Szkriptnyelvek
– 2. ZH –

2021. május 13.

Feladatok

1. (5 pont) Adott egy szöveges állomány, ami jelszavakat tartalmaz. Egy-egy sorban fel van tüntetve egy szabály is, aminek az adott jelszónak meg kell felelnie. A feladatunk az lesz, hogy megállapítsuk, hogy az állományban hány szabályos (valid) jelszó szerepel. Egy jelszó akkor szabályos, ha a hozzá tartozó szabálynak megfelel.

   Tegyük fel, hogy az állomány a következő sorokat tartalmazza:

   ```
   1-3 a: abcde
   1-3 b: cdefg
   2-9 c: ccccccccc
   ```

   A kettőspont előtt a szabály, utána pedig a jelszó szerepel.

   A szabály eleje a jelszóban két pozíciót határoz meg, ahol az 1 a jelszó első karakterét jelenti, a 2 a jelszó második karakterét jelenti, stb. A kettőspont előtt szereplő betű csakis a megadott két pozíció egyikén szerepelhet. Ha mind a két pozíción szerepel, akkor a jelszó nem szabályos. Az adott betű más pozíciókon is szerepelhet, de azokkal nem kell foglalkozni.

   Vagyis az `1-3 a` szabály jelentése: a jelszóban az `a` betűnek vagy az 1-es pozícióban, vagy a 3-as pozícióban kell szerepelnie (de mindkét helyen nem szerepelhet).

   Nézzük a fenti három példát:

   - `1-3 a: abcde` → szabályos, mivel az `a` betű szerepel az 1-es pozícióban, de nem szerepel a 3-as pozícióban
   - `1-3 b: cdefg` → nem szabályos, mivel a `b` betű sem az 1-es pozícióban, sem a 3-as pozícióban nem szerepel
   - `2-9 c: ccccccccc` → nem szabályos, mivel a `c` betű mind a 2-es pozícióban, mind a 9-es pozícióban szerepel

   Kérdés: hány darab szabályos jelszó szerepel a `passwords.txt` állományban?

   Futási példa:

   ```
   $ ./feladatX.py
   42
   ```

   A „42” helyett természetesen a helyes eredményt kell kiírni a képernyőre.

2. (3 pont) Írjon programot, ami parancssori argumentumként egy sztringet vár. A program írja ki a képernyőre a sztring módosított változatát.

   Ha a felhasználó nem ad meg egyetlen sztringet sem, vagy ha egynél több parancssori argumentumot ad meg, akkor írjon ki egy hibaüzenetet, s a program álljon le.

   A módosított sztringben minden számjegynek duplán kell szerepelnie. Példa:

```
$ ./feladatX.py
Hibás paraméterezés! Egyetlen sztringet kell megadni!

$ ./feladatX.py egy ketto
Hibás paraméterezés! Egyetlen sztringet kell megadni!

$ ./feladatX.py hello
hello

$ ./feladatX.py he9llo
he99llo

$ ./feladatX.py 123world456
112233world445566

$ ./feladatX.py a1b2c33d
a11b22c3333d
```

   A sztring átalakítását egy függvénnyel végezze, melynek a deklarációja:

```
def double_digits(text):
```

   A függvény megkapja az eredeti sztringet, majd visszaadja a módosított sztringet.

   A forráskódot lássa el típusannotációkkal! Ellenőrizze le, hogy nincs-e hiba:

```
$ make mypy
```

3. (5 pont) Írjon programot, ami feldolgozza a `numbers.csv` fájl tartalmát s kiír a képernyőre egy kis statisztikát.

   Tudjuk, hogy az az angol nyelvben a lebegőpontos számoknál tizedespontot, míg a magyarban tizedesvesszőt kell használni.

   A `numbers.csv` fájl minden sora a következőképpen néz ki (példa):

```
12.63543;6,8475;7.34534;.,.,.,
2.674;3.7365;19.863432;kjdfg,5654
12,674;13,7365;29,863432;a.b@c.de
72.674;123.7365;1119.863432;aa,bb
```

   Az első három „oszlopban” valós számok szerepelnek, míg a 4. oszlopban tetszőleges szöveg szerepelhet. Az egyes oszlopokat [;] választja el egymástól. A 4. oszloppal nem kell foglalkozni. A 4. oszlopban bármilyen karakter szerepelhet a [;] kivételével. A lényeg a három valós szám, viszont ezek magyarul (tizedesvesszővel) vagy angolul (tizedesponttal) is szerepelhetnek. A fenti példa első sorában egy angol, egy magyar, majd még egy angol szám szerepel.

   Az egyszerűség kedvéért egy valós számot angol számnak nevezünk, ha az angol helyesírás szerint van megadva. Ugyanígy beszélhetünk magyar számról is.

   Írjunk programot, ami elemzi a fájl tartalmát, s kiírja, hogy hány darab angol és magyar szám szerepel az input állományban. Továbbá számítsuk ki külön-külön az angol és magyar számok összegét.

   A programot a fenti példára futtatva a következő kimenetet kell kapnunk:

```
$ ./feladatX.py
A fájlban 8 db angol és 4 db magyar szám található.
Angol számok összege: 1362.53
Magyar számok összege: 63.12
```

   Az összegeket 2 tizedesjegy pontossággal írja ki! Egy kis segítség:

```
>>> round(3.14159, 4)
3.1416
```

   Figyelem! A programnak a `numbers.csv` fájlt kell feldolgoznia!

4. (5 pont) Írjon programot, ami elemzi a `szoveg.txt` nevű szöveges állományt. A fájlban szavak szerepelnek, egymástól szóközzel elválasztva.

   A program egy olyan kimenetet produkáljon, amiben a palindróm szavak változatlan formában jelennek meg, a többi szó viszont át van húzva. Az áthúzást úgy oldja meg, hogy egy n hosszúságú szó helyén n db `X` jelet jelenít meg (pl. `abcd` helyett az `XXXX` karaktersorozatot kell kiírni).

   Példa egy kisebb méretű input fájl esetén:

```
$ cat rovid.txt
ufx ormc vt gui pdw qwzzzwq
zujojuz vuv uujsq vv
sopu taa ttr ahevj hvd
aobiciboa aeiea wse os
hius qeet poiu yrnmq

$ ./feladatX.py
XXX XXXX XX XXX XXX qwzzzwq
zujojuz vuv XXXXX vv
XXXX XXX XXX XXXXX XXX
aobiciboa aeiea XXX XX
XXXX XXXX XXXX XXXXX
```

   A kimenet ugyanúgy nézzen ki, mint a bemenet, vagyis a kimenet i. sorában ugyanazok a szavak szerepeljenek, mint az input állomány i. sorában. A szavak sorrendje se változzon meg! Az egyetlen különbség csak annyi legyen, hogy a nem-palindróm szavak legyenek kitakarva.

   Figyelem! A programnak a `szoveg.txt` állományt kell feldolgoznia!

Oldalszám:	1	2	3	4	Összesen
Max. pontszám:	5	3	5	5	18
Pontszám:					