Σημειωματάριο Τετάρτης 25 Οκτ. 2017

Ένα πρόγραμμα που παίζει κρεμάλα

Σήμερα φτιάξαμε ένα πρόγραμμα που παίζει "κρεμάλα" με το χρήστη. Το πρόγραμμα `hangman.py` link (http://fourier.math.uoc.gr/~mk/prog1718/files/hangman.py)) τρέχει στο command line και συνοδεύεται από ένα αρχείο λέξεων, το `words.txt` (link (http://fourier.math.uoc.gr/~mk/prog1718/files/words.txt)) που περιέχει μια λέξη (αγγλικά) σε κάθε γραμμή του. Για να το τρέξετε καταβάσετε το αρχείο `hangman.py` και το αρχείο `words.txt` στο ίδιο directory πηγαίνετε εκεί με το command line και δώστε την εντολή

```
python3 hangman.py
```

Μπορείτε επίσης να το τρέξετε στο repl.it (https://repl.it/NNPp/6).

Πώς διαβάζουμε από αρχεία κειμένου

Ένα από τα νέα πράγματα που μάθαμε σήμερα είναι το πώς να διαβάζουμε από αρχεία κειμένου. Ας υποθέσουμε ότι έχουμε στον τρέχοντα κατάλογο ένα αρχείο με το όνομα `keimeno.txt` (link (http://fourier.math.uoc.gr/~mk/prog1718/files/keimeno.txt))και με περιεχόμενα τα παρακάτω.

```
Αρνάκι άσπρο και
  παχύ,

της μάνας του
 καμάρι.
```

Το αρχείο αυτό, όπως και κάθα αρχείο που περιέχει κείμενο, το βλέπουμε σα μια ακολουθία από γραμμές, η κάθε μια από τις οποίες είναι ένα string. Για να διαβάσουμε τα περιεχόμενα ενός αρχείου πρέπει πρώτα να το ανοίξουμε με τη συνάρτηση `open`.

```
f = open("keimeno.txt", "r")
```

Το δεύτερο όρισμα της συνάρτησης `open` προσδιορίζει ότι ανοίγουμε το αρχείο για διάβασμα (reading) και όχι για να το γράψουμε ή να το τροποποιήσουμε. Η `open` επιστρέφει (κι εμείς το αποθηκεύουμε στη μεταβλητή `f`) ένα αντικείμενο μέσω του οποίου έχουμε πρόσβαση στο αρχείο. Δεν είναι κάποιου τύπου από τους συνηθισμένους (int, float, string) αλλά κάτι που συνήθως ονομάζεται file handle ή file descriptor στον προγραμματισμό. Είναι μέσω της μεταβλητής `f` που έχουμε πρόσβαση στο αρχείο από δω και στο εξής.

Με τις επόμενες εντολές ανοίγουμε το αρχείο και με τη μέθοδο `.readlines()` διαβάζουμε όλες τις γραμμές του σε μια λίστα L από strings. Κλείνουμε έπειτα το αρχείο με τη μέθοδο `f.close()` (το νόημα αυτής της πράξης είναι ότι οι όποιες αλλαγές έχουν γίνει στο αρχείο πηγαίνουν και γράφονται στο δίσκο) και μετά τυπώνουμε τη λίστα L που φτιάξαμε από τα περιεχόμενα του αρχείου.

Παρατηρείστε ότι κάθε στοιχείο του L (που αντιστοιχεί σε μια γραμμή του αρχείου) τελειώνει με τον ειδικό χαρακτήρα `\n` (newline) ο οποίος σηματοδοτεί το τέλος της γραμμής. Οι δύο τελευταίες γραμμές του αρχείου δεν έχουν κανένα χαρακτήρα πέρα από το χαρακτήρα newline (όταν γράφαμε το αρχείο με τον editor απλά πατήσαμε σκέτο enter και αλλάξαμε γραμμή).

In [1]:
```
f = open("keimeno.txt", "r")
L = f.readlines()
f.close()
print(L)
```

```
['Αρνάκι άσπρο και\n', '  παχύ,\n', '\n', 'της μάνας του\n', ' καμάρι.\n', '\n', '\n']
```

Με τη μέθοδο `.strip()` ενός string μπορούμε και "καθαρίζουμε" το string από "λευκούς" χαρακτήρες (κενά, newlines, tabs, και ίσως ένα δύο άλλα ακόμη που δε φαίνονται). Όσοι λευκοί χαρακτήρες είναι στην αρχή ή στο τέλος του string διαγράφονται και η μέθοδος `.strip()` επιστρέφει το string "καθαρό" (τουλάχιστον στην αρχή και στο τέλος του). Φτιάχνουμε έτσι μια καινούργια λίστα LL από την L με τα στοιχεία της L καθαρισμένα από κενά και τυπώνουμε αυτά. Βλέπουμε ότι τώρα δεν υπάρχει οι χαρακτήρες newline.

