτα απ' όσα απέμεναν. Το πλήθος των σπίρτων έγινε αρνητικό και πάλι η επανάληψη τερματίστηκε.*

Επομένως, η επανάληψη συνεχίζεται όσο ισχύει ότι `matches > 0` *και τερματίζεται όταν ισχύει το αντίθετο, δηλαδή όταν* `matches <= 0`*.*

Αν η επανάληψη δεν σταματά τότε ελέγξτε ότι έχετε τοποθετήσει την εντολή που μειώνει τον αριθμό των σπίρτων μέσα στην **`while`** χρησιμοποιώντας τις κατάλληλες εσοχές και εκτελέστε ξανά το πρόγραμμα.

Τι τιμή παίρνει η συνθήκη που εξετάζει η **`while`**, ώστε να σταματήσει η επανάληψη;

Για να σταματήσει η επανάληψη, η συνθήκη της **`while`** *πρέπει να ελεγχθεί και να διαπιστωθεί ότι είναι ψευδής, δηλαδή ότι έχει την τιμή* **`False`***. Αντιθέτως, κάθε φορά που διαπιστώνεται ότι συνθήκη της* **`while`** *είναι αληθής, δηλαδή ότι έχει την τιμή* **`True`***, ξεκινά ένας ακόμα κύκλος της επανάληψης.*

7. Γιατί πιστεύετε ότι χρησιμοποιήσαμε τη μεταβλητή `matches` στο βήμα 4 για να αποθηκεύσουμε εκ νέου το αποτέλεσμα; Τι διαφορά θα είχε αν είχαμε χρησιμοποιήσει ένα άλλο όνομα στη θέση της;

Η τιμή της παράστασης `matches - removed` *αποθηκεύεται εκ νέου στη* `matches` *επειδή η τιμή της πρέπει να τροποποιηθεί, για να αντικατοπτρίζει το γεγονός ότι τα σπίρτα που απομένουν στο τραπέζι μειώθηκαν.*

Αν η τιμή της παράστασης `matches - removed` *αποθηκευτεί σε άλλη μεταβλητή, τότε η* `matches` *θα μείνει αμετάβλητη και ίση με το αρχικό πλήθος σπίρτων. Σε κάθε νέο γύρο του παιχνιδιού, η τιμή της έκφρασης* `matches - removed` *θα βασίζεται συνεχώς στο αρχικό πλήθος σπίρτων και δεν θα αντιστοιχεί στο πλήθος των σπίρτων που απομένουν.*

Τροποποιήστε την εντολή του βήματος 4 που αφαιρεί τα σπίρτα που ζήτησε ο παίκτης, ώστε να μην αποθηκεύει το αποτέλεσμα της ξανά στη μεταβλητή `matches`, αλλά σε μια νέα μεταβλητή, όπως παρακάτω:

```
remaining = matches - removed
```

Τροποποιήστε και την `print` που εμφανίζει τον αριθμό των σπίρτων που απομένουν, ώστε να εμφανίζει πλέον τη μεταβλητή `remaining`.

Εκτελέστε το πρόγραμμα δίνοντας διάφορες τιμές στον αριθμό των σπίρτων που αφαιρεί ο παίκτης. Τι παρατηρείτε;

Εκτελώντας το πρόγραμμα, το παιχνίδι ξεκίνησε με `13` *σπίρτα. Ζητήσαμε να αφαιρέσουμε* `3` *σπίρτα και το πρόγραμμα εμφάνισε μήνυμα ότι*

το πλήθος των σπίρτων που απομένει είναι `10`*. Στη συνέχεια όμως, ζητήσαμε να αφαιρέσουμε άλλα* `3` *σπίρτα και το πρόγραμμα εμφάνισε μήνυμα ότι το πλήθος των σπίρτων που απομένει είναι και πάλι* `10`*. Μετά ζητήσαμε ν' αφαιρέσουμε* `1` *σπίρτο και το πρόγραμμα εμφάνισε μήνυμα ότι το πλήθος των σπίρτων που απομένει είναι* `12`*. Το πλήθος των σπίρτων που εμφανίζει το πρόγραμμα ότι απομένουν δεν είναι σωστό.*

Γιατί πιστεύετε ότι συμβαίνει αυτό;

Επειδή η `matches` *δεν αλλάζει ποτέ τιμή και αντιστοιχεί στο αρχικό πλήθος σπίρτων. Συνεπώς και η* `remaining`*, της οποίας η τιμή δίνεται από την έκφραση* `matches - removed`*, αντιστοιχεί στο πλήθος των σπίρτων που απομένουν αν αφαιρέσουμε* `removed` *σπίρτα από το αρχικό πλήθος.*

Επαναφέρετε το πρόγραμμα, ώστε να χρησιμοποιεί τη μεταβλητή `matches` αντί για τη μεταβλητή `remaining`, όπως ήταν αρχικά.

Ποιός Παίζει;

Σε κάθε κύκλο του παιχνιδιού, το πρόγραμμα ρωτάει πόσα σπίρτα θα αφαιρεθούν. Ωστόσο, δεν καταγράφει ποιος από τους δύο παίκτες είναι που αφαιρεί κάθε φορά τα σπίρτα κι έτσι δεν είναι σε θέση να υπολογίσει ποιος είναι ο νικητής όταν τα σπίρτα τελειώσουν.

8. Ένας τρόπος για να λύσουμε το πρόβλημα είναι να χρησιμοποιήσουμε μια μεταβλητή `player` που να δείχνει τον *αριθμό* του παίκτη που παίζει σε κάθε γύρο.

   Τι τιμές θα παίρνει η μεταβλητή αυτή; Πώς θ' αλλάζει σε κάθε γύρο;

   Η μεταβλητή θα μπορούσε να έχει ως τιμή το `1` *ή το* `2` *και να εναλλάσσεται σε κάθε γύρο ανάμεσα σε αυτές τις δύο τιμές.*

   Μια απλή προσέγγιση είναι να χρησιμοποιήσουμε μια μεταβλητή `player`, η οποία σε κάθε γύρο θα παίρνει *εναλλάξ* την τιμή `1` ή `2`, υποδεικνύοντας έτσι ποιος παίκτης έχει σειρά να παίξει.

   Αρχικά, πριν ξεκινήσει η διαδικασία του παιχνιδιού, ορίζουμε ποιος παίκτης ξεκινάει πρώτος: αυτός θα είναι πάντα ο παίκτης με αριθμό `1`.

   Προσθέστε την κατάλληλη εντολή, ώστε η μεταβλητή `player` να παίρνει σαν αρχική τιμή το `1`.

