Web Development with Python Lesson 10

>>>

Ing. Patrik Dendis

STEP IT ACADEMY BRATISLAVA


IN CASE OF FIRE
1. git commit
2. git push
3. git out!

OBSAH PREZENTÁCIE

- Opakovanie
- OOP
- Abstrakcia, Dedenie, Polyformizmus, Enkapsulácia
- Statické metódy
- Magické metódy
- Dekorátory
- Descriptors

OPAKOVANIE

- Ako použijete debugovanie v kóde?
- Čo robí funkcia git commit?
- Čo robí funkcia git push?
- Aká je funkcia na aktualizovanie lokálneho repozitára zo vzdialeného?
- Aký je zmysel git branch?
- Ako zlúčite jednu branch do druhej?
- Ako sa volá základná branch?

ČO JE OOP?

- programovací prístup založený na koncepte objektov, ktoré môžu obsahovať údaje vo forme atribútov alebo vlastností, a kód vo forme procedúr(metód).

- Filozofia a spôsob myslenia, dizajnu a implementácia, kde kladieme dôraz na znovupoužitelnost

- základná jednotka je Objekt, v Pythone je všetko objekt

ČO JE OOP?

- programovací prístup založený na koncepte objektov, ktoré môžu obsahovať údaje vo forme atribútov alebo vlastností, a kód vo forme procedúr(metód).

- Filozofia a spôsob myslenia, dizajnu a implementácia, kde kladieme dôraz na znovupoužitelnost

- základná jednotka je Objekt, v Pythone je všetko objekt

- 4 piliere - **Abstrakcia, Dedenie, Enkapsulácia, Polymorfizmus**

TRIEDA(CLASS) - ABSTRAKCIA

Project

oop ~/PycharmProjects/oop
- venv
- main.py
- External Libraries
- Scratches and Consoles

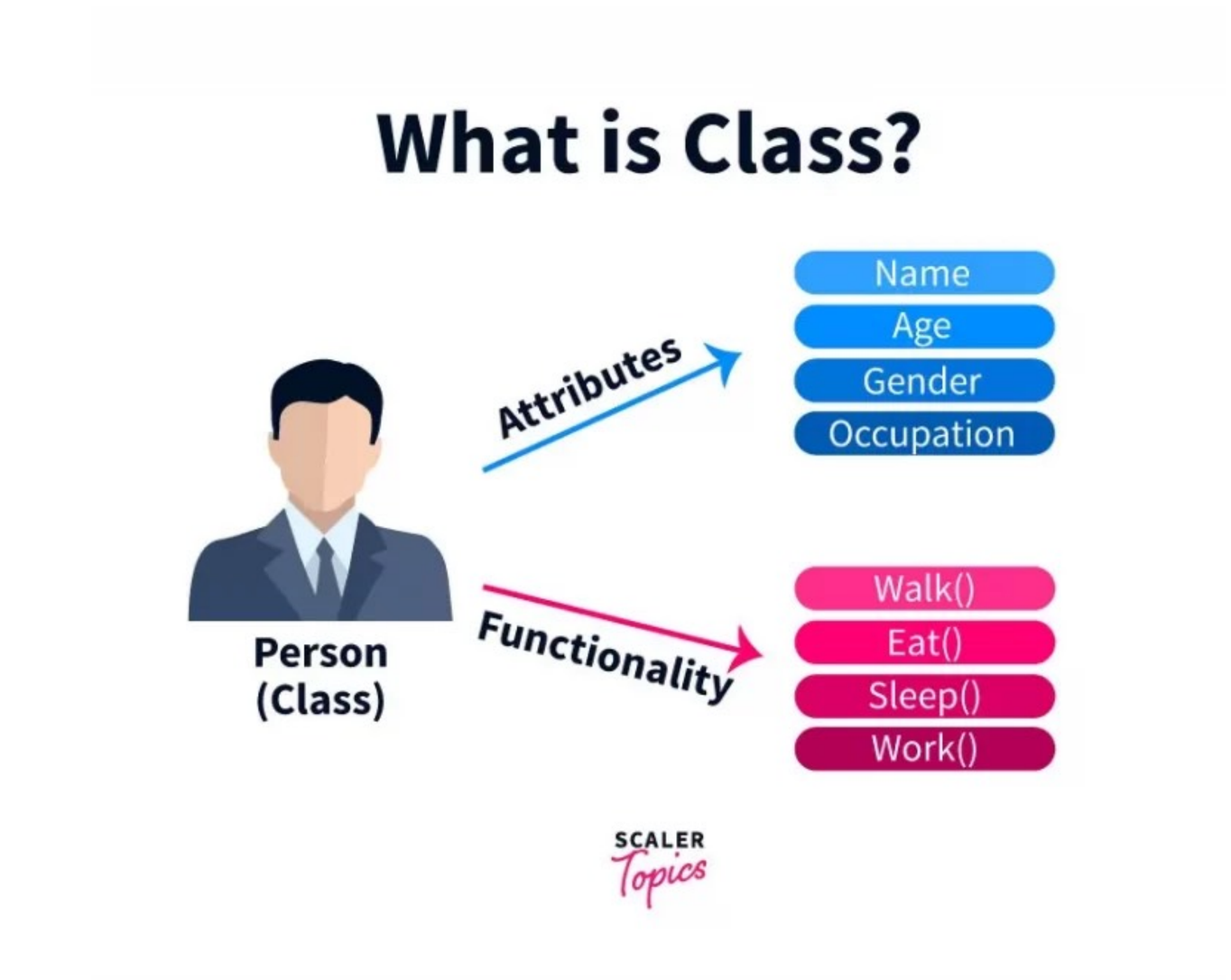
main.py

```python
cislo = 5
retazec = "ahoj"
zoznam = [cislo, retazec]
print(type(cislo))
print(type(retazec))
print(type(zoznam))
```

