Fej vagy írás

2015. október

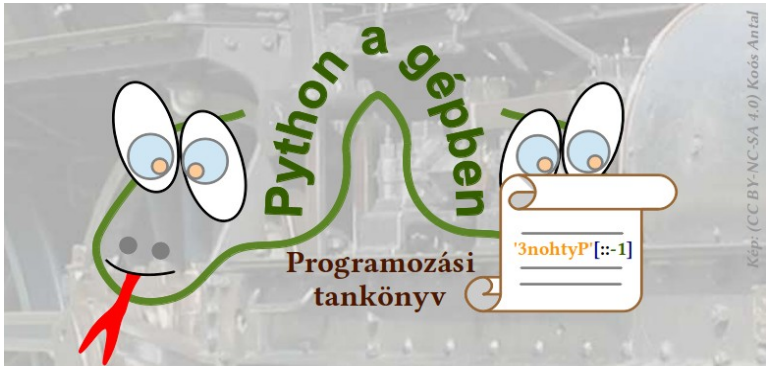
Az emelt szintű informatikai érettségi programozási feladatának megoldása Python 3 nyelven

A feladat leírása az Oktatási Hivatal honlapján: www.oktatas.hu/kozneveles/erettsegi/feladatsorok

Ajánlott olvasmányok:

A Python 3 dokumentációja angolul: docs.python.org

A „Python a gépben” c. könyv: eutlantis.k2os.hu/books



További feladatok: eutlantis.k2os.hu/infoktatas/erettsegi

```
#!/usr/bin/env python3
#-*- coding:utf-8 -*-
# fejvagyiras.py

import random
random.seed()

#--- 1. feladat ---
print("\n1. feladat")
print("A pénzfeldobás eredménye:",random.choice("IF") )

#--- 2. feladat ---
print("\n2. feladat")
tipp=input("Tippeljen! (F/I)= ").strip()

dobás= random.choice("IF")
print("A tipp", tipp+", a dobás eredménye", dobás, "volt.")
if dobás==tipp:
     print("Ön eltalálta.")
else:
     print("Ön nem találta el.")

#--- 3. feladat ---
print("\n3. feladat")
dobások=0
fejek=0
with open("kiserlet.txt") as ff:
     for sor in ff:
          if sor.strip()=="I":
               dobások+=1
          else:
               dobások+=1
               fejek+=1

print("A kísérlet", dobások,"dobásból állt.")
```

Az értelmezőnek megadjuk, hogy a programfájlban milyen kódolású karaktereket használunk. Az UTF-8 lehetővé teszi az ékezetes karakterek használatát is.

Importáljuk a véletlenszámok használatához szükséges modult, majd a *seed()* metódussal inicializáljuk.

A *choice()* metódus az argumentumként megadott objektum egy elemét választja ki véletlenszerűen.

A bekért karakterlánc elejéről és végéről levágjuk a *whitespace*-ket (szóközt, tabulátort stb.-t) a *strip()*-pel, hogy a z összehasonlításnál ne vezessenek félre.

A feladatkiírás szerint az adatfájlt úgy kell feldolgozni, hogy nem tároljuk el belső objektumban. A továbbiakban az adatokat majd többször is újra ki kell olvasni, amihez a fájl olvasási mutatóját mindig a fájl elejére kell állítani. Ez elérhető úgy is, hogy a fájlt lezárjuk, majd újra megnyitjuk. Ezt segíti, hogy a *with-as* blokkból való kilépés után a fájl automatikusan lezárásra kerül.

A beolvasott sor végéről le kell vágnunk a soremelés jelét, amit a *strip()*-pel teszünk meg és ami egyébként az összes *whitespace*-t eltávolítja a sor mindkét végéről.

```
#--- 4. feladat ---
print("\n4. feladat")
print("A kísérlet során a fej relatív gyakorisága {0:.2f}% volt.".format(fejek/dobások*100) )

#--- 5. feladat ---
print("\n5. feladat")
fejszám=0
sorozatok=0
with open("kiserlet.txt") as ff:
     for sor in ff:
          if sor.strip()=="F":
               fejszám+=1
          else:
               if fejszám==2:
                    sorozatok+=1
               fejszám=0

if fejszám==2:
     sorozatok+=1

print("A kísérlet során",sorozatok,"alkalommal dobtak pontosan két fejet egymás után.")

#--- 6. feladat ---
print("\n6. feladat")
hossz=0
kezdet=None
fejszám=0
with open("kiserlet.txt") as ff:
     for index,sor in enumerate(ff):
          if sor.strip()=="F":
               fejszám+=1
          else:
               if hossz< fejszám:
                    hossz, kezdet = fejszám, index-fejszám
               fejszám=0
```