In [2]:
```
f = open("keimeno.txt", "r")
L = f.readlines()
LL = []
for s in L:
    LL.append(s.strip())
f.close()
print(LL)
```

```
['Αρνάκι άσπρο και', 'παχύ,', '', 'της μάνας του', 'καμάρι.', '', '']
```

Ευκαιρία είναι τώρα να δούμε το μηχανισμό list comprehension, με τον οποίο μπορούμε να φτιάχνουμε μια λίστα από μια άλλη με πολύ σύντομο και ευανάγνωστο τρόπο. Το for loop του προηγούμενου κώδικα και η αρχικοποίηση της LL έχουν εδώ αντικατασταθεί από τη γραμμή

```
LL = [s.strip() for s in L]
```

In [3]:
```
f = open("keimeno.txt", "r")
L = f.readlines()
LL = [s.strip() for s in L]
f.close()
print(LL)
```

```
['Αρνάκι άσπρο και', 'παχύ,', '', 'της μάνας του', 'καμάρι.', '', '']
```

Ακόμη πιο περιεκτικά θα μπορούσαμε να το έχουμε γράψει ως εξής:

In [4]:
```
f = open("keimeno.txt", "r")
L = [s.strip() for s in f.readlines()]
f.close()
print(L)
```

```
['Αρνάκι άσπρο και', 'παχύ,', '', 'της μάνας του', 'καμάρι.', '', '']
```

Το πολύ περιεκτικό και πυκνό δεν είναι πάντα το καλύτερο (ή το γρηγορότερο στο να τρέξει), αλλά είναι συνήθως καλύτερο επειδή είναι ευανάγνωστο.