Run main

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdenc
<class 'int'>
<class 'str'>
<class 'list'>
```

TRIEDA(CLASS) - ABSTRAKCIA



TRIEDA(CLASS) - ABSTRAKCIA

Project

oop ~/PycharmProjects/oop

venv

Folder .py

External Libraries

Scratches and Consoles

main.py

```
1 usage
class Person:
    def __init__(self, name, age, gender, occupation):
        self.name = name
        self.age = age
        self.gender = gender
        self.occupation = occupation

    1 usage
    def pozdrav(self):
        print(f"Ahoj volam sa {self.name} a mam {self.age} rokov")


patrik = Person( name: "Patrik",  age: 30,  gender: "muz",  occupation: "Bratislava")
patrik.pozdrav()
```

Run main

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdendis/PycharmProjects/oop/main.py
Ahoj volam sa Patrik a mam 30 rokov
```

PRÍSTUP – ENKAPSULÁCIA

public, private, protected

Project

- oop ~/PycharmProjects/oop
  - venv
  - main.py
- External Libraries
- Scratches and Consoles

main.py

```python
1 usage
class Person:
    def __init__(self, name, age, gender, occupation):
        self.name = name
        self.age = age
        self._gender = gender # _protected
        self.__occupation = occupation # __private

    1 usage
    def pozdrav(self):
        print(f"Ahoj volam sa {self.name} a mam {self.age} rokov")

    1 usage
    def bydlisko(self):
        print(f"{self.name} byva v {self.__occupation}")


patrik = Person( name: "Patrik",  age: 30,  gender: "muz",  occupation: "Bratislava")
print(patrik.name)
print(patrik.__occupation)
print(patrik._gender)
patrik.pozdrav()
patrik.bydlisko()
```

