Tartalomjegyzék

2. gyakorlat - Függvényhívás, hibajavítás

Ezen a gyakorlaton is még f?leg az el?z? féléves Informatika 1 anyagát ismételjük, az ott tanultakat kell használni. Amire biztosan szükség lesz ma:

- Python alapjai, futtatása; karakterláncok; elágazások, ciklusok; függvények definiálása: Informatika1-2014/eloadas3
- Listák, szótárak: Informatika1-2014/eloadas4

Ha valaki nem hallgatta ezt a tárgyat, és nincs tisztában a python alapjaival, nyugodtan kérdezzen laboron, és igyekezzen egy-két héten belül behozni a lemaradást.

Feladatok

Feladatok a CloudCoder-en megtalálhatóak. Ajánlott sorrend:

1. kiertekel
2. kozel
3. vananagram

A **vananagram** feladathoz az eredeti kód itt van, ha valaki véletlenül eltüntette volna a rendszerben:

```
# Ez a függvény megszámolja egy szóban hogy melyik
# betü hányszor szerepel, és egy szótárban visszaadja
def betuszamolo(szo):
    szamolo = {}
    for betu in szo:
        # Ha már volt ilyen betü, növeljük a számát
        if betu in szamolo:
            szamolo[betu] += 1
        # Ha még nem volt ilyen betü, most már egyszer volt
        else:
            szamolo[betu] = 1
    return szamolo

# Ez a függvény megmondja hogy a szavak listájában van-e
# anagramma pár
def vananagram(l):
```

```
    # Ez a két for ciklus végigmegy minden i, j páron
    # hogy i < j < len(l)
    for i in range(len(l)):
        for j in range(i+1, len(l)):
            # Megszámoljuk mindkettöben a betüket, és
            # összehasonlítjuk a kettöt
            i_szamolo = betuszamolo(l[i])
            j_szamolo = betuszamolo(l[i])
            if i_szamolo == j_szamolo:
                return True
    # Ha nem volt anagramma, False-al térünk vissza
    return False
```

CloudCoder használata

A legtöbb python feladathoz gyakorlaton egy CloudCodernek nevezett rendszert fogunk használni. Ennek elõnye, hogy helyben ki is javítja a feladatot. Elérés:

- https://ccweb.math.bme.hu/cloudcoder/

Részletes leírás a tárgylapon.

Házi Feladat

A **pasziansz** feladathoz az eredeti kód itt van, ha valaki véletlenül eltüntette volna a rendszerben:

```
# Megmondja egy kártyáról hogy fekete-e
def fekete(kartya):
    szam, szin = kartya
    fekete_szinek = ["pikk", "treff"]
    if szin in fekete_szinek:
        return True
    else:
        return False

# Ez a függvény megmondja a legjobb kirakható
# sor értékét.
def pasziansz(kez):
    # Ha egy kártyánk sincs, 0 pont a max.
    if len(kez) == 0:
        return 0

    # Csökkenö sorrendbe tesszük a kártyákat.
    kez.sort(reverse=True)

    # Az els? kártya a legértékesebb.
    elozo = kez[0]
    pont = elozo[0]
    del kez[0]

    # Mindig megkeressük a legértékesebb lapot,
    # ami lehet a következö, mert más szinü.
    # Ezt addig csináljuk amig találunk ilyent
    talalt = True
    while talalt:
        talalt = False
        # Csökken? sorrendben végigmegyünk a kártyákon
        for i in range(len(kez)):
            lap = kez[i]
            # Ha a kártya különbözö szinü, betesszük,
```

```
            # és kitöröljük a kártyák listájáról
            if fekete(elozo) != fekete(lap):
                talalt = True
                elozo = lap
                pont += lap[0]
                del kez[i]
                break
    return pont
```

Feladatok megoldása

kiertekel

```
def kiertekel(cel, l):
    osszpont = 0
    cel_x, cel_y = cel
    for lo_x, lo_y in l:
        osszpont += pontertek(lo_x - cel_x, lo_y - cel_y)
    return osszpont
```

kozel

```
def skalar_szorzat(a, b):
    szorzat = 0
    for i in range(len(a)):
        szorzat += a[i] * b[i]
    return szorzat

def kozel(l):
    max_szorzat = skalar_szorzat(l[0], l[1])
    for i in range(len(l)):
        for j in range(i+1, len(l)):
            max_szorzat = max(max_szorzat, skalar_szorzat(l[i], l[j]))
    return max_szorzat
```

vananagram

Az a hiba, hogy a 24. sorban **l[j]** helyett **l[i]** van írva.

Házi feladatok megoldása

madarnyelv_dekod

```
def madarnyelv_dekod(szo, szotar):
    for szotar_szo in szotar:
        if szo == madarra(szotar_szo):
            return szotar_szo
    return None
```

pasziansz

Akkor van baj, ha valamelyik színb?l több van mint a másikból, de a legértékesebb lap nem ebb?l a színb?l van, mert akkor a másik színnel kezdjük a tevést, és így eggyel kevesebbet tudunk tenni ebb?l a színb?l, mint lehetne. Egy lehetséges megoldás, hogy miután leraktam az eredeti módszerrel minden kártyát amit lehetett, ha a maradék kártyák más szín?ek mint az els? kártya, akkor még az els? kártya elé tehetek bel?lük egyet, a legértékesebbet. Ebben a megoldásban a *return* el?tti 3 sor a módosítás:

```
# Ez a függvény megmondja a legjobb kirakható
# sor értékét.
def pasziansz(kez):
    # Ha egy kártyánk sincs, 0 pont a max.
    if len(kez) == 0:
        return 0

    # Csökkenö sorrendbe tesszük a kártyákat.
    kez.sort(reverse=True)

    # Az els? kártya a legértékesebb.
    elso = kez[0]
    elozo = kez[0]
    pont = elozo[0]
    del kez[0]

    # Mindig megkeressük a legértékesebb lapot,
    # ami lehet a következö, mert más szinü.
    # Ezt addig csináljuk amig találunk ilyent
    talalt = True
    while talalt:
        talalt = False
        # Csökken? sorrendben végigmegyünk a kártyákon
        for i in range(len(kez)):
            lap = kez[i]
            # Ha a kártya különbözö szinü, betesszük,
            # és kitöröljük a kártyák listájáról
            if fekete(elozo) != fekete(lap):
                talalt = True
                elozo = lap
                pont += lap[0]
                del kez[i]
                break

    # Ha a maradek kartyakbol egy meg a sor elejere
    # teheto, tegyuk oda.
    if len(kez) != 0:
        if fekete(elso) != fekete(kez[0]):
            pont += kez[0][0]

    return pont
```