   Θα προσθέσουμε πριν την επανάληψη την εντολή:

   ```
   player = 1
   ```

   Τοποθετήσατε αυτή την εντολή πριν από τη **`while`** ή μέσα σε αυτή; Ποια πιστεύετε ότι είναι η διαφορά;

   Αν τοποθετήσουμε την εντολή μέσα στη **`while`** *τότε θα επαναλαμβάνεται σε κάθε γύρο, με αποτέλεσμα να παίζει πάντα ο πρώτος παίκτης.*

9. *Τροποποιήστε* την εντολή του βήματος 3 που εμφανίζει το μήνυμα προτροπής, ώστε πλέον να απευθύνεται σε συγκεκριμένο παίκτη. Χρησιμοποιήστε κατάλληλα τη μεταβλητή `player`. Για παράδειγμα:

```
Παίκτη 1 πόσα σπίρτα θέλεις;
```

Τροποποιούμε την εντολή που εμφανίζει το μήνυμα προτροπής ως εξής:

```
    print("Παίκτη", player, "πόσα σπίρτα θέλεις;")
```

Εκτελέστε το πρόγραμμά σας. Εμφανίζει τον αριθμό του παίκτη που παίζει σε κάθε γύρο;

Όχι. Η `player` *έχει μονίμως την τιμή* `1` *κι έτσι το πρόγραμμα εμφανίζει πάντα τον ίδιο αριθμό παίκτη. Θα πρέπει η τιμή της* `player` *να εναλλάσσεται σε κάθε γύρο μεταξύ των τιμών* `1` *και* `2`*.*

10. Στο τέλος κάθε κύκλου της επανάληψης, η τιμή της μεταβλητής `player` θα πρέπει να τροποποιείται, ώστε να υποδεικνύει τον επόμενο παίκτη που έχει σειρά να παίξει.

Προσθέστε μέσα στη **`while`** τις κατάλληλες εντολές που θα δίνουν στην `player` την τιμή `2` όταν αυτή έχει την τιμή `1` και αντιστρόφως. Χρησιμοποιήστε την **`if`**–**`else`** για το σκοπό αυτό.

Θα προσθέσουμε τις ακόλουθες εντολές μέσα στη **`while`***:*

```
    if player == 1:
        player = 2
    else:
        player = 1
```

Είναι σημαντικό το σημείο μέσα στη **`while`** *στο οποίο θα τοποθετηθούν αυτές οι εντολές. Εμείς τις τοποθετήσαμε μετά από όλες τις εντολές που υπήρχαν ήδη στη* **`while`***, ώστε η εναλλαγή του παίκτη να είναι το τελευταίο πράγμα που γίνεται σε κάθε γύρο. Με δεδομένο ότι η* `player` *ξεκινά με την τιμή* `1`*, θα θέλαμε η πρώτη εναλλαγή να γίνει αφού παίξει ο πρώτος παίκτης.*

Εκτελέστε το πρόγραμμά. Εναλλάσσει πλέον σε κάθε γύρο τον αριθμό του παίκτη που παίζει;

Ναι, το μήνυμα απευθύνεται εναλλάξ στον έναν και στον άλλο παίκτη.

Εμφανίζεται σωστά ο αριθμός του παίκτη, δηλαδή την πρώτη φορά ο αριθμός `1` και στον επόμενο γύρο ο αριθμός `2`;

Ναι.

Αν εμφανίζεται στον πρώτο γύρο ο αριθμός `2` και στον επόμενο γύρο ο αριθμός `1`, που πιστεύετε ότι μπορεί να οφείλεται αυτό το σφάλμα;

Να απαντήσετε ακόμα κι αν η συμπεριφορά του δικού σας προγράμματος είναι ορθή.

Έχουμε καθορίσει ότι η αρχική τιμή της `player` *θα είναι το* `1`*. Αν η πρώτη προτροπή του προγράμματος απευθύνεται στον παίκτη* `2`*, αυτό σημαίνει ότι η εναλλαγή του παίκτη γίνεται πριν την εμφάνιση της πρώτης προτροπής.*

Σε περίπτωση που χρησιμοποιήσατε την **elif** και όχι την **else** στο πρόγραμμα σας, αντικαταστήστε την με μια **else** διαγράφοντας τη συνθήκη και εκτελέστε το πρόγραμμα. Θα διαπιστώσετε ότι λειτουργεί και πάλι σωστά.

Είτε τη χρησιμοποιήσατε, είτε όχι, γιατί είναι περιττή η **elif** στο παράδειγμά μας;

Θα μπορούσε κάποιος να υλοποιήσει την εναλλαγή του παίκτη ως εξής:

```
if player == 1:
    player = 2
elif player == 2:
    player = 1
```

Σίγουρα ο κώδικας λειτουργεί σωστά. Ωστόσο, η `player` *μπορεί να έχει μόνο δύο πιθανές τιμές: είτε την* `1`*, είτε τη* `2`*. Η συνθήκη της* **if** *ελέγχει μήπως η τιμή της* `player` *είναι η* `1`*. Αν αυτό δεν ισχύει, τότε γνωρίζουμε ήδη ότι η τιμή της* `player` *είναι η* `2` *και είναι περιττό να ελέγξουμε κάποια επιπρόσθετη συνθήκη (όπως γίνεται με την* **elif***).*

11. Όταν η επανάληψη τελειώσει, η μεταβλητή `player` δείχνει ποιος παίκτης είχε σειρά να παίξει όταν τελείωσαν τα σπίρτα. Συνεπώς, μπορούμε να χρησιμοποιήσουμε τη μεταβλητή `player` για να διαπιστώσουμε ποιος παίκτης κέρδισε.

    Προσθέστε την κατάλληλη εντολή μετά την επανάληψη, ώστε να εμφανίζει τον αριθμό του παίκτη που κέρδισε. Για παράδειγμα:

    ```
    Παίκτη 2 κέρδισες!
    ```

    Όταν ολοκληρωθεί η επανάληψη, η τιμή της μεταβλητής `player` *δείχνει τον νικητή (αφού ο παίκτης που έπαιξε πριν από αυτόν πήρε τα τελευταία σπίρτα). Συνεπώς, θα προσθέσουμε μετά την επανάληψη (χωρίς εσοχή) την εντολή:*

    ```
    print("Παίκτη", player, "κέρδισες!")
    ```

    Τι θα συμβεί αν τοποθετήσουμε την παραπάνω εντολή μέσα στην **while**;

    Αν κατά λάθος στοιχίσουμε αυτή την τελευταία εντολή μαζί με τις προηγούμενες, τότε θα εκτελείται κι αυτή σε κάθε κύκλο της επανάληψης, ανακυρήσσοντας εναλλάξ και τους δύο παίκτες ως νικητές!

