Sąrašų generavimas (*list comprehension*)

Sąrašų "suvokimas" 🤔

VILNIUS CODING SCHOOL

List comprehension - sąrašų "suvokimas" (??🤔)

Norint sugeneruoti sąrašą, galima naudotis ciklais (`for` su `.append()`), tačiau, jei reikia atlikti nesudėtingus veiksmus sąrašo generavime, tai nebūtina naudoti ciklą su `.append()` - yra glaustesnis užrašymas:

[išraiška for elementas in sąrašas]

Pavyzdžiai

```
# sąrašo užpildymas:
# su ciklu for:
a = []
for i in range(0, 10):
    a.append(round(i*3.14, 6))
# su list comprehension:
a = [round(i*3.14,6) for i in range(0,10)]
```

Naudojantis lambda, išpakavimu

VILNIUS CODING SCHOOL

```
# naudojantis lamba, map ir list:
a = list(map(lambda x : round(x*3.14, 6), range(0,10)))
print(a)
b = [*range(0,10)] #naudojantis išpakavimu
print(b)
c = [(lambda x: round(x*3.14, 6))(x) for x in range(0, 10)]
print(c)
```

👁 c sąrašo generavime panaudotas lambda funkcijos iškvietimas pateikiant lambda funkcijai argumentą x, skliausteliuose. Palyginimui:

```
arg = lambda x : x**2 # suteikiamas pavadinimas
print(arg(2)) #iškviečiama it įprasta funkcija
print( (lambda x: x**2)(2))
```

VILNIUS CODING SCHOOL

if sakiniai

Sąrašų generavime galima panaudoti ir `if` sakinį:

```
[išraiškaA for kintamasis in list if sąlygaA]
```

```
a = [x*2 for x in range(0,10) if x%2 == 0]
print(a)
```

Jei norima turėti `else` atitikmenį, struktūra šiek tiek keičiasi:

```
[išraiškaA if sąlygaA else išraiškaB for kintamasis
in sąrašas]
```

```
b = [x*2 if x%2==0 else x**2 for x in range(0,10)]
print(b)
```

Funkcijos funkcijose, rekursija ir dokumentavimas

Funkcijos funkcijose

Python'e funkcijos gali savyje turėti kitas, aprašytas funkcijas (*inested or inner functions*). Vienas iš tikslų taip daryti yra **enkapsuliacija** (*encapsulation*), kai norima vidinę funkciją apsaugoti ar paslėpti nuo to, kas vyksta pagrindiniame kode.

```
def out(x):
    if x < 0: #sąlygų tikrinimas, jei reikia
        raise ValueError("Argumentas turi būti daugiau nei nulis")
    def pw(arg):#vidinė funkcija
        return arg**x
    return pw #grąžinama vidinė funkcija
pw_two = out(2)
pw_three = out(3)
print(pw_two(3))
print(pw_three(3))
pw_err = out(-2)
```

👁 Čia iliustruojama ir tai, kad galima turėti kelias *inner* funkcijos instancijas su skirtingais *outer* funkcijos parametrais.

Rekursinės funkcijos

👁 *rekursinė funkcija*, arba *rekursija* - tai, kai funkcija pati save iškviečia. Naudojama tada, kai skaičiavimų rezultatas priklauso nuo ankstesnių veiksmų rezultatų. Klasikinis pavyzdys - *Fibonacci* seka, arba faktorialo skaičiavimas ($n!$):

```
def factorial(x):
    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))
num = 3
print( num, 'faktorialas yra', factorial(num))
```

VILNIUS CODING SCHOOL

Kitas pavyzdys - *Fibonacci* seka

$$
\begin{aligned}
F_0 &= 0 \\
F_1 &= 1 \\
F_n &= F_{n-1} + F_{n-2};\ n > 1
\end{aligned}
\tag{2}
$$

- Parašykite funkciją Fibonacci sekos n-tojo nario skaičiavimui: $F_n = ?$

Funkcijų dokumentavimas

Bet kuriuo atveju kodą reikia dokumentuoti. Nes esant didesniam projektui, sunku suvaldyti ir prisiminti, kuris kodas už ką atsakingas. **Python**'e tam naudojami komentarai ir dokumentacijos eilutės (*docstrings*).

```
def fn_with_docsatring(x):
    '''
    tarp šių kabučių yra docstring'as. Čia reiktų aprašyti, ką funkcija
↪   turi gauti, ką atlieka, ir ką gražina.
    Pvz:
    x : any type
    return : this very same x
    '''
    return x
```

