Μαύρο φίδι που μας έφαγε ….

… δως μου μια σκάλα να σωθώ!

Και τι φίδι. Ολόκληρος πύθωνας! Προφανώς αναφερόμαστε στο γνωστό παιχνίδι από τα παιδικά μας χρόνια όπου δυο ή περισσότεροι παίχτες προσπαθούν να τερματίσουν πρώτοι σε μια πίστα 100 τετραγώνων. Κάθε παίχτης ρίχνει το ζάρι και μετακινεί το πιόνι του τόσες θέσεις όσες δείχνει η ζαριά. Ως γνωστόν υπάρχουν τετράγωνα με την αρχή σκάλας που μας πάνε πιο κοντά στον τερματισμό και τετράγωνα με το κεφάλι φιδιού που μας πάνε πιο κοντά στην αφετηρία! Για να κερδίσει κάποιος παίχτης πρέπει να φτάσει πρώτος ακριβώς στο τετράγωνο 100. Εάν η ζαριά του είναι μεγαλύτερη τότε μετακινείται προς τα πίσω τα αντίστοιχα τετράγωνα.

Έννοιες: δομές δεδομένων, υποπρογράμματα

Το ταμπλό του παιχνιδιού

Πως και που θα καταχωρήσω τα 100 τετράγωνα ώστε να δημιουργήσω το ταμπλό του παιχνιδιού;

Σαν πρώτη σκέψη η δημιουργία ενός πίνακα 100 θέσεων, δηλαδή μιας δομής δεδομένων, θα μας έλυνε το πρόβλημα. Οι δομές δεδομένων είναι αυτό που λέει το όνομά τους, δομές που μπορούν να κρατήσουν μαζί μερικά δεδομένα. Με άλλα λόγια χρησιμοποιούνται για να αποθηκεύουν δεδομένα που έχουν σχέση μεταξύ τους. Υπάρχουν τέσσερις δομές δεδομένων ενσωματωμένες στη Python, οι λίστες, οι πλειάδες, τα λεξικά και τα σύνολα.

Για την υλοποίηση του ταμπλό θα χρησιμοποιήσουμε μια λίστα όπου θα αποθηκεύονται τα 100 τετράγωνα.

```
#ορισμός κενής λίστας για την αποθήκευση των τετραγώνων του ταμπλό παιχνιδιού
board = [ ]
```

Η λίστα είναι κενή. Για την ώρα δεν χρειάζεται να ανησυχείτε για το περιεχόμενο της

Πως θα καθορίσω τα τετράγωνα που θα λειτουργούν ως σκάλα ή ως φιδάκι;

Για κάθε μια από τις σκάλες πρέπει να εισάγουμε το τετράγωνο έναρξης (αρχή της σκάλας) και το τετράγωνο τερματισμού (τέλος της σκάλας). Την αρχή της σκάλας θα την τοποθετήσουμε μεταξύ των τετραγώνων 7 – 87 (έτσι ώστε να μην πέσει κάποιος παίχτης σε σκάλα από την πρώτη ζαριά αλλά και για να υπάρχει χώρος για την τοποθέτηση της εάν

μπει στο ανώτατο όριο. Το τέλος της σκάλας θα βρίσκεται τουλάχιστον 5 τετράγωνα μετά την αρχή της. Για την δημιουργία της σκάλας μπορούμε να χρησιμοποιήσουμε την **random.randint()**

```
def createScale():
    #Υποπρόγραμμα για την δημιουργία μιας σκάλας. Η σκάλα μπορεί να ξεκινάει
     μεταξύ του 7ου και 87ου τετραγώνου
    #επιλέγουμε τυχαία την αρχή της σκάλας
    startOfScale = random.randint(7,87)
    #επιλέγουμε τυχαία το τέλος της σκάλας. Θέλουμε η σκάλα να μας πηγαίνει το
     λιγότερο 5 θέσεις μπροστά
    endOfScale = random.randint(startOfScale+5, 99)
    #ενημέρωση του ταμπλό  με τη θέση της σκάλας
    board[startOfScale-1] = endOfScale
```

Στο τέλος του υποπρογράμματος ενημερώνουμε την λίστα board με την θέση της σκάλας όπως ορίστηκε τυχαία από την random.randint().

Προσοχή!!!

Οι δείκτες στις λίστες της Python ξεκινούν από το 0. Επομένως, στο παράδειγμα μας έχουμε (πριν την εισαγωγή σκάλας ή φιδιού που αλλάζει τις τιμές)

board[0] = 1, board[1] = 2, ……., board[98] = 99, board[99] = 100

π.χ. εάν η συνάρτηση επέλεξε startOfScale = 15 και endOfScale = 32 τότε η αρχική λίστα του ταμπλό board = [….13, 14, 15, 16, 17,…..] θα γίνει board = [….13, 14, 32, 16, 17,…..]