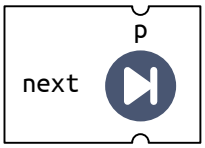
    Εκτελέστε το πρόγραμμά σας. Εμφανίζει σωστά τον αριθμό του παίκτη που κέρδισε;

    Ναι. Αν εκτελέσουμε το πρόγραμμα δύο φορές και φροντίσουμε ώστε την πρώτη να κερδίσει ο παίκτης `1` *και τη δεύτερη να κερδίσει ο παίκτης* `2`*, τότε θα δούμε ότι και στις δύο περιπτώσεις εμφανίζεται στο τέλος του παιχνιδιού το μήνυμα με τον σωστό αριθμό του παίκτη που νίκησε.*

12. Με τις εντολές που προσθέσαμε στο βήμα 10, η μεταβλητή `player` εναλλάσσεται σε κάθε γύρο μεταξύ των τιμών `1` και `2`.

Τώρα θα φτιάξουμε το πρώτο μας υποπρόγραμμα, το οποίο θα συντελεί στην υλοποίηση αυτής της λειτουργίας και θα μας βοηθήσει να "συμμαζέψουμε" λίγο τον κώδικα του κυρίως προγράμματος.

Το υποπρόγραμμα θα επιστρέφει τον αριθμό του παίκτη που παίζει στον επόμενο γύρο.



Η συνάρτηση `next` δέχεται σαν παράμετρο τον αριθμό `p` ενός παίκτη κι επιστρέφει τον αριθμό του παίκτη που παίζει μετά τον `p`.

Προσθέστε την παρακάτω εντολή που ορίζει τη συνάρτηση `next`, με παράμετρο τον αριθμό `p` ενός παίκτη.

```
def next(p):
```

Προσθέστε στη συνάρτηση τις κατάλληλες εντολές έτσι ώστε να ελέγχει τον αριθμό `p` του παίκτη και να επιστρέφει τον αριθμό του παίκτη που έχει σειρά να παίξει μετά από αυτόν.

Στο πρόγραμμά σας υπάρχουν ήδη οι εντολές που επιτελούν την παραπάνω λειτουργία, χρησιμοποιήστε ανάλογη προσέγγιση και στη συνάρτηση. Θυμηθείτε να υπάρχουν οι κατάλληλες εσοχές στις εντολές της συνάρτησης και να χρησιμοποιήσετε την εντολή **`return`**, προκειμένου να επιστρέψετε το αποτέλεσμα στο πρόγραμμα.

Θα συμπληρώσουμε τον ορισμό της συνάρτησης `next` ως εξής:

```
def next(p):
    if p == 1:
        return 2
    else:
        return 1
```

Η προσέγγιση είναι παρόμοια με κείνη που χρησιμοποιήσαμε στο κύριο πρόγραμμα. Εδώ όμως σκεφτόμαστε ανεξάρτητα από το κύριο πρόγραμμα. Εδώ εστιάζουμε μόνο στη λειτουργία που θέλουμε να υλοποιεί η συνάρτηση: με δεδομένο τον αριθμό `p` ενός παίκτη, ποιός είναι ο παίκτης που παίζει μετά από αυτόν; Με το υποπρόγραμμα δεν τροποποιείται απευθείας η τιμή της `player` αλλά επιστρέφεται η τιμή που θα πρέπει εκχωρηθεί στην `player` για να γίνει η εναλλαγή του παίκτη.

13. Μέσα στη **`while`**, *αντικαταστήστε* τις εντολές του προγράμματος που εναλλάσσουν τον αριθμό του παίκτη με την εντολή που ακολουθεί:

```
    player = next(player)
```

Η τιμή της μεταβλητής `player` είναι η τιμή που επιστρέφει η συνάρτηση `next`, δηλαδή ο αριθμός του επόμενου παίκτη.

Εκτελέστε το πρόγραμμα. Εμφανίζει σωστά τον αριθμό του παίκτη που παίζει σε κάθε γύρο;

Ναι, όπως συνέβαινε και πριν χρησιμοποιήσουμε τη `next`.

Παρατηρείτε κάποια διαφορά στη λειτουργία του προγράμματος;

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 λει-

τουργίας του.

14. Υπάρχουν κι άλλοι τρόποι να επιτευχθεί το ίδιο αποτέλεσμα, όπως φαίνεται στις εναλλακτικές υλοποιήσεις που ακολουθούν. Ωστόσο, ο τρόπος που λειτουργεί η συνάρτηση δεν έχει σημασία για εκείνους που την καλούν.

```
def next(p):
    return (p % 2) + 1
```

```
def next(p):
    return 3 - p
```

Η έκφραση `p%2` υπολογίζει το υπόλοιπο της ακέραιας διαίρεσης του `p` με το `2`.

Αντικαταστήστε τις εντολές της συνάρτησης `next` με όποια από τις παραπάνω εναλλακτικές υλοποιήσεις θέλετε.

Εκτελέστε ξανά το πρόγραμμα. Παρατηρείτε κάποια διαφορά στη λειτουργία του σε σχέση με πριν;

Όχι, και πάλι δεν υπάρχει καμία διαφορά στη λειτουργία του προγράμματος. Έχει όμως ενδιαφέρον ότι το κύριο πρόγραμμα δεν τροποποιήθηκε καθόλου, παρόλο που άλλαξε πλήρως ο τρόπος που λειτουργεί η `next`. Ουσιαστικά, για το κύριο πρόγραμμα δεν έχει σημασία με ποιον τρόπο λειτουργεί η `next`, αρκεί να επιστρέφει το αποτέλεσμα που πρέπει.

Για να κατανοήσετε καλύτερα τη λειτουργία των παραπάνω εκδοχών της συνάρτησης `next` συμπληρώστε τον πίνακα που ακολουθεί για τις τιμές `1` και `2` της παραμέτρου `p`.

πιθανές τιμές της παραμέτρου `p`	τιμές της έκφρασης `(p % 2) + 1`	τιμές της έκφρασης `3 - p`
`1`	`(1 % 2) + 1` → `1 + 1` → `2`	`3 - 1` → `2`
`2`	`(2 % 2) + 1` → `0 + 1` → `1`	`3 - 2` → `1`

Ποια από τις δύο υλοποιήσεις της συνάρτησης `next` επιλέξατε να χρησιμοποιήσετε στο πρόγραμμά σας; Τι σας ώθησε να κάνετε την επιλογή αυτή;

Επέλεξα τη δεύτερη γιατί υπολογίζει το αποτέλεσμα με μια απλή αφαίρεση.