Failų nuskaitymas ir įrašymas

Failų nuskaitymas

Su Python'u labai lengva nuskaityti paprastus tekstinius failus (.txt, .dat, .csv)

Failų nuskaitymo šablonas

```
failo_vardas = "failo_vardas.csv"
failas = open(failo_vardas, 'r') # r - nuskaitymo režimas
visas_turinys = failas.read()
#po nuskaitymo būtina failą uždaryti:
failas.close()
```

- `'r'` - failas atidarytas tik nuskaitymo režimu, rašyti į failą negalima.
- `'w'` - failas atidarytas rašymo režimu; jei failas egzistuoja - jo turinys bus išvalytas.

Failų nuskaitymas

- `'a'` - failas atidarytas papildymui (*append mode*). Kursorius yra failo pabaigoje. Jei failas neegzistuoja - jis bus sukurtas.

Metodai turinio nuskaitymui

```
.read() # grąžina visą turinį kaip tekstą
.readlines() # grąžina visą turinį kaip eilučių sąrašą.
```

Failų įrašymas

VILNIUS CODING SCHOOL

- Jei failas buvo atidarytas rašymo (`'w'`) ar papildymo (`'a'`) režimu, į failą galima įrašyti naują informaciją.

Failo įrašymo pavyzdys

```
f_name = "testas.txt"
failas = open(f_name, 'w')
tekstas = "Laba diena, čia tekstas, įrašytas su
 ↪  Python'u\n"
failas.write(tekstas)
failas.close()
```

Šiuo atveju failas bus sukurtas darbiniame Python'o aplanke. `'\n '` simbolis nurodo, jog eilutės pabaigoje reikia padėti naujos eilutės simbolį.

Failų papildymas

VILNIUS CODING SCHOOL

- `'a'` režimu atidarius failą, jį galima papildyti nauju tekstu.

Failo paildymas

```
f_name = "testas.txt"
failas = open(f_name, 'a')
tekstas = "Laba diena, čia tekstas, įrašytas su Python'u,
 ↪  papildymo režimu\n"
failas.write(tekstas)
failas.close()
```

`'\n '` simbolis nurodo, jog eilutės pabaigoje reikia padėti naujos eilutės simbolį.

Failų įrašymas

Metodai failų įrašymui

```
.write() # jau rodytas
.writelines([sequence of lines])
```

`.writelines()` metodui reikia nurodyti sąrašą eilučių, pvz.:

`['Pirma eilutė', 'Antra eilutė']`

Eilučių sąrašo įrašymas

```
f_name = "testas.txt"
failas = open(f_name, 'a')
tekstas = ['Pirma eilutė\n', 'Antra eilutė\n']
failas.writelines(tekstas)
failas.close()
```

Failų nuskaitymas ir įrašymas

- Sukurkite tekstinį failą su *Notepad* ar kita programa, failo plėtinys turėtų būti *.txt*
- Iš pirmojo Delfi straipsnio įkopijuokite 2 - 3 pastraipas ir išsaugokite.

Užduotis:

Visi atsakymai rašomi į tą patį *ats.txt* failą.

- Įrašykite tekstą iš sukurtojo failo didžiosiomis raidėmis
- Kas n-tąjį žodį didžiomis raidėmis, n įveda vartotojas.
- Kiekvieną žodį iš didžiosios raidės
- Kiekvienas žodis iš didžiosios raidės, kas antras - parašytas atvirkščiai, kas 5-tas - vien tik didžiosiomis raidėmis, kas 6-tame - simboliai `"a"` turi būti pakeisti `':)'` simboliais.

Failų nuskaitymas

Su Notepad'u susikurkite failą pavadinimu `'t.txt'`, ir jame įrašykite skaičius 1, 2, 3, 4, 5; po vieną skaičių į vieną eilutę.

- Su Python'u nuskaitykite šį failą po vieną eilutę ir išveskite į ekraną.
- Nuskaitydami po eilutę, padauginkite kiekvieną nuskaitytą skaičių iš
  - 2 ir išspausdinkite
  - iš jo paties ir išspausdinkite
- Ankstesniuose punktuose gautuosius rezultatus įrašykite į failą taip: nuskaitytas skaičius, sandauga su 2, sandauga su savimi pačiu. Turite gauti tokį failą:

```
1   2   1
2   4   4
3   6   9
```