Tartalomjegyzék

5. gyakorlat - Referenciak, Objektumok

Ezen a gyakorlaton az el?z? heti el?adás anyagát dolgozzuk fel.

Feladatok

Feladatok a CloudCoder-en megtalálhatóak. Ajánlott sorrend:

1. osztok
2. evek
3. unnep_datum
4. docstringek
5. ponttabla

A **ponttabla** feladat ugyan kicsit bonyolultabb, mint amiket itt általában laboron feladok, de jobban próbál egy valós problémához hasonlítani ami veletek is el?fordulhat. Úgy gondolom olyan szinten kéne körülbelül most lennetek, hogy meg tudjátok azt oldani, ha nem is 10-20 perc alatt, de néhány óra alatt. Tananyag szempontjából nincsen semmi új benne, de a megírt programok bonyolultságában is kell fejl?dni.

Feladatok megoldása

osztok

```
import pprint

def osztok(szamok, printer):
    oszto_szotar = {}
    for szam in szamok:
        l = []
        for i in range(1, szam+1):
            if szam % i == 0:
                l.append(i)
        oszto_szotar[szam] = l

    return printer.pformat(oszto_szotar)
```

evek

```
import datetime
def evek(szulinapok):
    d = {}
    for szulinap in szulinapok:
        ev = szulinap.year
        if ev in d:
            d[ev] += 1
        else:
            d[ev] = 1
    return d
```

unnep_datum

```
import datetime

def unnep_datum(nev):
    if nev == "forradalom":
        return datetime.date(1848, 3, 15)
    elif nev == "allamalapitas":
        return datetime.date(1083, 8, 20)
    elif nev == "otvenhat":
        return datetime.date(1956, 10, 23)
```

docstringek

```
def jo_docstring(szoveg):
    if not szoveg:
        return False
    if not szoveg[0].isupper():
        return False
    if szoveg.find('.') == -1 or szoveg.find('.') < len(szoveg)-1:
        return False
    return True

def docstringek(javaslatok):
    return filter(jo_docstring, javaslatok)
```

Alternatív megoldás lambdával:

```
def docstringek(javaslatok):
    return filter(lambda s : s and s[0].isupper() and s.find('.') == len(s) - 1, javaslatok)
```

ponttabla

```
def ponttabla(tablazat):
    print tablazat
    sorok = tablazat.split("\n")
    nevek = sorok[0].split("|")[1:]
    n = len(nevek)
    for i in range(n):
        nevek[i] = nevek[i].strip()

    pontok = [0 for i in range(n)]
    for sor in sorok[1:]:
        elemek = sor.split("|")
        for i in range(n):
            pontok[i] += int(elemek[i+1])
```

```
    pont_szotar = {}
    for i in range(n):
        pont_szotar[nevek[i]] = pontok[i]
    return pont_szotar
```

Házi feladatok megoldása

kodolt_datum

```
import datetime

def kodolt_datum(kod):
    dekodolas = {'e' : 1, 'k' : 2, 'h' : 3, 'n' : 4, 'o' : 5, 'a' : 6, 't' : 7, 'y' : 8, 'i' : 9,
    ev = 1000 * dekodolas[kod[0]] + 100 * dekodolas[kod[1]] + 10 * dekodolas[kod[2]] + dekodolas[k
    honap = 10 * dekodolas[kod[4]] + dekodolas[kod[5]]
    nap = 10 * dekodolas[kod[6]] + dekodolas[kod[7]]
    return datetime.date(ev, honap, nap)
```

nev_neptun

```
def nev_neptun(neptunok):
    return sorted(neptunok.values())
```

Vagy ha a lista *sort()* metódusát szeretnénk használni akkor:

```
def nev_neptun(neptunok):
    kodok = neptunok.values()
    kodok.sort()
    return kodok
```