



>>>

Web Development with Python Lesson 10



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?

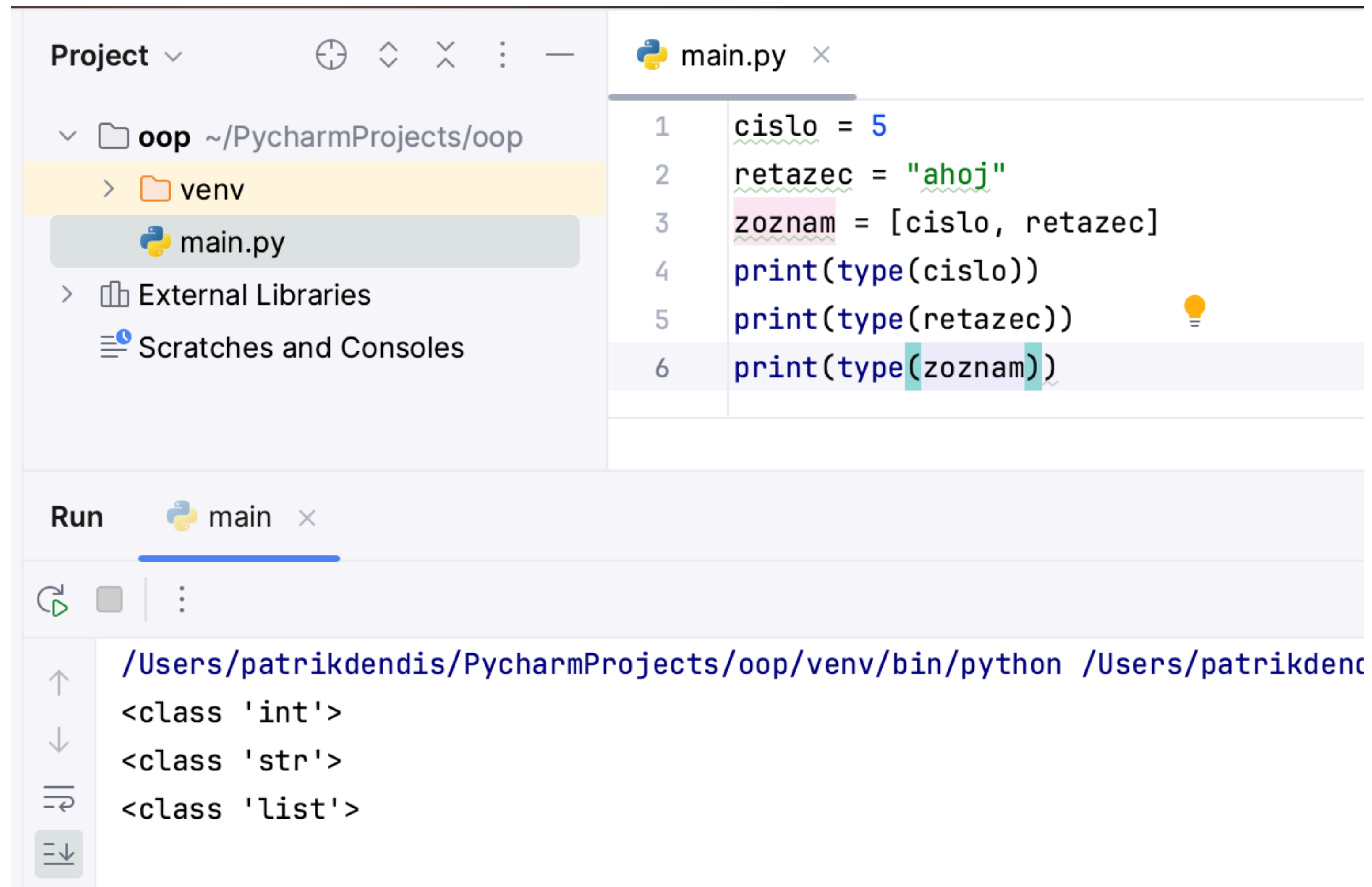
- programovací přístup založený na koncepte objektov, ktoré môžu obsahovať údaje vo forme atribútov alebo vlastností, a kód vo forme procedúr(metód).
- Filozofia a spôsob myslenia, dizajnu a implementácia, kde kladieme dôraz na znovupoužitelnosť
- základná jednotka je Objekt, v Pythone je všetko objekt

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 znovupoužitelnosť
- základná jednotka je Objekt, v Pythone je všetko objekt
- 4 piliere – **Abstrakcia, Dedenie, Enkapsulácia, Polymorfizmus**