Εντοπίζετε κάποιο πλεονέκτημα στις εναλλακτικές υλοποιήσεις σε σχέση με την αρχική;

Μου αρέσει το γεγονός ότι οι εναλλακτικές υλοποιήσεις υπολογίζουν το αποτέλεσμα απευθείας, χωρίς να χρειαστεί να ελεγχθεί η τιμή της παραμέτρου `p` με μια `if`. Ο τρόπος που λειτουργούν μοιάζει με "κόλπο".

Ποια ή ποιες από τις υλοποιήσεις της `next`, συμπεριλαμβανομένης της αρχικής, θεωρείτε ότι μπορεί να γενικευτεί εύκολα, ώστε η `next` να είναι επαναχρησιμοποιήσιμη σε παιχνίδια όπου συμμετέχουν περισσότεροι από δύο παίκτες;

Η γενίκευση της `next` ζητείται ως επέκταση στο αντίστοιχο κεφάλαιο. Μπορείτε να διαβάσετε την ακριβή εκφώνηση, αλλά και να συμβουλευτείτε την προτεινόμενη λύση.

Όταν συμμετέχουν, για παράδειγμα, τέσσερις παίκτες, τότε ο παίκτης που παίζει μετά τον `1` είναι ο `2`, μετά τον `2` ο `3`, μετά τον `3` ο `4` και μετά από αυτόν πάλι ο `1`.

Η αρχική υλοποίηση της `next` *με την* **`if`**–**`else`** *μπορεί να επεκταθεί και σε τέτοια παιχνίδα, αφού θα έλεγε κανείς ότι υπάρχουν και πάλι δύο περιπτώσεις: αν ο* `p` *δεν είναι ο τελευταίος παίκτης τότε ο επόμενος είναι ο* `p + 1`*. Σε διαφορετική περίπτωση, ο επόμενος παίκτης είναι ο* `1`*.*

Η υλοποίηση με το υπόλοιπο επίσης μπορεί να γενικευτεί σε περισσότερους παίκτες. Αντί να υπολογίζουμε το υπόλοιπο της διαίρεσης με το `2`*, αρκεί να υπολογίζουμε το υπόλοιπο της διαίρεσης με το πλήθος των παικτών.*

Αντιθέτως, η τρίτη υλοποίηση φαίνεται να "πηγαινοέρχεται" ανάμεσα σε δύο τιμές μόνο. Δεν γίνεται αυτή η προσέγγιση να γενικευτεί σε περισσότερους παίκτες.

Μη Λέμε κι Ό,τι Θέλουμε

Το πρόγραμμά μας επιτρέπει στον παίκτη που παίζει κάθε φορά ν' αφαιρέσει όσα σπίρτα θέλει. Θα επεκτείνουμε το πρόγραμμα, έτσι ώστε να ελέγχει τον αριθμό των σπίρτων που ζητά ν' αφαιρέσει ο παίκτης και να μην επιτρέπει κινήσεις που παραβιάζουν τους κανόνες του παιχνιδιού.

15. Για το σκοπό αυτό θα υλοποιήσουμε αρχικά μια συνάρτηση που θα δέχεται σαν παράμετρο το πλήθος `m` των σπίρτων που έχουν απομείνει και θα επιστρέφει το *μέγιστο* πλήθος σπίρτων που επιτρέπεται ν' αφαιρεθούν.

    Σύμφωνα με τους κανόνες του παιχνιδιού, ένας παίκτης επιτρέπεται να αφαιρέσει από ένα μέχρι και τρία σπίρτα κάθε φορά, αρκεί τα σπίρτα που έχουν απομείνει στο τραπέζι να είναι περισσότερα από τρία. Σε διαφορετική περίπτωση, θα πρέπει πάρει το πολύ όσα σπίρτα βρίσκονται στο τραπέζι.

    Προσθέστε την παρακάτω εντολή στο πρόγραμμά σας για να δηλώσετε τη συνάρτηση `maxMatches`.

```
def maxMatches(m):
```

    Χρησιμοποιήστε κατάλληλα την **`if`**–**`else`**, ώστε η συνάρτηση να επιστρέφει το μέγιστο αριθμό σπίρτων που μπορούν να αφαιρεθούν ανάλογα με τον αριθμό των σπίρτων `m` που βρίσκονται στο τραπέζι.

    Αν τα σπίρτα `m` *που απομένουν είναι περισσότερα από τρία, τότε το μέγιστο πλήθος που μπορεί ν' αφαιρεθεί είναι τρία. Σε διαφορετική περίπτωση, ένας παίκτης μπορεί να αφαιρέσει το πολύ όσα σπίρτα απομένουν. Συνεπώς, θα συμπληρώσουμε τη* `maxMatches` *ως εξής:*

```
def maxMatches(m):
    if m > 3:
        return 3
    else:
        return m
```

16. Προσθέστε τις παρακάτω εντολές αμέσως μετά την εντολή του βήματος 9 που διαβάζει τον αριθμό των σπίρτων που θέλει ν' αφαιρέσει ο παίκτης.

```
print("Παίκτη", player, "πόσα σπίρτα θέλεις;")
removed = int(input())
limit = maxMatches(matches)
if removed < 1 or removed > limit:
    print("Πάρε από 1 μέχρι και", limit, "σπίρτα")
```

Τι αποτέλεσμα πιστεύετε ότι θα έχει η προσθήκη των παραπάνω εντολών;

Η πρώτη εντολή καλεί την `maxMatches` *για να υπολογίσει το μέγιστο πλήθος σπίρτων που επιτρέπεται ν' αφαιρεθούν και ονομάζει την τιμή που επιστρέφεται* `limit`*. Η δεύτερη εντολή ελέγχει αν ο αριθμός σπίρτων που πληκτρολόγησε ο παίκτης είναι* έγκυρος*, δηλαδή τουλάχιστον ίσος με* `1` *και το πολύ ίσος με* `limit`*. Σε περίπτωση που ο αριθμός σπίρτων δεν είναι έγκυρος, τότε εμφανίζεται ένα σχετικό μήνυμα.*

Εκτελέστε το πρόγραμμα δίνοντας μια σειρά από λανθασμένες τιμές στον αριθμό των σπίρτων που αφαιρούνται. Εμφανίζεται το μήνυμα για τον μη-έγκυρο αριθμό σπίρτων;

Εκτελώντας το πρόγραμμα, ζητήσαμε να αφαιρέσουμε διαδοχικά `4` *και* `5` *σπίρτα. Και στις δύο περιπτώσεις εμφανίστηκε το μήνυμα.*

Σε περίπτωση που ο παίκτης δώσει μη-έγκυρη τιμή το πρόγραμμα την ξαναζητά ή συνεχίζει κανονικά τη λειτουργία του;

Όταν ο παίκτης δώσει μη-έγκυρη τιμή τότε το πρόγραμμα εμφανίζει σχετικό μήνυμα, όμως αμέσως μετά εμφανίζει και το μήνυμα για τα σπίρτα που απομένουν, όπου φαίνεται ότι το πλήθος των σπίρτων που ζητήθηκαν έχει αφαιρεθεί, παρόλο που δεν ήταν έγκυρο. Επομένως, παρά το μήνυμα περί μη-έγκυρης τιμής, το πρόγραμμα συνεχίζει κανονικά την εκτέλεση των εντολών της επανάληψης, χωρίς να ζητά εκ νέου από τον παίκτη να πληκτρολογήσει μια έγκυρη τιμή.