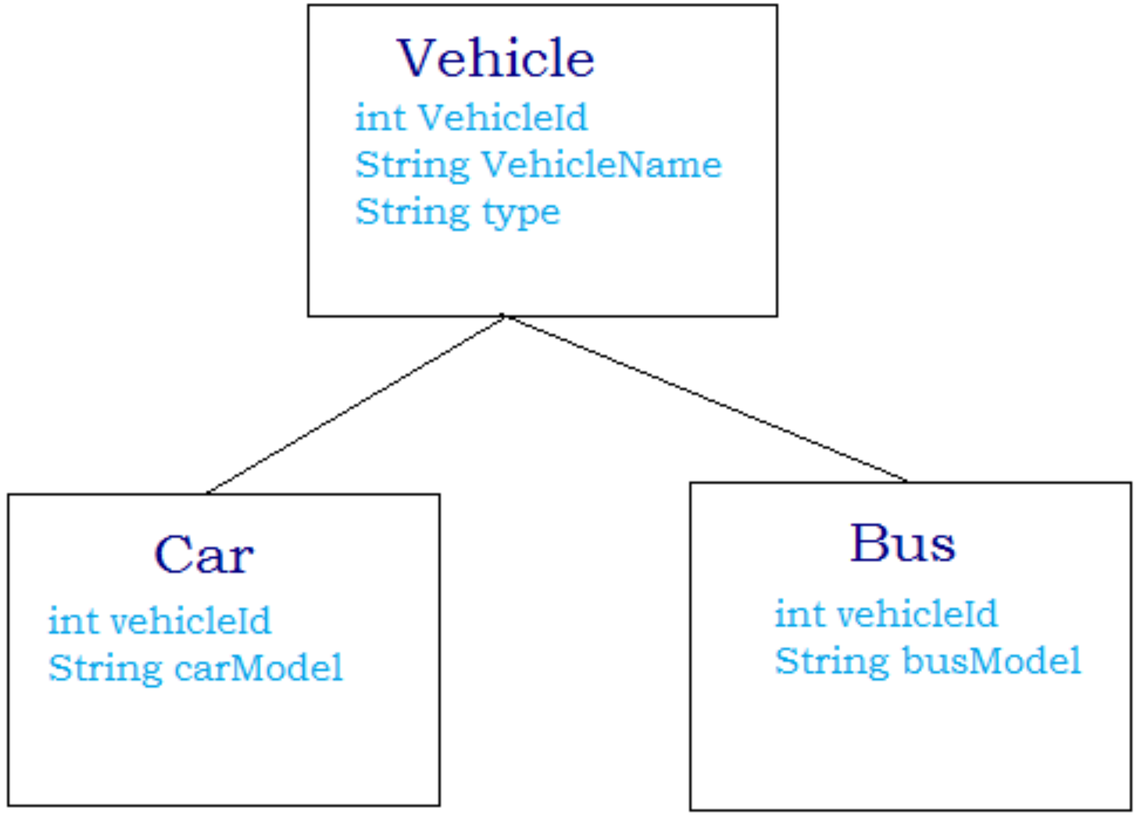
Person › bydlisko()

Run main

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdendis/PycharmProjects/oop/main.py
Patrik
Traceback (most recent call last):
  File "/Users/patrikdendis/PycharmProjects/oop/main.py", line 17, in <module>
    print(patrik.__occupation)
AttributeError: 'Person' object has no attribute '__occupation'
```

DEDENIE

triedy môžu dediť vlastnosti a metódy od iných tried



DEDENIE

```python
class Person:
    def __init__(self, name, age, gender, occupation):
        self.name = name
        self.age = age
        self._gender = gender # _protected
        self.__occupation = occupation # __private

    def pozdrav(self):
        print(f"Ahoj volam sa {self.name} a mam {self.age} rokov")

    def bydlisko(self):
        print(f"{self.name} byva v {self.__occupation}")

class Student(Person):
    def __init__(self, name, age, gender, occupation, score):
        super().__init__(name, age, gender, occupation)
        self.score = score

    def jeGenius(self):
        if self.score > 90:
            print(f"{self.name} je genius a {self._gender}")
        else:
            print(f"{self.name} nie je genius a {self._gender}")


patrik = Student(name: "Patrik", age: 30, gender: "muz", occupation: "Bratislava", score: 95)
milan = Student(name: "Milan", age: 20, gender: "muz", occupation: "Bratislava", score: 40)

print(patrik.name)
patrik.pozdrav()
patrik.bydlisko()
patrik.jeGenius()
milan.jeGenius()
```

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin
Patrik
Ahoj volam sa Patrik a mam 30 rokov
Patrik byva v Bratislava
Patrik je genius a muz
Milan nie je genius a muz

Process finished with exit code 0
```

ZADANIE

Create a class City. Class fields should store the following: city name, region name, country name, number of citizens, zip code, area code. Implement method to get full address.