Το πρόγραμμα hangman.py

In [ ]:
```
# Εισάγουμε τις βιβλιοθήκες (modules ονομάζονται συνήθως στην python)
# os (για operating system) και random (για τυχαιότητα)
import os, random

penalty = 0 # Αρχική ποινή του παίκτη

used=[] # Ποια γράμματα έχει χρησιμοποιήσει ο παίκτης μέχρι στιγμής.

filename="words.txt" # Σε ποιο αρχείο είναι αποθηκευμένες οι λέξεις που χρησιμοποιούμε.

# Στην παρακάτω μεταβλητή έχουμε αποθηκεύσει σε μια λίστα από strings τα οριζόντια κομμάτια
# της κρεμάλας μαζί με τον κρεμασμένο.
# θα ζωγραφίζουμε κάποια κομμάτια από αυτό το σχήμα, ανάλογα με το ποια είναι η ποινή του παίκτη.
#
# Παρατηρείστε το διπλό backslash (\) στο τέλος κάποιων γραμμών. Επειδή το backslash παίζει κάποιο
# ειδικό ρόλο στα strings (σκεφτείτε π.χ. πώς προσδιορίζουμε το χαρακτήρα newline με \n) ο μόνος
```

```
# τρόπος για να προσδιορίσουμε το backslash ως ένα χαρακτήρα το string μας είναι να το γράψουμε διπλό.
# Έτσι αν τυπώσετε ένα string με διπλό backslash θα εμφανιστεί μόνο ένα. Αν π.χ. δώσετε print("abcd\\")
# θα τυπωθεί abc\
parts = [
            '-----|',
            '|    o',
            '|    i',
            '|   / \\',
            '|    |',
            '|   / \\',
            '|  d   b'
        ] # Τα επίπεδα κάτω από το οριζόντιο ξύλο της κρεμάλας, που πάντα το τυπώνουμε, είναι 6.
          # Γι' αυτό και το maxpenalty παρακάτω βγαίνει 5. Όταν η ποινή γίνει 6 ο παίκτης έχει χάσει.

maxpenalty = len(parts)-2 # Μέγιστη επιτρεπτή ποινή. Με ένα πόντο παραπάνω χάνει το παιχνίδι. Όσα είναι τ
α
                          # στοιχεία της λίστας parts μείον 1.

# Εδώ διαβάζουμε όλες τις λέξεις από το αρχείο μας, στη λίστα WL. Αμέσως καθαρίζουμε όλα τα strings από λ
ευκούς
# χαρακτήρες με τη μέθοδο .strip() και τα μετατρέπουμε όλα σε μικρά γράμματα με τη μέθοδο .lower()
f = open(filename, "r")
WL = [s.strip().lower() for s in f.readlines()]
f.close()

# Επιλέγουμε ένα τυχαίο στοιχείο της WL με τη συνάρτηση random.choice(). Αυτή θα είναι η λέξη μας.
theword = random.choice(WL)
# Στη μεταβλητή show κρατάμε τη λέξη με τη μορφή που τη δείχνουμε στον παίκτη. Αρχικά θα είναι όλο παύλε
ς, όσες
# και το μήκος της λέξης theword.
show = len(theword)*"-"

# Αυτή είναι η βασική ανακύκλωση του προγράμματος που τρέχει συνεχώς έως ότου διακοπεί με break (όταν ο π
αίκτης
# χάσει ή όταν κερδίσει).

while True:
    os.system('clear') # Καλούμε την εντολή του λειτουργικού συστήματος 'clear' η οποία καθαρίζει το term
inal.
                       # Το ποια είναι αυτή η εντολή εξαρτάται από το σε ποιο λειτουργικό σύστημα δουλεύο
υμε.
                       # Στο linux η εντολή clear καθαρίζει το τερματικό οπότε καλούμε αυτήν. Στα Windows
```

```
 (όταν
                        # το πρόγραμμά μας τρέχει στη γραμμή εντολών των Windows η αντίστοιχη εντολή είναι
 μάλλον η 'cls')

    print("************************************************************")
    print() # Τυπώνει μια κενή γραμμή. Για αισθητικούς λόγους.
    print(parts[0]) # Τυπώνει το πρώτο string της λίστας parts (οριζόντιο ξύλο της κρεμάλας). Τυπώνεται π
άντα.
    for i in range(penalty): # Τυπώνουμε τόσα μετέπειτα στάδια της κρεμάλας όσοι και βαθμοί ποινής
        print(parts[i+1])
    for i in range(penalty+1, len(parts)): # Από τα υπόλοιπα στάδιο της κρεμάλας τυπώνουμε μόνο τον πρώτο
        print(parts[i][0]) # χαρακτήρα (κατακόρυφο ξύλο)

    if penalty > maxpenalty: # Ελέγχουμε αν η ποινή έχει ξεπεράσει την επιτρεπτή. Αν ναι ο παίκτης έχασε.
        print()
        print("Έχασες. Η λέξη ήταν {} και έφτασες μέχρι την {}".format(theword, show))
        break # τελειώνει το while άρα και το πρόγραμμα

    # Μετά την κρεμάλα δείχνουμε στο χρήστη πληροφορίες για το παιχνίδι.
    print()
    print("Η λέξη είναι: {}".format(show)) # Δείχνουμε την (ημι-)κρυμμένη λέξη
    print()
    print("Έχεις χρησιμοποιήσει τα γράμματα: {}".format(used)) # Δείχνουμε επίσης το ποια γράμματα έχει χ
ρησιμοποιήσει ήδη
    print()
    print("Η ποινή σου είναι {}. Μέγιστη ποινή είναι {}".format(penalty, maxpenalty)) # Δείχνουμε την ποι
νή του παίκτη.
    print()
    print()
    c = input("--> Δώσε το επόμενο γράμμα (και πάτα enter): ")
    c = c.lower() # Διαβάζουμε ένα string από τον παίκτη και το κάνουμε σε μικρά γράμματα

    if len(c)>1: # Αν έδωσε πάνω από 1 γράμμα ξαναπάμε στην αρχή
        print()
        input("--> Δώσε μόνο ένα γράμμα. Πάτα enter.")
        continue

    if c in used: # Αν έδωσε ήδη χρησιμοποιημένο γράμμα ξαναπάμε στην αρχή
        print()
        input("--> Το έχεις ήδη χρησιμοποιήσει. Πάτα enter.")
        continue
    else:
```

```
    used.append(c) # Αν νέο γράμμα το προσθέτουμε στη λίστα used.

# Αν το γράμμα δεν είναι στη λέξη τότε αυξάνουμε την ποινή κατά 1
if c not in theword:
    penalty += 1

# Εδώ φτιάχνουμε τη νέα μορφή της λέξης show
s=""
for i in range(len(theword)):
    if c==theword[i]: # Αν το γράμμα που έδωσε είναι στη θέση αυτή τότε βάζουμε αυτό στη show
        s += c
    else: # αλλιώς αφήνουμε στη show ό,τι είχε εκεί.
        s += show[i]
show=s

# Αν η λέξη show έχει γίνει ίδια με την theword τότε ο παίκτης κέρδισε.
if show == theword:
    print()
    print("Η λέξη ήταν [{}].".format(theword))
    print("Κέρδισες.")
    break
```