Γιατί πιστεύετε ότι συμβαίνει αυτό;

Ο κώδικας που προσθέσαμε κάνει απλά έναν έλεγχο (με την **if***). Δεν περιλαμβάνεται σε αυτόν τον κώδικα κάποια εντολή που να ξαναζητά από το χρήστη να πληκτρολογήσει τιμή, σε περίπτωση που η αρχική τιμή που πληκτρολογεί δεν είναι έγκυρη.*

Τι θα προτείνατε για να διορθωθεί η λειτουργία του προγράμματος;

Το πρόγραμμά μας ήδη περιλαμβάνει εντολές που ζητούν από το παίκτη να πληκτρολογήσει το πλήθος των σπίρτων που επιθυμεί να αφαιρέσει. Θα πρέπει λοιπόν αυτές τις εντολές να τις αντιγράψουμε και μέσα στην **if***, ώστε να ξαναζητείται το πλήθος των σπίρτων σε περίπτωση μη-έγκυρης τιμής.*

17. *Προσθέστε* εκ νέου *και* μέσα στην **if** τις εντολές του βήματος 9 που προτρέπουν τον παίκτη να δώσει αριθμό σπίρτων και ζητούν την

απάντησή του, ώστε όταν δίνει μη-έγκυρη απάντηση το πρόγραμμα να ξαναζητά τα σπίρτα που θέλει ο παίκτης να αφαιρέσει.

Με αυτή την προσθήκη, ο σχετικός κώδικας γίνεται πλέον:

```
print("Παίκτη", player, "πόσα σπίρτα θέλεις;")
removed = int(input())
limit = maxMatches(matches)
if removed < 1 or removed > limit:
    print("Πάρε από 1 μέχρι και", limit, "σπίρτα")
    print("Παίκτη", player, "πόσα σπίρτα θέλεις;")
    removed = int(input())
```

Εκτελέστε το πρόγραμμα δίνοντας διαδοχικά μια μη-έγκυρη τιμή και στη συνέχεια μια έγκυρη. Λειτουργεί σωστά;

Ναι! Αρχικά ζητήσαμε 4 *σπίρτα και εμφανίστηκε το μήνυμα περί μη-έγκυρης τιμής. Όμως, αντί το πρόγραμμα να προχωρήσει όπως πριν, τώρα ξαναζήτησε το πλήθος των σπίρτων που θα αφαιρεθούν. Ζητήσαμε* 3 *σπίρτα και το πρόγραμμα προχώρησε κανονικά, αφαιρώντας τα από το σύνολο.*

Εκτελέστε ξανά το πρόγραμμα δίνοντας διαδοχικά δύο μη-έγκυρες τιμές. Τι παρατηρείτε;

Αρχικά ζητήσαμε 4 *σπίρτα και εμφανίστηκε το μήνυμα περί μη-έγκυρης τιμής, χωρίς τα σπίρτα να αφαιρεθούν. Στη συνέχεια, ξαναζητήσαμε* 4 *σπίρτα και δυστυχώς τα σπίρτα αφαιρέθηκαν κανονικά από το σύνολο.*

18. Παρόλο που στην πρώτη μη-έγκυρη τιμή το πρόγραμμα εμφανίζει μήνυμα σφάλματος στον παίκτη και του ζητάει να δώσει ξανά την τιμή των σπίρτων, στην επόμενη του επιτρέπει να συνεχίσει με τον αριθμό σπίρτων που διάλεξε. Θα χρησιμοποιήσουμε τη **while**, προκειμένου το πρόγραμμα να ξαναζητά τον αριθμό των σπίρτων που θέλει ο παίκτης όσο η απάντηση του παραμένει εκτός ορίων.

   Τροποποιήστε την **if** και στη θέση της βάλτε την εντολή **while** ώστε να κάνει τον έλεγχο για την εγκυρότητα του αριθμού των σπίρτων, όπως παρακάτω:

   ```
   while removed < 1 or removed > limit:
   ```

   Εκτελέστε το πρόγραμμα και δώστε μια σειρά από λανθασμένες και έγκυρες τιμές. Λειτουργεί σωστά;

   Η **while** *"εγκλωβίζει" το χρήστη σε έναν κύκλο, ο οποίος διακόπτεται μόνο όταν δοθεί έγκυρη τιμή. Όσες μη-έγκυρες τιμές κι αν δώσει ο χρήστης, το πρόγραμμα δεν συνεχίζει μέχρι να δοθεί μια τιμή εντός των επιτρεπτών ορίων. Μόνο όταν γίνει αυτό, το πρόγραμμα συνεχίζει με τις εντολές που ακολουθούν, αφαιρώντας τα σπίρτα που ζητήθηκαν και εμφανίζοντας το μήνυμα για τον αριθμό των σπίρτων που απομένουν.*

19. Με τις εντολές που προσθέσαμε στα βήματα 16, 17 και 18, το πρόγραμμά μας δε διαβάζει απλά από τον παίκτη το πλήθος των σπίρ-

των που επιθυμεί ν' αφαιρέσει, αλλά επίσης *ελέγχει* την τιμή που δίνει ο παίκτης και *εξασφαλίζει* ότι αυτή δεν παραβιάζει τους κανόνες του παιχνιδιού. Τώρα θα κατασκευάσουμε ένα υποπρόγραμμα το οποίο υλοποιεί αυτή ακριβώς τη λειτουργία.

Προσθέστε την εντολή που ακολουθεί για να ορίσετε τη συνάρτηση `readMatches`, η οποία δέχεται ως παραμέτρους τον αριθμό `p` του παίκτη που έχει σειρά να παίξει και το πλήθος `m` των σπίρτων που απομένουν.

```
def readMatches(p,m):
```

Προσθέστε στη συνάρτηση τις κατάλληλες εντολές έτσι ώστε να εμφανίζει μήνυμα προτροπής προς τον παίκτη με αριθμό `p`, να ζητά από τον παίκτη τον αριθμό των σπίρτων που επιθυμεί να αφαιρέσει και, αφού εξασφαλίσει με τους απαραίτητους ελέγχους ότι ο αριθμός των σπίρτων είναι έγκυρος με βάση τα σπίρτα `m` που απομένουν, να επιστρέφει τον αριθμό αυτό.