A *{ 0:.2f }* minta szerint a *format()* metódus első (azaz 0. indexű) argumentumát lebegőpontos számként, két tizedesjegy pontossággal kérjük kiírni. A minta után szereplő % jel nem formázó utasítás, hanem csak közönséges, kiírandó karakter.

Számoljuk a közvetlenül egymás után álló fejeket, és amikor egy ilyen sorozat megszakad, akkor a *sorozatok* változó értékét eggyel megnöveljük, ha a fejek száma pontosan kettő volt. Így bármilyen hosszúságú sorozatokat is megszámlálhatunk.

Egy fejsorozat megszakad, ha a következő sor nem „F”-et tartalmaz, vagy ha a fájl végére értünk, ezért a ciklus után is meg kell vizsgálni a *fejszám* értékét.

A feladat az előzőhöz hasonló, de most a leghosszabb fejsorozatot keressük. Több ilyen is lehet, de a feladat kiírása szerint elég egyetlennek megadni a fájlbeli kezdő sorszámát. Az *enumerate()* függvény együtt szolgáltatja a sor indexét és a sor tartalmát (általában egy objektum soron következő elemét és annak objektumbeli indexét).

```
if hossz<fejszám:
     hossz, kezdet = fejszám, index-fejszám

print("A leghosszabb tisztafej sorozat {} tagból áll, kezdete a(z) {}. dobás.".format( hossz,kezdet+1))

#--- 7. feladat ---
#print("\n7. feladat")
négyesek=[]
f3f=0
f3í=0
for i in range(1000):
     eset=""
     for k in range(4):
          eset+= random.choice("FI")
     négyesek.append( eset)
     if eset=="FFFF":
          f3f+=1
     elif eset=="FFFI":
          f3í+=1

with open("dobasok.txt","w") as ff:
     ff.write("FFFF: {}, FFFI: {}\n".format(f3f,f3í))
     for eset in négyesek:
          ff.write( eset+" ")

#-----------------------------------------------------------------------
# További feladatok: http://sites.google.com/site/eutlantis/erettsegi
# Ajánlott olvasmány: www.interkonyv.hu/konyvek/koos_antal_python_a_gepben
```

Írásra nyitottuk meg a fájlt ('w').

A *write()* metódus nem ír ki automatikusan egy soremelő jelet is, így az első kiírásnál ezt meg kell adnunk a feladat követelménye szerint.

A fenti kód megfelel az érettségi vizsga követelményeinek, de nem az egyetlen ilyen, számos más megvalósítás is elképzelhető. A feladatkiírások általában közlik, hogy az adatfájl hibátlanságát és az input() által beolvasott adatok megfelelőségét nem kell ellenőrizni, a fenti program ennek figyelembevételével készült. Az alábbiakban gyakorlásképpen bemutatunk néhány részletet egy továbbfejlesztett változatból.

```python
# extra_fejvagyiras.py   részletek

#--- 2. feladat ---
print("\n2. feladat")
írásminta=list("IiÍí")
fejminta=list("Ff")
tipp=input("Tippeljen! (F/I)= ").strip()
if tipp in írásminta:
     tipp="I"
elif tipp in fejminta:
     tipp="F"
else:
     print("Hibás tipp!")
     exit(1)

dobás= random.choice("IF")
print("A tipp {}, a dobás eredménye {} volt.".format(tipp,dobás) )
print("Ön eltalálta." if dobás==tipp else "Ön nem találta el.")

#--- 3. feladat ---
print("\n3. feladat")
dobások=0
fejek=0
with open("kiserlet.txt") as ff:
     for sor in ff:
          sor=sor.strip()
          if sor in írásminta:
               dobások+=1
          elif sor in fejminta:
               dobások+=1
               fejek+=1

print("A kísérlet {} dobásból állt.".format(dobások) )
```

Elfogadjuk az I- és F-betű különféle változatait is az *input()* eredményeképpen.