STATICKÉ METÓDY

metódy v triede, ktoré nepotrebujú inštanciu triedy na to, aby boli volané

main.py

```
2 usages
class MojaKalkulacka:
    1 usage
    @staticmethod
    def sucet(a, b):
        return a + b
    1 usage
    @staticmethod
    def sucin(a, b):
        return a*b

print(MojaKalkulacka.sucet( a: 2,  b: 2))
print(MojaKalkulacka.sucin( a: 2,  b: 3))
```

Run main

```
/Users/patrikdendis/PycharmProjects/oop
4
6

Process finished with exit code 0
```

METÓDY TRIED

metóda ktorá pri volaní dostáva ako prvý parameter odkaz na samotnú triedu, na ktorej je volaná, namiesto inštancie triedy

main.py

```python
class Person:
    def __init__(self, name, age, gender, occupation):
        self.name = name
        self.age = age
        self._gender = gender # _protected
        self.__occupation = occupation # __private

    @staticmethod
    def pozdrav():
        print("Ahoj osoba")

    1 usage
    @classmethod
    def vytvorOsobu(cls, name, year, gender, occupation):
        return cls(name, date.today().year - year, gender, occupation)

    def bydlisko(self):
        print(f"{self.name} byva v {self.__occupation}")


patrik = Person.vytvorOsobu( name: "Patrik",  year: 1993,  gender: "muz",  occupation: "Bratislava")
print(patrik.age)
```

Run main

```
/Users/patrikdendis/PycharmProjects/oo
31

Process finished with exit code 0
```

POLYMORFIZMUS

metódy môžu mať rovnaký názov, ale fungovať odlišne v závislosti od objektu, na ktorom sú volané

main.py

```
3 usages
class Zviera:
    1 usage (1 dynamic)
    def hlas(self):
        raise NotImplementedError("Podtrieda musí implementovať túto metódu")

1 usage
class Pes(Zviera):
    1 usage (1 dynamic)
    def hlas(self):
        return "Haf"

1 usage
class Kohut(Zviera):
    1 usage (1 dynamic)
    def hlas(self):
        return "Kodkodak"

1 usage
class Macka(Zviera):
    1 usage (1 dynamic)
    def hlas(self):
        return "Mňau"

1 usage
def vydaj_zvuk(zviera):
    return zviera.hlas()

pes = Pes()
macka = Macka()
kohut = Kohut()

for zviera in [pes, macka, kohut]:
    print(vydaj_zvuk(zviera))
```

Run main

```
/Users/patrikdendis/
Haf
Mňau
Kodkodak

Process finished wit
```

POLYMORFIZMUS

main.py

```python
class Pes():
    def hlas(self):
        return "Haf"

class Kohut():
    def hlas(self):
        return "Kodkodak"

class Macka():
    def hlas(self):
        return "Mňau"

pes = Pes()
macka = Macka()
kohut = Kohut()

for zviera in [pes, macka, kohut]:
    print(zviera.hlas())
```

Run main

```
/Users/patrikdendis/PycharmProjects/
Haf
Mňau
Kodkodak

Process finished with exit code 0
```

POLYMORFIZMUS

```python
1 usage
class Pes():
    1 usage
    def hlas(self):
        return "Haf"

1 usage
class Kohut():
    1 usage
    def hlas(self):
        return "Kodkodak"

1 usage
class Macka():
    def hlas22(self):
        return "Mňau"

pes = Pes()
macka = Macka()
kohut = Kohut()

for zviera in [pes, macka, kohut]:
    print(zviera.hlas())
```

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdendis/PycharmProjec
Traceback (most recent call last):
  File "/Users/patrikdendis/PycharmProjects/oop/main.py", line 18, in <module>
    print(zviera.hlas())
AttributeError: 'Macka' object has no attribute 'hlas'
Haf

