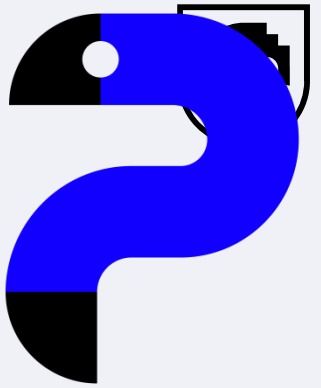
Code Academy



1 LYGIS

5 paskaita. Objektinis programavimas (1 dalis), klasės.

2023

Aurimas Aleksandras Nausėdas

Šiandien išmoksite

01 Kas yra klasės

02 Kaip kurti klasės objektus

03 Atlikti veiksmus naudojant klasės objektus

04 Susipažinsime su objektinio programavimo principais

```
class Kate:
    def __init__(self, spalva, kojos):
        self.spalva = spalva
        self.kojos = kojos
```

```
kate1 = Kate("pilka", 4)
kate2 = Kate("juoda", 3)
```

```
print(kate1.spalva)
print(kate2.kojos)

# juoda
# 3
```

```
kate2.kojos = 4
print(kate1.kojos)
print(kate2.kojos)

# 4
# 4
```

Objektinis programavimas

Objektinis programavimas – programavimo būdas, naudojant objektus ir jų sąveikas

Objektas – į vieną vienetą (klasę) sutalpintos susijusios savybės ir funkcionalumas (kintamieji, funkcijos ir t.t.)

Kaip sukuriama objekto klasė:
Objekto klasė duomenų nesaugo. Ji yra lyg instrukcija, pagal kurią sukuriamas objektas (kuris saugo objekto duomenis).

init metodas (konstruktorius) yra automatiškai įvykdomas kuriant objektą. Jame gali būti inicijuojamos savybės (objekto kintamieji), paleidžiami metodai (funkcijos) ir t. t.

Objekto kintamieji vadinami savybėmis (Property), o funkcijos – metodais (Methods)

```python
class Kate:
    def __init__(self, spalva, kojos):
        self.spalva = spalva
        self.kojos = kojos


    def miaukseti(self):
        print("Miau")
```

```python
kate1 = Kate("pilka", 4)
kate1.miaukseti()

# Miau
```

Kaip sukuriamas metodas (objekto funkcija)

```python
class Kate:
    def __init__(self, spalva, kojos):
        self.spalva = spalva
        self.kojos = kojos

    def _judinti_kojas(self):
        pass

    def _ziureti_kur_begi(self):
        pass

    def begti(self):
        self._judinti_kojas()
        self._ziureti_kur_begi()
        print("Bėgu")
```

```python
muza = Kate("pilka", 4)
muza.begti()

# Bėgu
```

Objektinio programavimo principai.

- Inkapsuliacija (Encapsulation) – vidiniai objekto (klasės) duomenys yra slepiami ir pasiekiami tik metodais (savybėmis, funkcijomis). Tai leidžia neprisirišti prie vidinės objekto struktūros, jį nesunkiai pakeisti kitu arba pakeisti jo struktūrą, nekeičiant pirminio kodo
- Abstrakcija (Abstraction) – galimybė naudotis objektais, nesigilinant į tai, kaip jie veikia. Supaprastina objektų naudojimą, sumažina pakeitimų poveikį likusiams kodui

Kiti du - kitoje paskaitoje

```python
class Kate:
    def __init__(self, spalva = "juoda", kojos = 4):
        self.spalva = spalva
        self.kojos = kojos
```

```python
kate3 = Kate("pilka", 4)
print(kate3.spalva, kate3.kojos)

# pilka 4
```

```python
kate4 = Kate()
print(kate4.spalva, kate4.kojos)

# Juoda 4
```

Kaip sukurti objektą su skirtingu kiekiu savybių

```python
class Kate:
    def __init__(self, spalva, kojos):
        self.spalva = spalva
        self.kojos = kojos
```

```python
kate5 = Kate("Juoda", 4)
print(kate5)

# <__main__.Kate object at 0x000001CADACB6940>
```

```python
class Kate:
    def __init__(self, spalva, kojos):
        self.spalva = spalva
        self.kojos = kojos

    def __str__(self):
        return f"Spalva: {self.spalva}, kojos: {self.kojos}"
```

```python
kate5 = Kate("Juoda", 4)
print(kate5)

# Spalva: Juoda, kojos: 4
```

Kaip pakeisti objekto spausdinimą (str metodas)

```python
pasisveikinimas = "Sveikas, pasauli"

# Objekto tipas, pavadinimas
print(type(pasisveikinimas))

# <class 'str'>


# Vieta darbinėje atmintyje
print(id(pasisveikinimas))

# 2053418975424


# Reikšmė
print(pasisveikinimas)

# Sveikas, pasauli


print(pasisveikinimas.split())

# ['Sveikas,', 'pasauli']


print(pasisveikinimas.upper())

# SVEIKAS, PASAULI
```