Στην ουσία ο κώδικας που χρειάζεστε υπάρχει έτοιμος στο κύριο πρόγραμμα. Ακολουθήστε ανάλογη προσέγγιση και στη συνάρτηση, χρησιμοποιώντας τις "τοπικές" μεταβλητές `p` και `m`. Σε περίπτωση που δυσκολευτείτε μπορείτε να ανατρέξετε στα βήματα 16, 17 και 18.

Θυμηθείτε να υπάρχουν οι κατάλληλες εσοχές στις εντολές της συνάρτησης και να χρησιμοποιήσετε την εντολή **`return`**, προκειμένου να επιστρέψετε το αποτέλεσμα στο πρόγραμμα.

Ο ορισμός της συνάρτησης `readMatches` *θα συμπληρωθεί ως εξής:*

```
def readMatches(p,m):
    print("Παίκτη", p, "πόσα σπίρτα θέλεις;")
    r = int(input())
    limit = maxMatches(m)
    while r < 1 or r > limit:
        print("Πάρε από 1 μέχρι και", limit, "σπίρτα.")
        print("Παίκτη", p, "πόσα σπίρτα θέλεις;")
        r = int(input())
    return r
```

Η προσέγγιση είναι ουσιαστικά ίδια με εκείνη που χρησιμοποιήσαμε στο κύριο πρόγραμμα. Εδώ όμως εστιάζουμε μόνο στη λειτουργία που θέλουμε να υλοποιεί η συνάρτηση: με δεδομένο τον αριθμό `p` *ενός παίκτη, και τα σπίρτα* `m` *που απομένουν, ποιό είναι το (απαραίτητα έγκυρο) πλήθος σπίρτων που επιθυμεί να αφαιρέσει ο παίκτης* `p`*;*

20. Με τις εντολές που προσθέσαμε στα βήματα 16, 17 και 18, το πρόγραμμά μας εξασφαλίζει ότι η τιμή που θα πάρει η `removed` από τον παίκτη θα είναι συμβατή με τους κανόνες του παιχνιδιού. Τώρα διαθέτουμε για τον σκοπό αυτό την συνάρτηση `readMatches`.

Αντικαταστήστε όποιες εντολές του προγράμματος κρίνετε απαραίτητο με την κλήση της συνάρτησης, όπως φαίνεται παρακάτω.

```
removed = readMatches(player, matches)
```

Με τη χρήση της `readMatches` *για την ανάγνωση των σπίρτων που θα αφαιρεί σε κάθε κύκλο ο παίκτης, η βασική επανάληψη του παιχνιδιού παίρνει την παρακάτω μορφή:*

```
while matches > 0:
    removed = readMatches(player, matches)
    matches = matches - removed
    print("Σπίρτα που απομένουν:", matches)
    player = next(player)
```

Εκτελέστε το πρόγραμμα. Λειτουργεί σωστά; Παρατηρείτε κάποια διαφορά στη λειτουργία του;

Το πρόγραμμα λειτουργεί σωστά. Παρά τις εσωτερικές αλλαγές, ο χρήστης δεν αντιλαμβάνεται καμία απολύτως διαφορά στα μηνύματα που βλέπει και στον τρόπο που αλληλεπιδρά με το πρόγραμμα.

Παρατηρήστε την έκταση και την πολυπλοκότητα που έχει αυτή την στιγμή το κύριο πρόγραμμα. Πιστεύετε ότι η χρήση συναρτήσεων έχει βελτιώσει την *αναγνωσιμότητα* του προγράμματος;

Ένα πρόγραμμα είναι αναγνώσιμο όταν το διαβάζει κανείς και καταλαβαίνει πως λειτουργεί χωρίς να καταβάλει μεγάλη προσπάθεια.

Το κύριο πρόγραμμα εξακολουθεί να είναι μικρό και συμπαγές, όπως ήταν πριν ξεκινήσουμε το βήμα 15. Όλος ο κώδικας για τον έλεγχο εγκυρότητας έχει ενσωματωθεί μέσα στις συναρτήσεις που κατασκευάσαμε. Στο κύριο πρόγραμμα, η μόνη ουσιαστική αλλαγή που έχει γίνει είναι ότι η τιμή της `removed` *δεν διαβάζεται απευθείας από το χρήστη με την* `input`*, αλλά ελεγχόμενα, με τη* `readMatches`*.*

Χαζό Μηχάνημα

Θα τροποποιήσουμε τώρα το πρόγραμμα, ώστε να αναλαμβάνει το ρόλο ενός από τους δύο παίκτες. Δεν είναι ανάγκη να καταλήξουμε από την αρχή σε κάποια ευφυή στρατηγική για το πρόγραμμά μας.

21. Μπορείτε να προτείνετε έναν απλό - "χαζό" τρόπο για να επιλέγει το πρόγραμμα το πλήθος των σπίρτων που θα αφαιρεί όταν έχει σειρά να παίξει;

Όταν είμαστε αρχάριοι σ' ένα παιχνίδι, παίζουμε λίγο-πολύ τυχαία. Και σ' αυτή την περίπτωση, μπορεί το πρόγραμμα να επιλέγει τυχαία το πλήθος των σπίρτων που θα αφαιρεί κάθε φορά που έχει σειρά να παίξει.

Θα κατασκευάσουμε μια απλή συνάρτηση η οποία θα επιλέγει *τυχαία* το πλήθος των σπίρτων που θα αφαιρεί το πρόγραμμα, φροντίζοντας αυτό το πλήθος να μην υπερβαίνει το μέγιστο πλήθος σπίρτων που επιτρέπεται να αφαιρεθούν.

Πώς θα υπολογίζει η συνάρτηση τον μέγιστο πλήθος σπίρτων που μπορούν να αφαιρεθούν;

Θα χρησιμοποιηθεί η συνάρτηση `maxMatches`*, η οποία περιέχεται ήδη στο πρόγραμμα και υλοποιεί αυτή ακριβώς τη λειτουργία.*

Η `maxMatches` *αναπτύχθηκε για να χρησιμοποιηθεί στα πλαίσια του ελέγχου εγκυρότητας, αλλά αποδεικνύεται χρήσιμο "εξάρτημα" που επαναχρησιμοποιείται και σε διαφορετικό σημείο του προγράμματος, με διαφορετικό σκοπό.*

Προσθέστε στο πρόγραμμα την παρακάτω εντολή που ορίζει τη συνάρτηση `randomMatches`, με παράμετρο το πλήθος `m` των σπίρτων που απομένουν.

```
def randomMatches(m):
```

Προσθέστε στη συνάρτηση την κατάλληλη εντολή ή εντολές, προκειμένου να επιστρέφει στο πρόγραμμα έναν τυχαίο ακέραιο αριθμό από το 1 μέχρι και το μέγιστο επιτρεπτό αριθμό σπίρτων που μπορεί να αφαιρέσει ο παίκτης.

