Code Academy

1 LYGIS

8 paskaita.
Triukai su sąrašais

2023

Aurimas Aleksandras Nausėdas

Šiandien išmoksite

01 Funkcijas map, filter, reduce, mean, median, sum, max, min

02 Sąrašo apimtis (List Comprehension)

03 Duomenų rūšiavimo būdus

Funkcija map

Sąrašo keitimas (įprastas būdas):

```
sarasas = [4, 3, 2, 1]

sarasas_2 = []
for skaicius in sarasas:
    sarasas_2.append(skaicius ** 2)

print(sarasas_2)

# [16, 9, 4, 1]
```

Būdas su map funkcija:

```
sarasas = [4, 3, 2, 1]
naujas = map(lambda x: x**2, sarasas)
print(naujas)

# <map object at 0x000001AA7B5D8048>

print(list(naujas))

# [16, 9, 4, 1]
```

Datos išskirstymas su map

```
data = "2000-03-25"
y, m, d = map(int, data.split("-"))

print(y)
print(m)
print(d)

# 2000
# 3
# 25
```

```
skaiciai = "4, 3, 2, 1"
p, a, t, k = map(float, skaiciai.split(", "))
print(p, a, t, k)

# 4.0 3.0 2.0 1.0
```

```python
sarasas = [4, 3, 2, 1]

def daugiau_nei_2(sarasas):
    sarasas_2 = []
    for skaicius in sarasas:
        if skaicius > 2:
            sarasas_2.append(skaicius)
    return sarasas_2

print(daugiau_nei_2(sarasas))

# [4, 3]
```

Sąrašo filtravimas (įprastas būdas)

```python
sarasas = [4, 3, 2, 1]
naujas = filter(lambda x: x > 2, sarasas)
print(list(naujas))

# [4, 3]
```

```python
sarasas = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
naujas = filter(lambda x: x % 2 == 0, sarasas)
print(list(naujas))

# [2, 4, 6, 8, 10]
```

```python
import calendar

metai = list(range(1900, 2150))
naujas = list(filter(calendar.isleap, metai))
print(naujas)
```

Funkcija filter

```python
from functools import reduce

sarasas = [4, 3, 2, 1]
naujas = reduce(lambda x, y: x + y, sarasas)
print(naujas)

# 10
```

```python
from functools import reduce

sarasas = [4, 3, 2, 1]
naujas = reduce(lambda x, y: x * y, sarasas)
print(naujas)

# 24
```

Funkcija reduce

```
sarasas = [4, 3, 2, 1, 5, 6, 7, 10, 9, 8]

print(sum(sarasas))

# 55

print(min(sarasas))

# 1

print(max(sarasas))

# 10
```

Funkcijos sum, max, min

```python
from statistics import mean, median

sarasas = [2, 9, 10, 39, 45]

print(mean(sarasas))

# 21

print(median(sarasas))

# 10
```

Vidurkis ir mediana

```python
sarasas = [4, 3, 2, 1]
naujas = [x**2 for x in sarasas]
print(naujas)

# [16, 9, 4, 1]
```

```python
sarasas = [4, 3, 2, 1]
naujas = [x for x in sarasas if x > 2]
print(naujas)

# [4, 3]
```

Sąrašo apimtis (List comprehension)

```python
sarasas = list(range(20))

def lyginiai(sarasas):
    naujas = []
    for skaicius in sarasas:
        if skaicius % 2 == 0:
            naujas.append(skaicius)
    return naujas

print(lyginiai(sarasas))

# [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Lyginių skaičių paieška (įprastas būdas)

```python
sarasas = list(range(20))
lyginiai = [x for x in sarasas if x % 2 == 0]

print(lyginiai)
```

Lyginių skaičių paieška (su list comprehension)

```
sarasas = [4, 3, 2, 1]
naujas = (x**2 for x in sarasas)
print(naujas)

# <generator object <genexpr> at 0x00000261CF927C00>

print(list(naujas))

