Python i Pygame

Programiranje arkadnih igrica u Pythonu koristeći Pygame

Pretraživanje

Čitanje iz datoteke

- Program koji čita liniju po liniju iz datoteke i ispisuje na ekran

```
file = open("bad_guys.txt")
for line in file:
    line = line.strip()
    print(line)
file.close()
```

File je objekt koji predstavlja sve linije iz datoteke
Metoda strip() uklanja razmake i entere na kraju svakog reda
close() zatvara datoteku

- Program koji čita liniju po liniju iz datoteke i sprema u niz

```
file = open("bad_guys.txt")
name_list = []
for line in file:
    line = line.strip()
    name_list.append(line)
file.close()
```

Linearno pretraživanje

- Prolazi se kroz cijelu listu od početka do kraja tražeći zadani niz
- Program koji čita liniju po liniju iz datoteke i u pročitanim linijama traži zadani niz znakova

```
file = open("bad_guys.txt")
name_list = []
for line in file:
    line = line.strip()
    name_list.append(line)
file.close()

# Linearno pretrazivanje
key = "Morgiana the Shrew"
i = 0
while i < len(name_list) and name_list[i] != key:
    i += 1
if i < len(name_list):
    print( "The name is at position", i)
else:
    print( "The name was not in the list." )
```

Program ispisuje broj reda u kojem se nalazi traženi niz znakova

Binarno pretraživanje

- Ispituje se sredina liste i ovisno o ishodu ispitivanje se nastavlja na jednom od dva dijela liste
- Program koji čita liniju po liniju iz datoteke i u pročitanim linijama binarnim pretraživanjem traži zadani niz znakova

```
file = open("bad_guys.txt")
name_list = []
for line in file:
    line = line.strip()
    name_list.append(line)
file.close()
# Binarno pretrazivanje
key = "Morgiana the Shrew"
donja_granica = 0
gornja_granica = len(name_list)-1
found = False
# Ponavljaj dok ne pronadjes ili dodjes do donje / gornje granice
while donja_granica <= gornja_granica and not found:
    # Pronadji sredinu
    middle_pos = (donja_granica + gornja_granica) // 2
    # Odluci da li povisujemo donju granicu
    # ili snizujemo gornju granicu
    # ili smo pronasli
    if name_list[middle_pos] < key:
        donja_granica = middle_pos + 1
    elif name_list[middle_pos] > key:
        gornja_granica = middle_pos - 1
    else:
        found = True
if found:
    print( "The name is at position", middle_pos)
else:
    print( "The name was not in the list." )
```

Program ispisuje broj reda u kojem se nalazi traženi niz znakova

Kviz

- Slijedi link ispod:

  http://programarcadegames.com/quiz/quiz.php?file=search&lang=en

Kviz odgovori

- P1: Before reading from a file a program must:
  - Reset the file with a reset command.
  - Initialize it with the init command.
  - Open it with the open command.

- P2: In order to read in each line of the file a program should:
  - Use a for loop.
  - Access each line line in an array.
  - Use the read_all_lines command.

Kviz odgovori, nastavak

- P3: What will happen if a program fails to close a file after reading it?
  - The file will be be marked as busy and will be inaccessible until the program ends.
  - Nothing, it doesn't really matter if you don't close the file.
  - The file will not be able to be used until the computer restarts.
  - The programmer will get a call from his mother reminding him that he forgot to close his that file, again.

- P4: What processing usually needs to be done on a line after it has been read in?
  - The dilithium crystals must be recalibrated before use.
  - The line needs to be converted to uppercase.
  - The carriage return and/or line feed need to be stripped of the end of the line.
  - The line needs to be converted from a string of integers to a string of letters.

Kviz odgovori, nastavak

- P5: What is wrong with this linear search?

```
i = 0
while my_list[i] != key and i < len(my_list):
    i += 1
```

  - The second check should be <= not <.
  - The first check should be == not !=.
  - The second check should be > not <.
  - The loop needs to check to see if we ran out of list items before checking to see if the item is e

- P6: After this code runs, what is the proper way to tell if the item was found or not?

```
i = 0
while i < len(my_list) and my_list[i] != key:
    i += 1
```

  - if my_list[i] == key:
  - if i > len(my_list):
  - if my_list[i] != key:
  - if i == len(my_list):

Kviz odgovori, nastavak

- P7: A binary search starts looking:
  - At the end of the list.
  - At a random spot in the list.
  - In the middle of the list.
  - At the beginning of the list.

- P8: Using a binary search, a list of 128 elements takes at most how many times through the search loop?
  - 1
  - 128
  - 7
  - 64

Kviz odgovori, nastavak

- P9: In a binary search, how do you know if an element is not in the list and the search should stop?
  - When i is greater than or equal to the list length.
  - When the key is not equal to the middle element.
  - The lower bound is equal or greater than the upper bound.
  - After every item has been checked.

- P10: If the key is less than the middle element, the search should:
  - Move the lower bound up to the middle element.
  - Move the upper bound down to the middle element.