TRIEDA(CLASS) - ABSTRAKCIA

7



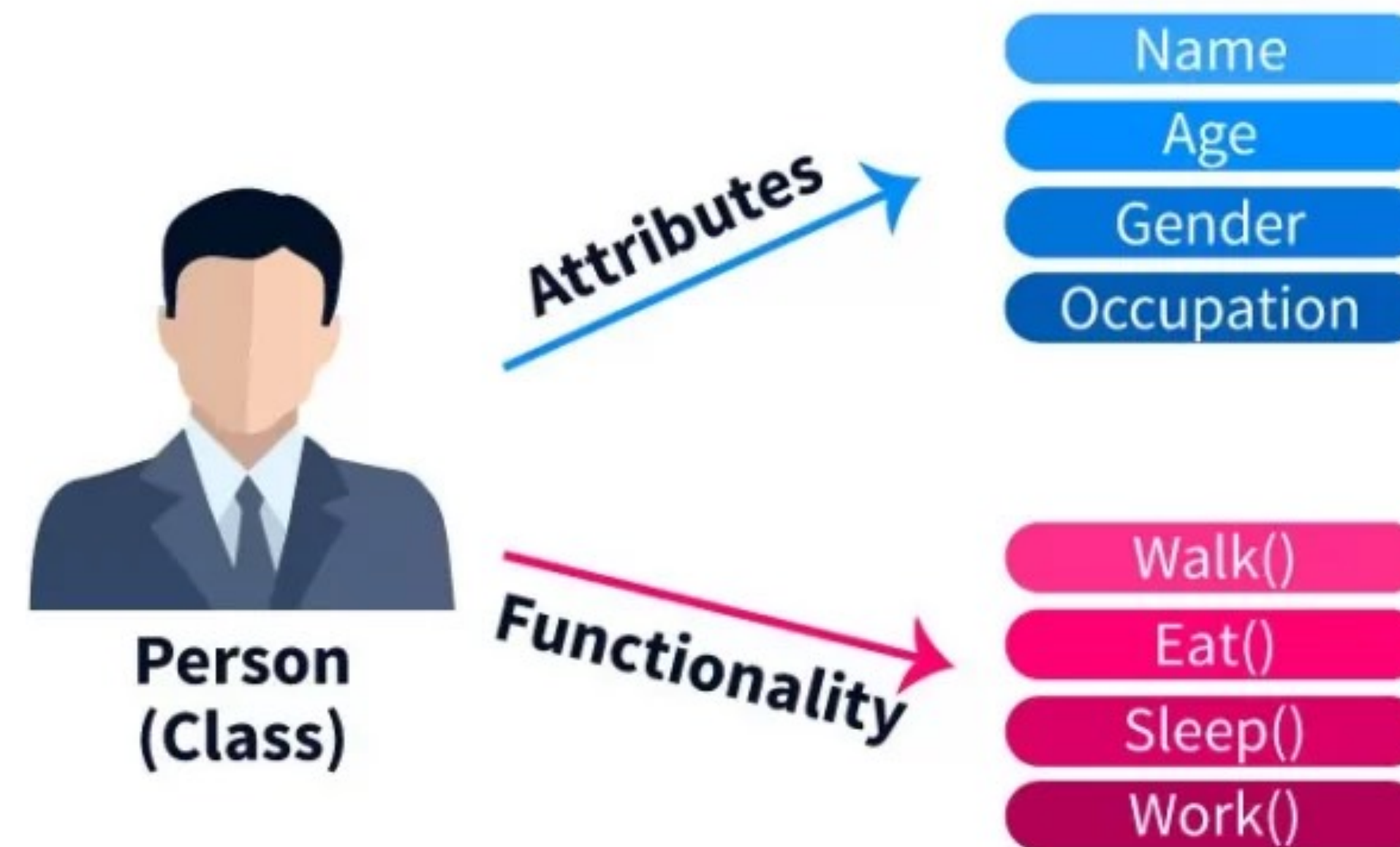
The screenshot shows the PyCharm IDE interface. The top-left pane displays the project structure with a folder named 'oop' containing a 'venv' folder and a 'main.py' file. The top-right pane shows the code in 'main.py':

```
1 cislo = 5
2 retazec = "ahoj"
3 zoznam = [cislo, retazec]
4 print(type(cislo))
5 print(type(retazec))
6 print(type(zoznam))
```

The bottom pane shows the output of the program, which is the type of each variable: `<class 'int'>`, `<class 'str'>`, and `<class 'list'>`.