Το τετράγωνο με τον αριθμό 15 βρίσκεται στην θέση 14 και σε αυτήν την θέση πρέπει να εισάγουμε την σκάλα που οδηγεί στο τετράγωνο με τον αριθμό 32

board[startOfScale – 1] = endOfScale δηλαδή board[15 – 1] = 32 ή board[14] = 32

Την ίδια διαδικασία θα ακολουθήσουμε για το φιδάκι, μόνο που θα έχουμε αντίστροφη πορεία. Το κεφάλι του φιδιού θα βρίσκεται σε μεγαλύτερο αριθμό τετραγώνου από την ουρά.

```
def createSnake():
    #Υποπρόγραμμα για την δημιουργία ενός φιδιού.
    #Επιλέγουμε τυχαία την αρχή (κεφάλι) του φιδιού. Το κεφάλι του φιδιού μπορεί
    να βρίσκεται μεταξύ του 11ου και 99ου τετραγώνου
    startOfSnake = random.randint(11,99)
    #Επιλέγουμε τυχαία το τέλος (ουρά) του φιδιού. Το φιδάκι θα έχει μεταβλητό
    μήκος μεγαλύτερο όμως από 5.
    endOfSnake = random.randint(2, startOfSnake - 5)
    #ενημέρωση του ταμπλό με τη θέση του φιδιού
    board[startOfSnake - 1] = endOfSnake
```

Μην ξεχάσετε να εισάγετε την βιβλιοθήκη random στην αρχή του προγράμματος.

Πως θα δημιουργήσω το ταμπλό του παιχνιδιού με τις 4 σκάλες και τα 4 φιδάκια;

Αρχικά θα "γεμίσουμε" την κενή λίστα board με τους αριθμούς από το 1 έως το 100. Η συνάρτηση **append()** εφαρμόζεται σε μια λίστα και προσθέτει ένα νέο στοιχείο στο τέλος αυτής. Έπειτα, με τη χρήση της επαναληπτικής δομής for θα καλέσουμε 4 φορές τις συναρτήσεις createSnake και createScale για την εισαγωγή των φιδιών και των σκαλών στην λίστα board.

```
def createBoard():
    #Υποπρόγραμμα για την δημιουργία του ταμπλό
    #Εισάγουμε στην λίστα board τους αριθμούς από το 1 εως το 100
    for i in range(100):
    board.append(i + 1)
    # προσθέτουμε 4 σκάλες και 4 φιδάκια στο ταμπλό του παιχνιδιού
    for i in range(4):
        createScale()
        createSnake()
    # Η τελική μορφή του ταμπλό του παιχνιδιού
    print("Το ταμπλό του παιχνιδιού","\n", board, "\n")
```

Πόσοι παίχτες θα παίξουν;

Θα δημιουργήσουμε ένα παιχνίδι δυο παιχτών. Θα δημιουργήσουμε δυο λίστες. Την players που θα περιέχει τους δυο παίχτες και την positionOfPlayer που θα αποθηκεύει τις τρέχουσες θέσεις των δυο παιχτών στον πίνακα.

```
#Ορισμός λίστας players για την αποθήκευση των παιχτών και λίστας positionOfPlayer που θα αποθηκεύει την τρέχουσα θέση του κάθε παίχτη. Έχουμε δυο παίχτες και η αρχική τους θέση είναι το τετράγωνο 1
players = [1,2]
positionOfPlayer = [1,1]
```

Διαδικασία παιχνιδιού

Έχουμε τους παίχτες, έχουμε το ταμπλό του παιχνιδιού, δεν μένει παρά να καθορίσουμε την διαδικασία του παιχνιδιού. Το παιχνίδι εξελίσσεται σε διαδοχικούς γύρους έως ότου κάποιος παίχτης φτάσει στο τετράγωνο με τον αριθμό 100.

```
def round():
  #ολοκλήρωση ενός γύρου για όλους τους παίχτες. Εμφάνιση των νεων θέσεων
   των παιχτών
```

Ένας ένας με την σειρά!