Process finished with exit code 1
```

MAGICKÉ METÓDY

špeciálne metódy, ktoré začínajú a končia dvojitým podčiarkovníkom

- __init__(self, [...]) - konštruktor objektu, volaný pri vytváraní novej inštancie triedy.

- __str__(self) - vráti reprezentáciu objektu vo forme reťazca

- __len__(self) - vráti dĺžku objektu, volané, keď používate funkciu

- __add__(self, other) - umožňuje definovať správanie pre operátor "+"

- __getitem__(self, key) - Umožňuje pristupovať k prvkom objektu pomocou hranatých zátvoriek, napríklad **objekt[key]**

MAGICKÉ METÓDY

2 usages

```python
class Book:
    def __init__(self, title, pages):
        self.title = title
        self.pages = pages


kniha1 = Book( title: "Harry Potter",  pages: 400)
kniha2 = Book( title: "Pan Prstenov",  pages: 1242)
print(str(kniha1))
```

```
/Users/patrikdendis/PycharmProjects/oop/venv/
<__main__.Book object at 0x105b0cbb0>

Process finished with exit code 0
```

MAGICKÉ METÓDY

main.py

2 usages

```python
class Book:
    def __init__(self, title, pages):
        self.title = title
        self.pages = pages

    def __str__(self):
        return f"Kniha sa vola {self.title} a ma {self.pages} stran"



kniha1 = Book( title: "Harry Potter",  pages: 400)
kniha2 = Book( title: "Pan Prstenov",  pages: 1242)
print(str(kniha1))
```

Run main

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/pyt
Kniha sa vola Harry Potter a ma 400 stran

Process finished with exit code 0
```

MAGICKÉ METÓDY

3 usages

```python
class Book:
    def __init__(self, title, pages):
        self.title = title
        self.pages = pages

    def __str__(self):
        return f"Kniha sa vola {self.title} a ma {self.pages} stran"

    def __add__(self, other):
        return Book( title: f"{self.title} {other.title}", self.pages + other.pages)


kniha1 = Book( title: "Harry Potter",  pages: 400)
kniha2 = Book( title: "Pan Prstenov",  pages: 1242)

zlucena_kniha = kniha1 + kniha2
print(str(zlucena_kniha))
```

3 34

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users
Kniha sa vola Harry Potter Pan Prstenov a ma 1642 stran

Process finished with exit code 0
```

MAGICKÉ METÓDY

main.py

2 usages

```python
class Book:
    def __init__(self, title, pages):
        self.title = title
        self.pages = pages

    def __str__(self):
        return f"Kniha sa vola {self.title} a ma {self.pages} stran"



kniha1 = Book( title: "Harry Potter",  pages: 400)
kniha2 = Book( title: "Pan Prstenov",  pages: 1242)

zlucena_kniha = kniha1 + kniha2
print(str(zlucena_kniha))
```

Run main

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdendis/PycharmP
Traceback (most recent call last):
  File "/Users/patrikdendis/PycharmProjects/oop/main.py", line 14, in <module>
    zlucena_kniha = kniha1 + kniha2
TypeError: unsupported operand type(s) for +: 'Book' and 'Book'

Process finished with exit code 1
```

MAGICKÉ METÓDY

main.py

3 usages

```python
class Book:
    def __init__(self, title, pages):
        self.title = title
        self.pages = pages

    def __str__(self):
        return f"Kniha sa vola {self.title} a ma {self.pages} stran"

    def __add__(self, other):
        return Book( title: f"{self.title} {other.title}", self.pages + other.pages)


kniha1 = Book( title: "Harry Potter",  pages: 400)
kniha2 = Book( title: "Harry Potter",  pages: 400)

if kniha1 == kniha2:
    print("je to ta ista kniha")
else:
    print("uplne ina kniha")
```

MAGICKÉ METÓDY

Magic method	Comparison operator, the behavior of which is determined by the method
__eq__	==
__ne__	!=
__lt_	<
__le__	<=
__gt__	>
__ge__	>=

MAGICKÉ METÓDY

oop py

3 usages

```python
class Book:
    def __init__(self, title, pages):
        self.title = title
        self.pages = pages

    def __str__(self):
        return f"Kniha sa vola {self.title} a ma {self.pages} stran"

    def __add__(self, other):
        return Book( title: f"{self.title} {other.title}", self.pages + other.pages)

    def __eq__(self, other):
        return self.title == other.title and self.pages == other.pages


kniha1 = Book( title: "Harry Potter",  pages: 400)
kniha2 = Book( title: "Harry Potter",  pages: 400)

if kniha1 == kniha2:
    print("je to ta ista kniha")
else:
    print("uplne ina kniha")
```