# [16, 9, 4, 1]
```

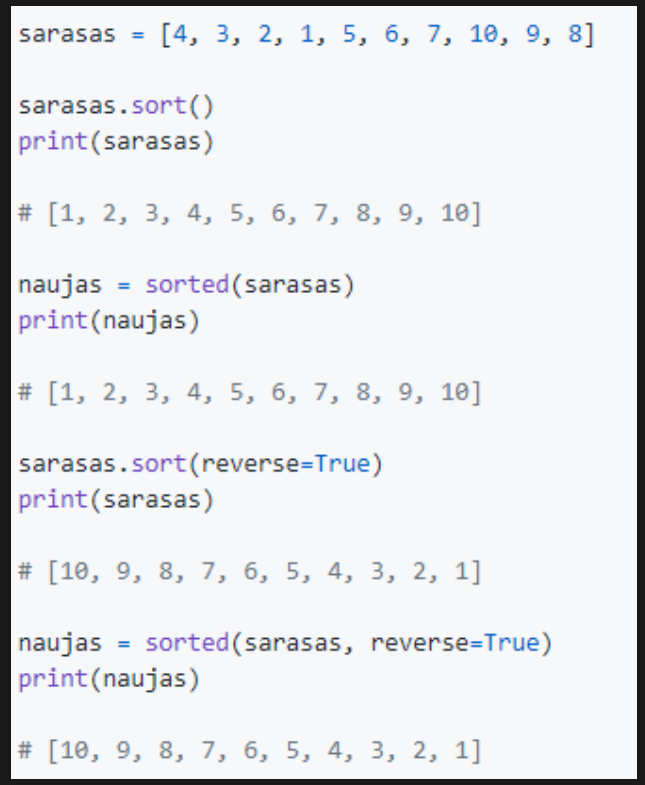
```
sarasas = [4, 3, 2, 1]
naujas = [x**2 for x in sarasas]
print(naujas)

# [16, 9, 4, 1]
```

Atkreipkime dėmesį į rezultato kintamojo tipą!

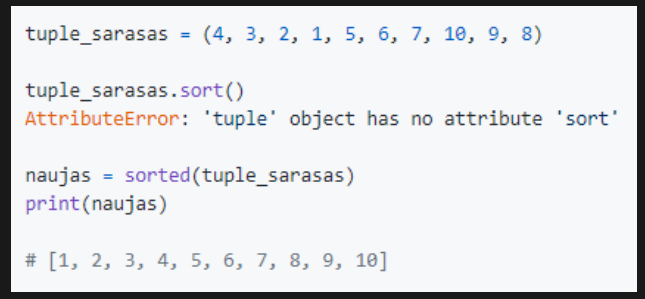
```python
sarasas = [2.5, 2, "Labas", True, 5, 7, 8, 2.8, "Vakaras"]

int_kiekis = sum(type(c) is int for c in sarasas)
print(int_kiekis)

# 4

str_kiekis = sum(type(c) is str for c in sarasas)
print(str_kiekis)

# 2
```

```python
sarasas = [2.5, 2, "Labas", True, 5, 7, 8, 2.8, "Vakaras"]
bool_kiekis = sum(type(c) is bool for c in sarasas)
print(bool_kiekis)

# 1

float_kiekis = sum(type(c) is float for c in sarasas)
print(float_kiekis )

# 2
```

Sąrašo elementų tipų skaičiavimas



```python
sarasas = [4, 3, 2, 1, 5, 6, 7, 10, 9, 8]

sarasas.sort()
print(sarasas)

# [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

naujas = sorted(sarasas)
print(naujas)

# [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

sarasas.sort(reverse=True)
print(sarasas)

# [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

naujas = sorted(sarasas, reverse=True)
print(naujas)

# [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```



```python
tuple_sarasas = (4, 3, 2, 1, 5, 6, 7, 10, 9, 8)

tuple_sarasas.sort()
AttributeError: 'tuple' object has no attribute 'sort'

naujas = sorted(tuple_sarasas)
print(naujas)

# [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Duomenų rūšiavimas

- sort() - nepakeičia esamo sąrašo
- sorted() - gražina pakeistą sąrašą

```python
zodynas = {"Vardas": "Donatas", "Pavardė": "Noreika", "Amžius": None}
surusiuotas = sorted(zodynas)

print(surusiuotas)

# ['Amžius', 'Pavardė', 'Vardas']
```

Žodynų rūšiavimas

```python
sarasas = [-2, 5, -4, 7, -5, 9]
surusiuotas = sorted(sarasas)
print(surusiuotas)

