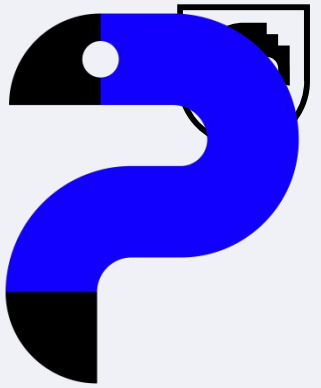
Code Academy



1 LYGIS

6 paskaita. Objektinis programavimas (2 dalis), paveldėjimas.

2021

Python pradedančiųjų kursai

Šiandien išmoksite

01 Kas yra paveldėjimas

02 Kas yra polimorfizmas

03 Atlikti veiksmus naudojant paveldėtus klasės objektus

04 Išmoksime perrašyti paveldėtas funkcijas

```python
class Gyvunas():
    def __init__(self, vardas, spalva):
        self.vardas = vardas
        self.spalva = spalva

    def begti(self):
        print("Bėgu")


class Kate(Gyvunas):
    def miaukseti(self):
        print("Miau")


class Suo(Gyvunas):
    def loti(self):
        print("Au")


vezlys = Gyvunas("Tadas", "Rudas")
vezlys.begti()

# Bėgu

vezlys.miaukseti()

# AttributeError: 'Gyvunas' object has no attribute 'miaukseti'
```

```python
kate1 = Kate("Mūza", "Pilka")
suo1 = Suo("Čakas", "Baltas")

kate1.begti()

# Bėgu

kate1.miaukseti()

# Miau

kate1.loti()

# AttributeError: 'Kate' object has no attribute 'loti'

suo1.loti()

# Au
```

Paveldėjimas (Inheritance)

Galimybė apjungti panašių objektų funkcionalumą, naudojant tėvines klases. Tai leidžia nekartoti panašaus ar to paties kodo. Taip pat nekeičiant paties objekto kodo, papildyti arba keisti jo funkcionalumą.

```python
class Gyvunas():
    def __init__(self, vardas, spalva):
        self.vardas = vardas
        self.spalva = spalva

    def begti(self):
        print("Bėgu")


class Vezlys(Gyvunas):
    def begti(self):
        print("Aš lėtai einu, ne bėgu")

gyvunas = Gyvunas("Jonas", "d")
gyvunas.begti()

vezlys = Vezlys("Tadas", "Rudas")
vezlys.begti()

# Aš lėtai einu, ne bėgu
```

Polimorfizmas (Polymorphism)

Galimybė operacijas (metodus) vykdyti skirtingai, priklausomai nuo konkrečios klasės (ar duomenų tipo) realizacijos, metodo kvietėjui nežinant apie tuos skirtumus. Tai pasiekiama perrašant tam tikrus metodus vaikinėse klasėse.

Metodo (funkcijos) perrašymas (Overriding)

```python
class Gyvunas():
    def __init__(self, vardas, spalva):
        self.vardas = vardas
        self.spalva = spalva

    def begti(self):
        print("Bėgu")


class Vezlys(Gyvunas):
    def begti(self):
        super().begti()
        print("Aš lėtai einu, ne bėgu")


vezlys = Vezlys("Tadas", "Rudas")
vezlys.begti()
```

Kaip pasiekti tėvinės klasės metodą

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


class Vaikas(Tevas):
    def __init__(self, vardas, pavarde, mokymosi_istaiga):
        super().__init__(vardas, pavarde)
        self.mokymosi_istaiga = mokymosi_istaiga


tevas = Tevas("Rokas", "Budreika")
vaikas = Vaikas("Urtė", "Budreikaitė", "Čiurlionio menų gimnazija")

print(tevas.mokymosi_istaiga)
# AttributeError: 'Tevas' object has no attribute 'mokymosi_istaiga'

print(vaikas.mokymosi_istaiga)
# Čiurlionio menų gimnazija
```

Kaip vaikinei klasei pridėti papildomas savybes