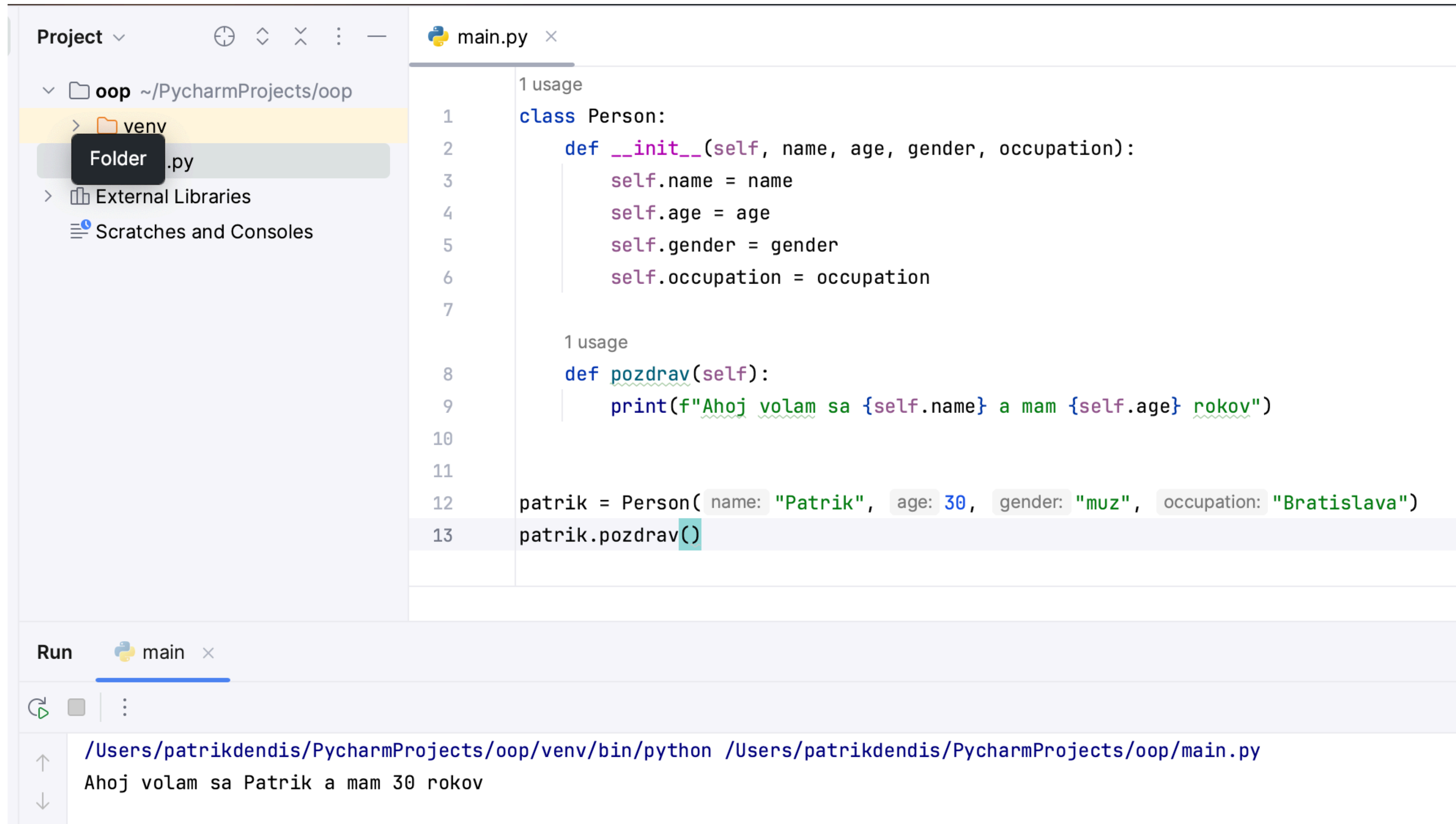
TRIEDA(CLASS) - ABSTRAKCIA

What is Class?