Με χρήση της επαναληπτικής δομής for οι παίχτες θα παίζουν ο εναλλάξ.

```
for i in range(2):
    print ("Παίχτη ", i+1, "ήρθε η σειρά σου. Ρίξε το ζάρι με ENTER")
    input()
```

Πως θα προσομοιώσουμε το ρίξιμο των ζαριών;

Θα χρησιμοποιήσουμε την randint() που παράγει τυχαίες τιμές εντός συγκεκριμένου εύρους που καθορίζουμε εμείς.

```
    #παραγωγή τυχαίας τιμής για το ζάρι
    dice = random.randint(1,6)
```

Η τιμή του ζαριού θα προστεθεί στην τρέχουσα θέση (την πρώτη φορά που θα ρίξει το ζάρι είναι το τετράγωνο 1) ώστε να μας δώσει την καινούρια θέση του παίχτη

```
    newPosition = positionOfPlayer[i] + dice
```

Μέτα από κάθε ρίψη του παίχτη ελέγχουμε εάν ξεπερνάει με την ζαριά του το τετράγωνο με τον αριθμό 100 ώστε να κινηθεί προς τα πίσω στο ταμπλό του παιχνιδιού.

```
    #ελέγχουμε την περίπτωση που κάποιος παίχτης με την ζαριά του ξεπερνάει το
     τετράγωνο με τον αριθμό 100
    if newPosition <=  100:
        positionOfPlayer[i] = board[newPosition - 1]
    else:
        #εαν ο παίχτης φτάσει στο 100 και του περισσεύουν κινήσεις από την ζαριά
        του τότε κινείται αντίστροφα στο ταμπλό
        outOfBoard = newPosition - 100
        positionOfPlayer[i] = board[newPosition - outOfBoard*2 - 1]
    print("Εφερες ", dice, "Η νέα θέση σου είναι το ", positionOfPlayer[i],"\n")
```

Με τη ολοκλήρωση του γύρου εκτυπώνουμε τις νέες θέσεις των παιχτών.

```
print("Οι νέες θέσεις των παιχτών είναι ", positionOfPlayer,"\n")
```

Ας παίξουμε

Το παιχνίδι ξεκινά καλώντας την συνάρτηση createBoard για την δημιουργία ενός νέου ταμπλό παιχνιδιού. Ορίζουμε μια Boolean μεταβλητή που θα λειτουργεί ως σημαία για τον τερματισμό του προγράμματος

```
#Κυριως πρόγραμμα
createBoard()
endOfGame = False
```

Όσο η μεταβλητή endOfGame είναι False εκτελείται έλεγχος για το εάν έχουμε νικητή ώστε διαφορετικά να προχωρήσουμε στον επόμενο γύρο. Αυτό γίνεται αναζητώντας την τιμή 100 στην λίστα positionOfPlayer που περιέχει την θέση των παιχτών.

```
while endOfGame == False:
```

```
item = 100
if item in positionOfPlayer:
  print("Έχουμε νικητή")
  for i in range(2):
    if positionOfPlayer[i] == 100:
      print("Συγχαρητήρια παίχτη ", i + 1, ".Κέρδισες!")
      endOfGame = True
else:
  round()
```

Τροποποιήσεις - Επεκτάσεις

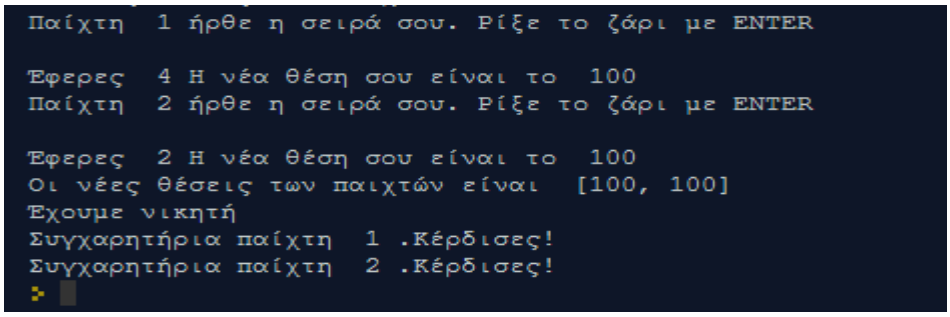
1) Στην αρχή του προγράμματος δημιουργήσαμε την λίστα players[1,2]. Η λίστα αυτή είναι απαραίτητη κατά την εκτέλεση του προγράμματος; Σε ποια / ες περιπτώσεις θα μας ήταν χρήσιμη;