```python
class Irasas:
    def __init__(self, suma):
        self.suma = suma

class PajamuIrasas(Irasas):
    pass

class IslaiduIrasas(Irasas):
    pass

biudzetas = []

irasas1 = PajamuIrasas(2000)
irasas2 = IslaiduIrasas(20)
biudzetas.append(irasas1)
biudzetas.append(irasas2)

for x in biudzetas:
    if isinstance(x, PajamuIrasas):
        print(x.suma)
        print("Čia pajamos")
    elif isinstance(x, IslaiduIrasas):
        print(x.suma)
        print("Čia Išlaidos")
```

Kaip patikrinti, kokiai klasei priklauso objektas (biudžeto pavyzdys)

Užduotis nr. 1

Sukurti programą, kuri:

- Turėtų klasę Automobilis
- Automobilis turėtų savybes: metai, modelis, kuro_tipas
- Automobilis turėtų metodus: vaziuoti, stoveti, pildyti_degalu, kurie atitinkamai atspausdintų „Važiuoja“, „Priparkuota“, „Degalai įpilti“
- Sukūrus objektą, automatiškai atspausdintų automobilio metus, modelį ir kuro tipą
- Turėtų klasę Elektromobilis (jo tėvinis objektas – Automobilis)
- Elektromobilis pakeistų Automobilio metodą pildyti_degalu taip, kad jis atspausdintų „Baterija įkrauta“
- Elektromobilis turėtų metodą vaziuoti_autonomiskai, kuris spausdintų „Važiuoja autonomiškai“
- Sukurti norimą Automobilio objektą
- Sukurti norimą Elektromobilio objektą
- Su sukurtu Automobilio objektu paleisti funkcijas vaziuoti, stoveti, pildyti_degalu
- Su sukurtu Elektromobilio objektu paleisti funkcijas vaziuoti, stoveti, pildyti_degalu, vaziuoti_autonomiskai

Užduotis nr. 2

Sukurti programą, kuri:

- Turėtų klasę Darbuotojas
- Darbuotojas turėtų savybes: vardas, valandos_ikainis, dirba_nuo
- Turėtų privatų metodą kuris paskaičiuotų, kiek darbuotojas nudirbo dienų nuo įvestos dienos (dirba_nuo) iki šiandien (turint omeny, kad darbuotojas dirba 7 dienas per savaitę)
- Turėtų metodą paskaiciuoti_atlyginima, kuris panaudodamas aukščiau aprašytu metodu, paskaičiuotų bendrą atlyginimą (turint omeny, kad darbuotojas dirba 8 valandas per dieną)
- Turėtų klasę NormalusDarbuotojas, kuri pakeistų Darbuotojo klasę taip, kad ji skaičiuotų atlyginimą, dirbant darbuotojui 5 dienas per savaitę
- Sukurti norimą Darbuotojo objektą
- Sukurti norimą NormalusDarbuotojas objektą
- Su abiem objektais paleisti funkciją paskaiciuoti_atlyginima

Užduotis nr. 3

Patobulinti 5 pamokos biudžeto programą:

- Sukurti tėvinę klasę *Irasas*, kurioje būtų savybės *suma* , iš kurios klasės *PajamuIrasas* ir *IslaiduIrasas* paveldėtų visas savybes.
- Į klasę *PajamuIrasas* papildomai pridėti savybes *siuntejas* ir *papildoma_informacija*, kurias vartotojas galėtų įrašyti.
- Į klasę *IslaiduIrasas* papildomai pridėti savybes *atsiskaitymo_budas* ir *isigyta_preke_paslauga*, kurias vartotojas galėtų įrašyti.
- Atitinkamai perdaryti klasės *Biudzetas* metodus *gauti_balansa* ir *gauti_ataskaita* kad pasiėmus įrašą iš žurnalo, atpažintų, ar tai yra pajamos ar išlaidos (pvz., panaudojus isinstance() metodą) ir atitinkamai atliktų veiksmus.
- Padaryti, kad vartotojui (per konsolę) būtų leidžiama įrašyti pajamų ir išlaidų įrašus, peržiūrėti balansą ir ataskaitą.

Namų darbas

Užbaigti klasėje nepadarytas užduotis