A „*c=a* ***if*** *kifejezés* ***else*** *b”* feltételes értékadást alkalmazzuk, de most nincs szükségünk az eredménynek egy („c”) változóba történő eltárolására.

Az adatfájlban is elfogadjuk az I- és F-betű különféle változatait. Vegyük észre, hogy az esetleges üres sorok itt nem okoznak problémát.

```
#--- 6. feladat ---
print("\n6. feladat")
hossz=0
kezdet=None     # a leghosszabb fejsorozat kezdő indexe
kezdőfej=None   # az éppen számolt fejsorozat kezdő indexe
fejszám=0

with open("kiserlet.txt") as ff:
     for index,sor in enumerate(ff):
          sor=sor.strip()
          if sor in fejminta:
               fejszám+=1
               if fejszám==1:
                    kezdőfej=index
          elif sor in írásminta:
               if hossz< fejszám:
                    hossz, kezdet = fejszám, kezdőfej
               fejszám=0

if hossz<fejszám:
     hossz, kezdet = fejszám, kezdőfej

print("A leghosszabb tisztafej sorozat {} tagból áll, kezdete a(z) {}. sor.".format( hossz,kezdet+1))

#--- 7. feladat ---
print("\n7. feladat")
négyesek=[]
dnégy=dict()
for i in range(1000):
     eset=""
     for k in range(4):
          eset+= random.choice("FI")
     négyesek.append(eset)
     dnégy[eset]= dnégy.get(eset,0)+1
```

Ismét elfogadjuk az I- és F-betű különféle változatait és az üres sorokat, de az utóbbiak miatt módosítani kellett az eredeti progamrészt, mert a dobások száma már nem egyezik meg a sorok számával.

„dobás” helyett „sor”-t kell írnunk

Nemcsak a három vagy négy fejet tartalmazó sorozatokat számoljuk meg, hanem az összeset (16 félét). Ezt egy szótár segítségével tesszük, amiben a kulcs a sorozat mintája lesz karakterláncként, az érték pedig az előfordulási száma. Egy sorozat legelső előfordulásakor a *dnégy[eset]= dnégy[eset]+1* kifejezés jobb oldalán lévő hivatkozás rendkívüli eseményt okozna, hiszen még nincs ilyen kulcs a szótárban. Ezt elkerülhetjük a get() metódussal, amelyik ilyenkor a második argumentumot adja vissza és „elnyeli” a rendkívüli eseményt.

```
with open("dobasok.txt","w") as ff:
     ff.write("FFFF: {}, FFFI: {}\n".format(dnégy["FFFF"],dnégy["FFFI"]))
     for eset in négyesek:
          ff.write( eset+" ")

def hasonlítandó( eset_darab):
     eset,darab= eset_darab
     return eset

rendnégy=sorted( dnégy.items(), key= hasonlítandó )
#rendnégy=sorted( dnégy.items(),key=lambda eset_db: eset_db[0] )

for eset in rendnégy:
     print(eset[0]+": ",eset[1])
```

A szótárban gyűjtött adatokat szeretnénk sorba rendezve kiírni, de nem az értékek, hanem a kulcsok szerint. A rendezést a *sorted()* függvénnyel végezzük, aminek a nevesített *key* argumentumaként megadjuk, hogy mi szerint kell rendeznie. Ezt egy olyan függvény címének a megadásával tudjuk megtenni, amely visszatérési értékként a rendezésnél figyelembeveendő értéket, illetve objektumot eredményezi. A függvényt most a „*hasonlítandó*” névvel definiáltuk; bemeneti adatként a szótárból (kulcs,érték), azaz (esetminta,darabszám) párosítású *tuple*-kat fog kapni. Ezeket a kételemű objektumokat a szótár *items()* metódusával tudjuk előállítani.

A külön függvénydefiníció helyett használhatunk ún. *lambda* függvényt is.