Run main

```
/Users/patrikdendis/PycharmProjects/oo
je to ta ista kniha

Process finished with exit code 0
```

DEKORÁTORY

```
2 usages
def info_decorator(method_to_decorate):
    def wrapper(self):
        print("Informacie:")
        return method_to_decorate(self)
    return wrapper

1 usage
class Book:
    def __init__(self, title, pages):
        self.title = title
        self.pages = pages

    1 usage
    @info_decorator
    def show_info(self):
        return f"Kniha sa vola {self.title} a ma {self.pages} stran"


1 usage
class Movie:
    def __init__(self, title, minutes):
        self.title = title
        self.minutes = minutes

    1 usage
    @info_decorator
    def show_info(self):
        return f"Film sa vola {self.title} a ma {self.minutes} minut"


kniha = Book( title: "Harry Potter",  pages: 400)
film = Movie( title: "Avatar",  minutes: 230)
print(kniha.show_info())
print(film.show_info())
```

```
/Users/patrikdendis/PycharmProjects/oop/venv/
Informacie:
Kniha sa vola Harry Potter a ma 400 stran
Informacie:
Film sa vola Avatar a ma 230 minut

Process finished with exit code 0
```

METÓDY

main.py

1 usage

```python
class Book:
    def __init__(self, title, pages, price):
        self.title = title
        self.pages = pages
        self.price = price

    1 usage
    def price_to_usd(self, rate):
        return self.price * rate


kniha = Book( title: "Harry Potter",  pages: 400,  price: 10)
print(kniha.price_to_usd(1.09))
```

Run main

```
/Users/patrikdendis/PycharmProjects/oop/ve
10.9

Process finished with exit code 0
```

DESCRIPTORS

```python
class Book:
    def __init__(self, title, pages, price):
        self.title = title
        self.pages = pages
        self.__price = price

    3 usages
    @property
    def price(self):
        return self.__price

    2 usages
    @price.setter
    def price(self, value):
        if value >= 0:
            self.__price = value
        else:
            raise ValueError("Price is negative")


kniha = Book( title: "Harry Potter",  pages: 400,  price: 10)
print(kniha.price)
kniha.price = 20
print(kniha.price)
kniha.price = -10
```

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdendis/Pyc
Traceback (most recent call last):
  File "/Users/patrikdendis/PycharmProjects/oop/main.py", line 23, in <module>
    kniha.price = -10
  File "/Users/patrikdendis/PycharmProjects/oop/main.py", line 16, in price
    raise ValueError("Price is negative")
ValueError: Price is negative
10
20

Process finished with exit code 1
```

DESCRIPTORS

```python
class Book:
    def __init__(self, title, pages, price):
        self.title = title
        self.pages = pages
        self.__price = price

    3 usages
    @property
    def price(self):
        return self.__price

    2 usages
    @price.setter
    def price(self, value):
        if value >= 0:
            self.__price = value
        else:
            raise ValueError("Price is negative")


kniha = Book( title: "Harry Potter",  pages: 400,  price: 10)
print(kniha.price)
kniha.price = 20
print(kniha.price)
kniha.price = -10
```

```
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdendis/Pyc
Traceback (most recent call last):
  File "/Users/patrikdendis/PycharmProjects/oop/main.py", line 23, in <module>
    kniha.price = -10
  File "/Users/patrikdendis/PycharmProjects/oop/main.py", line 16, in price
    raise ValueError("Price is negative")
ValueError: Price is negative
10
20

Process finished with exit code 1
```

ZADANIE

Create a class to convert from metric to imperial, and vice versa. Implement this functionality as static methods. Be sure to implement a converter for units of length.

ZADANIE

Create a base class Shape with a method for calculating the area. Create derived classes: a rectangle, circle, right triangle with their own methods for calculating the area. Practicing polymorphism

ĎAKUJEM ZA POZORNOSŤ