# [-5, -4, -2, 5, 7, 9]

sarasas = [-2, 5, -4, 7, -5, 9]
surusiuotas = sorted(sarasas, key=abs)
print(surusiuotas)

# [-2, -4, 5, -5, 7, 9]
```

Rušiavimas pagal absoliutinę reikšmę

Objektų rūšiavimas sąraše

```python
class Darbuotojas:
    def __init__(self, vardas, pavarde, atlyginimas):
        self.vardas = vardas
        self.pavarde = pavarde
        self.atlyginimas = atlyginimas

    def __repr__(self):
        return (f"({self.vardas}, {self.pavarde}, {self.atlyginimas})")

d1 = Darbuotojas("Tadas", "Rutkauskas", 1500)
d2 = Darbuotojas("Domas", "Radzevičius", 2000)
d3 = Darbuotojas("Rokas", "Ramanauskas", 1000)
sarasas = [d1, d2, d3]

def rusiavimas(darbuotojas):
    return darbuotojas.vardas

surusiuotas = sorted(sarasas, key=rusiavimas)
print(surusiuotas)

#[(Domas, Radzevičius, 2000), (Rokas, Ramanauskas, 1000), (Tadas, Rutkauskas, 1500)]

surusiuotas = sorted(sarasas, key=lambda e: e.atlyginimas)
print(surusiuotas)

# [(Rokas, Ramanauskas, 1000), (Tadas, Rutkauskas, 1500), (Domas, Radzevičius, 2000)]

from operator import attrgetter
surusiuotas = sorted(sarasas, key=attrgetter("atlyginimas"))
print(surusiuotas)
```

Užduotis nr. 1

Sukurti programą, kuri:

- Prie kiekvieno sakinio su jūsų pasirinktu tekstu, pridėtų šauktuką ir atspausdintų naują sakinį.

Patarimai:

- Naudoti map (su lambda) arba comprehension, " ".join()

Užduotis nr. 2

Sukurti programą, kuri:

- Sukurtų sąrašą iš skaičių nuo 0 iki 50
- Padaugintų visus sąrašo skaičius iš 10 ir atspausdintų
- Atrinktų iš sąrašo skaičius, kurie dalinasi iš 7 ir atspausdintų
- Pakeltų visus sąrašo skaičius kvadratu ir atspausdintų
- Su kvadratų sąrašu atliktų šiuos veiksmus: atspausdintų sumą, mažiausią ir didžiausią skaičių, vidurkį, medianą
- Surūšiuotų ir atspausdintų kvadratų sąrašą atbulai

Patarimai:

- Naudoti map, filter arba comprehension, sum, min, max, mean, median, %
- from statistics import mean, median

Užduotis nr. 3

Duotas sąrašas: sarasas = [2.5, 2, "Labas", True, 5, 7, 8, 2.8, "Vakaras"]
Sukurti programą, kuri:

- Paskaičiuotų ir atspausdintų visų sąrašo skaičių sumą
- Sudėtų ir atspausdintų visus sąrašo žodžius
- Suskaičiuotų ir atspausdintų, kiek sąraše yra loginių (boolean) kintamųjų su True reikšme

Patarimai:

- Naudoti filter arba comprehension, sum, " ".join()

Užduotis nr. 4

Sukurti programą, kuri:

- Turėtų klasę Zmogus, su savybėmis vardas ir amzius
- Klasėje būtų **repr** metodas, kuris atvaizduotų vardą ir amžių
- Inicijuoti kelis Zmogus objektus su vardais ir amžiais
- Įdėti sukurtus Zmogus objektus į naują sąrašą
- Surūšiuotų ir atspausdintų sąrašo objektus pagal vardą ir pagal amžių (ir atbulai)

Patarimai:

- Naudoti sorted, attrgetter, reverse, funkciją **repr**
- from operator import attrgetter

Namų darbas

Užbaigti klasėje nepadarytas užduotis

Išspręsti paskaitos uždaviniai (įkelti ketvirtadienį)	https://github.com/aurimas13/Python-Beginner-Course/tree/main/Programs
DB browser for SQLite Duomenų bazės SQLite programa	https://sqlitebrowser.org/

Naudinga informacija