Simbolių eilutė (String) kaip objektas

```python
class Kate:
    def __init__(self, spalva, kojos):
        self.spalva = spalva
        self.kojos = kojos


kates = []

kate1 = Kate("Juoda", 4)
kate2 = Kate("Balta", 4)
kate3 = Kate("Pilka", 4)

kates.append(kate1)
kates.append(kate2)
kates.append(kate3)

for kate in kates:
    print(kate.spalva, kate.kojos)

# Juoda 4
# Balta 4
# Pilka 4
```

Kaip objektus sudėti į masyvą ir iš jo išimti

```python
class Kate:
    def __init__(self, vardas, amzius, spalva="juoda"):
        # Savybės:
        self.vardas = vardas
        self.amzius = amzius
        self.spalva = spalva

    # Metodas:
    def miaukseti(self, miauksejimas="Miau", kartai=1):
        print(miauksejimas * kartai)

    def __str__(self):
        return f"Katė vardu {self.vardas}"

    def __repr__(self):
        return f"Katė vardu {self.vardas}"


kates = []

while True:
    pasirinkimas = int(input("Pasirinkite:\n1 - įvesti katę\n2 - peržiūrėti visas kates\n3 - išeiti iš programos\n"))
    if pasirinkimas == 1:
        vardas = input("Įveskite katės vardą")
        amzius = int(input("Įveskite katės amžių"))
        spalva = input("Įveskite katės spalvą")
        kate = Kate(vardas, amzius, spalva)
        kates.append(kate)
    if pasirinkimas == 2:
        for kate in kates:
            print(kate)
    if pasirinkimas == 3:
        print("Viso gero")
        break
```

Kad nereikėtų taip kartotis

Užduotis nr. 1

Parašyti klasę Sakinys, kuri turi savybę tekstas ir metodus, kurie:

- Gražina tekstą atbulai
- Gražina tekstą mažosiomis raidėmis
- Gražina tekstą didžiosiomis raidėmis
- Gražina žodį pagal nurodytą eilės numerį
- Gražina, kiek tekste yra nurodytų simbolių arba žodžių
- Gražina tekstą su pakeistu nurodytu žodžiu arba simboliu
- Atspausdina, kiek sakinyje yra žodžių, skaičių, didžiųjų ir mažųjų raidžių

Susikurti kelis klasės objektus ir išbandyti visus metodus

Užduotis nr. 2

Sukurti klasę Sukaktis, kuri turėtų savybę data (galima atskirai įvesti metus, mėnesius ir kt.) ir metodus, kurie:

- Gražina, kiek nuo įvestos sukakties praėjo metų, savaičių, dienų, valandų, minučių, sekundžių
- Gražina, ar nurodytos sukakties metai buvo keliamieji
- Atima iš nurodytos datos nurodytą kiekį dienų ir gražina naują datą
- Prideda prie nurodytos datos nurodytą kiekį dienų ir gražina naują datą

Užduotis nr. 3

Perdaryti:

- 1 užduotį taip, kad jei kuriant objektą, nepaduodamas joks tekstas, veiksmai turi būti atliekami su „default“ tekstu
- 2 užduotį taip, kad jei kuriant objektą, nepaduodamas jokia data, veiksmai turi būti atliekami su programuotojo gimtadieniu

Užduotis nr. 4

Perdaryti:

- 1 užduotį taip, kad spausdinant sakinio objektą, spausdintų ne objekto adresą, o įvestą tekstą;
- 2 užduotį taip, kad spausdinant datos objektą, spausdintų ne objekto adresą, o įvestą datą

Užduotis nr. 5

Padaryti minibiudžeto programą, kuri:

- Leistų vartotojui įvesti pajamas
- Leistų vartotojui įvesti išlaidas
- Leistų vartotojui parodyti pajamų/išlaidų balansą
- Leistų vartotojui parodyti biudžeto ataskaitą (visus pajamų ir išlaidų įrašus su sumomis)
- Leistų vartotojui išeiti iš programos

Rekomendacija, kaip galima būtų padaryti:

- Programa turi turėti klasę *Irasas*, kuri turėtų argumentus *tipas* (*Pajamos* arba *Išlaidos*) ir *suma*. Galima prirašyti **str** metodą, kuris gražintų, kaip bus atvaizduojamas spausdinamas objektas.
- Programa turi turėti klasę *Biudzetas*, kurioje būtų:
- Metodas **init**, kuriame sukurtas tuščias sąrašas *zurnalas*, į kurį bus dedami sukurti pajamų ir išlaidų objektai
- Metodas prideti_pajamu_irasa(self, suma), kuris priimtų paduotą sumą, sukurtų pajamų objektą ir įdėtų jį į biudžeto žurnalą
- Metodas prideti_islaidu_irasa(self, suma), kuris priimtų paduotą sumą, sukurtų išlaidų objektą ir įdėtų jį į biudžeto žurnalą
- Metodas gauti_balansą(self), kuris gražintų žurnale laikomų pajamų ir išlaidų balansą.
- Metodas parodyti_ataskaita(self), kuris atspausdintų visus pajamų ir išlaidų įrašus (nurodydamas kiekvieno įrašo tipą ir sumą).

Namų darbas

Užbaigti klasėje nepadarytas užduotis

Naudinga informacija

Python String Class and Object

Platesnė informacija apie "str" klasę

https://www.youtube.com/watch?v=YfleC1nWM6A