Ο ορισμός της συνάρτησης `randomMatches` *θα συμπληρωθεί ως εξής:*

```
def randomMatches(m):
    return random.randint(1,maxMatches(m))
```

Ίσως όμως έτσι το πρόγραμμα να "αυτοκτονεί", δηλαδή να αφαιρεί όλα τα σπίρτα ακόμα κι όταν έχει τη δυνατότητα να αφήσει τον αντίπαλό του με ένα (και να τον αναγκάσει να χάσει). Μια μικρή τροποποίηση:

```
def randomMatches(m):
    if m == 1:
        return 1
    elif m <= 4:
        return m - 1
    else:
        return random.randint(1,maxMatches(m))
```

Έτσι, όταν απομένει μόνο ένα σπίρτο, το πρόγραμμα αναγκάζεται να το πάρει και όταν τα σπίρτα είναι δύο, τρία ή τέσσερα, τότε το πρόγραμμα αφήνει τον αντίπαλο με το τελευταίο σπίρτο. Σε αυτές τις περιπτώσεις δεν γίνεται πια τυχαία επιλογή των σπίρτων που θ' αφαιρεθούν.

22. Ας εστιάσουμε τώρα στο κύριο πρόγραμμα. Θα χρησιμοποιήσουμε μια μεταβλητή `computer`, η οποία *στην αρχή του παιχνιδιού* θα παίρνει τυχαία την τιμή 1 ή 2. Η τιμή της `computer` υποδεικνύει τον αριθμό του παίκτη που παίζει αυτοματοποιημένα, δηλαδή το ρόλο του τον έχει αναλάβει το ίδιο το πρόγραμμα.

    Προσθέστε στο πρόγραμμα την κατάλληλη εντολή που θα επιλέγει τυχαία είτε την τιμή 1 είτε την τιμή 2 και θα την αποθηκεύει στη μεταβλητή `computer`.

    Πριν την αρχή της επανάληψης, θα προστεθεί η εντολή:

    ```
    computer = random.randint(1,2)
    ```

23. Θυμηθείτε ότι στο τέλος κάθε γύρου, η μεταβλητή `player` εναλλάσσεται μεταξύ των τιμών `1` και `2`, υποδεικνύοντας ποιος παίκτης έχει σειρά να παίξει στον επόμενο γύρο. Τώρα λοιπόν θα πρέπει να ελέγχουμε αν ο επόμενος παίκτης είναι ο άνθρωπος ή το πρόγραμμά μας, ώστε να καλέσουμε σε κάθε περίπτωση την ανάλογη συνάρτηση που θα μας επιστρέψει το πλήθος των σπίρτων που θα αφαιρεθούν.

    Χρησιμοποιήστε την **`if`**–**`else`**, ώστε η μεταβλητή `removed` να παίρνει την τιμή που επιστρέφει η συνάρτηση `randomMatches` όταν ο παίκτης που παίζει είναι ο υπολογιστής, και την τιμή που επιστρέφει η συνάρτηση `readMatches` σε διαφορετική περίπτωση. Για να ελέγξετε με την **`if`** ποιος έχει σειρά να παίξει, συγκρίνετε τις τιμές των μεταβλητών `player` και `computer`.

    Μέσα στην επανάληψη, στο σημείο όπου μέχρι πρότινος υπήρχε μόνο η κλήση της `readMatches`*, τώρα ο κώδικας επεκτείνεται ως εξής:*

```
    if player == computer:
        removed = randomMatches(matches)
    else:
        removed = readMatches(player, matches)
```

    Όταν οι μεταβλητές `player` *και* `computer` *έχουν την ίδια τιμή, αυτό σημαίνει πως είναι σειρά του προγράμματος να επιλέξει κίνηση κι έτσι καλείται η* `randomMatches` *για να υπολογιστεί το πλήθος των σπίρτων που πρέπει να αφαιρεθούν. Σε διαφορετική περίπτωση, επιλέγει κίνηση ο χρήστης, μέσω της* `readMatches`*, όπως γινόνταν και προηγουμένως.*

24. *Προσθέστε* την κατάλληλη εντολή, ώστε όταν παίζει ο υπολογιστής να εμφανίζεται το πλήθος των σπίρτων που αφαιρεί. Για παράδειγμα:

```
Ο υπολογιστής παίρνει 2 σπίρτα.
```

    Στην περίπτωση που παίζει το ίδιο το πρόγραμμα, προσθέτουμε την εντολή:

```
    if player == computer:
        removed = randomMatches(matches)
        print("Ο υπολογιστής παίρνει", removed, "σπίρτα.")
    else:
        removed = readMatches(player, matches)
```

25. Χρειάζεται να φροντίσουμε μια ακόμα σημαντική λεπτομέρεια. Μέχρι στιγμής, στο τέλος του παιχνιδιού ανακοινώνεται ο *αριθμός* του παίκτη που έχασε. Τώρα που μόνο ο ένας παίκτης είναι άνθρωπος, είναι προτιμότερο να εμφανίζουμε διαφοροποιημένα μηνύματα.

    Προσθέστε τις κατάλληλες εντολές στο πρόγραμμα, ώστε ο νικητής ν' ανακοινώνεται ως εξής: αν κερδίσει ο παίκτης, να ανακοινώνεται ο αριθμός του, όπως ακριβώς συνέβαινε μέχρι στιγμής. Αν κερδίσει το πρόγραμμα, να εμφανίζεται το μήνυμα:

```
Ο υπολογιστής κέρδισε.
```

Όπως έχουμε δει, η μεταβλητή `player` *υποδεικνύει στο τέλος του παιχνιδιού ποιος είναι ο νικητής. Αν η* `player` *έχει την ίδια τιμή με την* `computer`*, τότε νικητής είναι το πρόγραμμα.*

```
if player == computer:
    print("Ο υπολογιστής κέρδισε.")
else:
    print("Παίκτη", player, "κέρδισες!")
```

Εκτελέστε το πρόγραμμά σας δύο φορές. Φροντίστε τη μία φορά να κερδίσετε εσείς ως παίκτης, ενώ την άλλη ο υπολογιστής. Εμφανίζει το κατάλληλο μήνυμα σε κάθε περίπτωση;

Εκτελώντας το πρόγραμμα, μπορούμε να πάρουμε τα τελευταία σπίρτα, χάνοντας επίτηδες. Στην περίπτωση αυτή το πρόγραμμα εμφανίζει το σωστό μήνυμα: ότι ο υπολογιστής κέρδισε. Σε μια δεύτερη εκτέλεση, με προσεκτικές κινήσεις όταν τα σπίρτα είναι λίγα, δεν είναι ιδιαίτερα δύσκολο να αφήσουμε το πρόγραμμα με ένα σπίρτο, αναγκάζοντάς το να χάσει. Και πάλι, το πρόγραμμα εμφανίζει το σωστό μήνυμα.