2) Το πρόγραμμα μας πρέπει να καθορίζει την σειρά με την οποία θα παίζουν οι παίχτες. Αυτό θα μπορούσε να υλοποιηθεί με διάφορους τρόπους.
Ένας απλός τρόπος είναι να γίνεται μια τυχαία ταξινόμηση της λίστας players. Εναλλακτικά μπορούμε να ακολουθήσουμε τον κλασικό τρόπο που χρησιμοποιούμε στα επιτραπέζια παιχνίδια. Ο παίχτης που θα φέρει την μεγαλύτερη ζαριά κερδίζει Για το σκοπό αυτό όλοι οι παίκτες να ρίχνουν από μια ζαριά. Πρώτος θα είναι αυτός που θα πετύχει τη μεγαλύτερη ζαριά. Στη συνέχεια, οι υπόλοιποι οι παίκτες θα ακολουθούν κυκλικά. Για παράδειγμα, αν τη μεγαλύτερη ζαριά την πετύχει ο 2ος παίκτης και την δεύτερη μεγαλύτερη ο 3^{ος} τότε η σειρά θα είναι: 2, 3, 1, 2, 3, 1, κ.ο.κ. Θεωρήστε ότι στη φάση αυτή οι ζαριές των παικτών θα είναι διαφορετικές μεταξύ τους. Για την προσομοίωση κάθε ζαριάς θα χρησιμοποιήσουμε την συνάρτηση *random*

3) Με μια πιο προσεχτική ματιά στα υποπρογράμματα createScale και createSnake θα διαπιστώσουμε πως μπορούν να δημιουργηθούν μεγάλες σκάλες (12 → 99)και να οδηγήσουν έναν παίχτη γρήγορα στον τερματισμό ή μεγάλα φιδάκια (99 → 2) και φτου κι από την αρχή! Τροποποιήστε τα υποπρογράμματα createScale και createSnake έτσι ώστε να περιορίζεται το μέγεθος της σκάλας και του φιδιού αντίστοιχα.

4) Τροποποιήστε κατάλληλα το πρόγραμμα έτσι ώστε τα τετράγωνα έναρξης και προορισμού για την σκάλα και το φιδάκι να γίνονται αποδεκτά μόνο αν πρόκειται για απλά τετράγωνα, δηλαδή τετράγωνα που δεν περιέχουν ήδη κάποια άλλη σκάλα ή φιδάκι.

5) Σε μια από τις εκτελέσεις του προγράμματος βγήκε το παρακάτω αποτέλεσμα.



Κέρδισαν και οι δυο παίχτες!

Εάν παρατηρήσετε προσεκτικά το πρόγραμμα, θα διαπιστώσετε πως στο υποπρόγραμμα round() εάν κάποιος παίχτης τερματίσει, το πρόγραμμα δεν σταματά αλλά συνεχίζεται μέχρι να ολοκληρωθεί ο γύρος.

Προτείνετε μια λύση που να διορθώνει αυτό το πρόβλημα και να σταματά το πρόγραμμα όταν τερματίσει κάποιος από τους παίχτες.

Μια λίστα είναι μια δομή δεδομένων που συγκρατεί μια διατεταγμένη συλλογή στοιχείων, δηλαδή μπορείτε να αποθηκεύσετε μια ακολουθία (sequence) αντικειμένων στη λίστα. Αυτό είναι εύκολο να το φανταστείτε αν σκεφτείτε μια λίστα για αγορές, όπου έχετε μια λίστα με αντικείμενα που πρέπει να αγοράσετε, και εκτός αυτού πιθανόν έχετε κάθε στοιχείο σε διαφορετική γραμμή στη λίστα αγορών σας, ενώ στην Python τοποθετείτε κόμματα ανάμεσα στα στοιχεία. Η λίστα των στοιχείων πρέπει να κλείνεται σε αγκύλες (δηλαδή [και]) έτσι ώστε να καταλαβαίνει η Python ότι καθορίζετε μια λίστα. Αφού έχετε δημιουργήσει μια λίστα μια φορά μπορείτε να προσθέσετε, να μετακινήσετε ή να ψάξετε για στοιχεία σ' αυτή τη λίστα. Από τη στιγμή που μπορούμε να προσθέσουμε και να μετακινήσουμε στοιχεία, λέμε ότι η λίστα είναι ένας μεταβλητός τύπος δεδομένων (mutable data type), δηλαδή αυτός ο τύπος μπορεί να αλλαχθεί.

Η range() είναι μια ενσωματωμένη συνάρτηση της γλώσσας Python, η οποία, ανάμεσα σε άλλα, χρησιμοποιείται για την υπόδειξη του αριθμού των επαναλήψεων που θα εκτελεστούν σε ένα βρόχο. Η δομή της είναι της μορφής range(αρχή, μέχρι, βήμα), όπου *αρχή, μέχρι, βήμα* ακέραιοι αριθμοί. Οι ενδείξεις της *αρχής* και του *βήματος* δεν είναι υποχρεωτικές και, αν δεν αναφέρονται, θα αρχίσει από 0 και θα συνεχίσει με βήμα 1. Αντίθετα η ένδειξη *μέχρι* πρέπει πάντα να αναφέρεται.