Άνθρωπος Εναντίον Μηχανής

Όταν το πρόγραμμά μου παίζει τυχαία δεν έχει και μεγάλο ενδιαφέρον. Θα το κάνουμε λίγο πιο "έξυπνο".

26. Θα πρέπει πρώτα εμείς να σχεδιάσουμε έναν καλύτερο τρόπο παιχνιδιού και μετά να τον περιγράψουμε με τις κατάλληλες εντολές. Είναι ευκολότερο ν' αρχίσουμε μελετώντας ποιες είναι οι ενδεδειγμένες κινήσεις όταν απομένουν 2, 3 ή 4 σπίρτα: ο παίκτης που έχει σειρά να παίξει μπορεί να αφαιρέσει αντίστοιχα 1, 2 ή 3 σπίρτα και να κερδίσει άμεσα. Αντίθετα, όταν απομένουν 5 σπίρτα η κατάσταση είναι δύσκολη: όσα σπίρτα και να αφαιρέσει ο παίκτης που έχει σειρά να παίξει, η έκβαση είναι στα χέρια του αντιπάλου.

Ανάλογες δυσάρεστες καταστάσεις, που θα ονομάσουμε «ανεπιθύμητες νησίδες», αντιμετωπίζει ο παίκτης που έχει σειρά να παίξει όταν απομένουν 9, 13, 17, κ.ο.κ σπίρτα. Όσα σπίρτα κι αν αφαιρέσει, ο αντίπαλος μπορεί να τον στείλει στην επόμενη νησίδα.

Προκειμένου να κατανοήσετε καλύτερα τη στρατηγική που πρέπει να ακολουθήσει το πρόγραμμα (ή ο οποιοσδήποτε παίκτης) για να παίζει "εξυπνότερα" συμπληρώστε τον πίνακα που ακολουθεί. Στην πρώτη στήλη δίνεται ο αριθμός των σπίρτων. Στη δεύτερη στήλη θα συμπληρώσετε το υπόλοιπο της διαίρεσης του αριθμού των σπίρτων με το 4, ενώ στην τρίτη στήλη τον αριθμό των σπίρτων που πρέπει να αφαιρεθούν, προκειμένου να οδηγηθεί ο αντίπαλος σε μια "ανεπιθύμητη" νησίδα, σύμφωνα με την περιγραφή που προηγήθηκε.

σπίρτα `m`	`m%4`	κίνηση πόσα αφαιρούνται
6	2	1
7	3	2
8	0	3
9	1	αδιάφορο
10	2	1
11	3	2
12	0	3
13	1	αδιάφορο
14	2	1
15	3	2
16	0	3

Παρατηρείτε κάποια σχέση ανάμεσα στον αριθμό της δεύτερης και της τρίτης στήλης;

Οι δύο στήλες είναι άμεσα συνδεδεμένες. Η τιμή της παράστασης `m % 4` *μπορεί να καθορίσει αυτόματα πόσα σπίρτα πρέπει να αφαιρεθούν.*

Στις περιπτώσεις που ο αριθμός των σπίρτων που βρίσκονται στο τραπέζι είναι τέτοιος που δεν μπορεί να οδηγήσει τον παίκτη σε νίκη (για παράδειγμα το `13`), πώς θα μπορούσε να επιλέγει τα σπίρτα που θα αφαιρέσει;

Σε αυτές τις περιπτώσεις δεν έχει σημασία πόσα σπίρτα θα επιλέξει ο παίκτης. Θα μπορούσε απλά να επιλέγει τυχαία.

27. Με βάση τα παραπάνω, μπορούμε να υλοποιήσουμε ένα υποπρόγραμμα που δέχεται σαν παράμετρο το πλήθος `m` των σπίρτων που απομένουν και επιστρέφει το πλήθος των σπίρτων που πρέπει να αφαιρεθούν ώστε ο αντίπαλος να οδηγηθεί σε μια ανεπιθύμητη νησίδα. Στην περίπτωση που ο παίκτης βρίσκεται ήδη σε ανεπιθύμητη νησίδα, επιστρέφεται ένας τυχαίος αριθμός σπίρτων, καλώντας την `randomMatches`.

```
def computeMatches(m):
    mod = m % 4
    if mod == 0:
        return 3
    elif mod == 1:
        return randomMatches(m)
    elif mod == 2:
        return 1
    else:
        return 2
```

28. Στο κύριο πρόγραμμα, *αντικαταστήστε* την κλήση της συνάρτησης `randomMatches` που επιλέγει ένα τυχαίο πλήθος σπίρτων, με μια κλήση στην `computeMatches`, ώστε πλεόν το πρόγραμμα να επιλέγει το πλήθος των σπίρτων που αφαιρείται ακολουθώντας συγκεκριμένη στρατηγική.

Αντικαθιστώντας την κλήση της μιας συνάρτησης με την άλλη, ο κώδικας έχει ως εξής:

```
if player == computer:
    removed = computeMatches(matches)
```

Εκτελέστε το πρόγραμμά σας. Λειτουργεί σωστά;

Όταν το πρόγραμμα ξεκινά πρώτο, χωρίς ο αρχικός αριθμός των σπίρτων να αντιστοιχεί σε ανεπιθύμητη νησίδα, παίζει ακριβώς όπως προβλέπεται από τον πίνακα, οδηγεί τον αντίπαλο σε κάθε γύρο σε μια ανεπιθύμητη νησίδα και τελικά κερδίζει. Όταν το πρόγραμμα ξεκινά δεύτερο και του δίνεται η ευκαιρία να φέρει τον αντίπαλό του σε ανεπιθύμητη νησίδα τότε και πάλι παίζει όπως προβλέπεται από τον πίνακα και κερδίζει.

Υπάρχουν μόνο δύο περιπτώσεις να κερδίσει ένας παίκτης το πρόγραμμα, με δεδομένο ότι παίζει τέλεια σε κάθε κίνηση: αν ξεκινήσει πρώτος χωρίς να βρίσκεται σε ανεπιθύμητη νησίδα ή αν ξεκινήσει πρώτο το πρόγραμμα, ευρισκόμενο σε ανεπιθύμητη νησίδα.