TRIEDA(CLASS) - ABSTRAKCIA

9



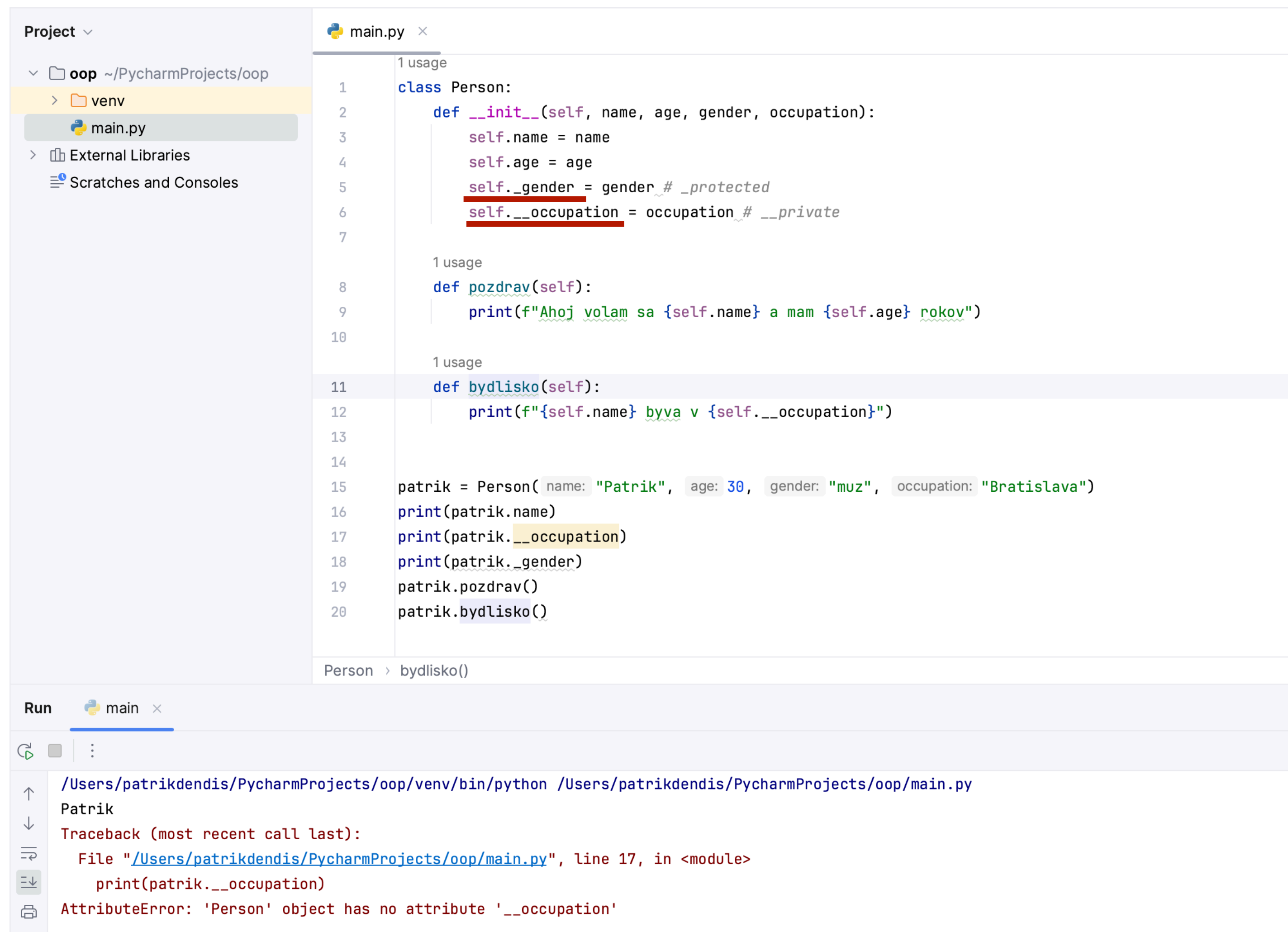
The screenshot shows the PyCharm IDE interface. On the left, the 'Project' view displays a directory structure with 'oop' as the root, containing a 'venv' folder and 'External Libraries'. A tooltip 'Folder .py' is visible over the 'venv' folder. The main editor window shows 'main.py' with the following code:

```
1 usage
2 class Person:
3     def __init__(self, name, age, gender, occupation):
4         self.name = name
5         self.age = age
6         self.gender = gender
7         self.occupation = occupation
8
9     def pozdrav(self):
10         print(f"Ahoj volam sa {self.name} a mam {self.age} rokov")
11
12 patrik = Person(name: "Patrik", age: 30, gender: "muz", occupation: "Bratislava")
13 patrik.pozdrav()
```

Below the editor, the 'Run' tab is active, showing the command executed: `/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdendis/PycharmProjects/oop/main.py`. The output of the program is displayed as: `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

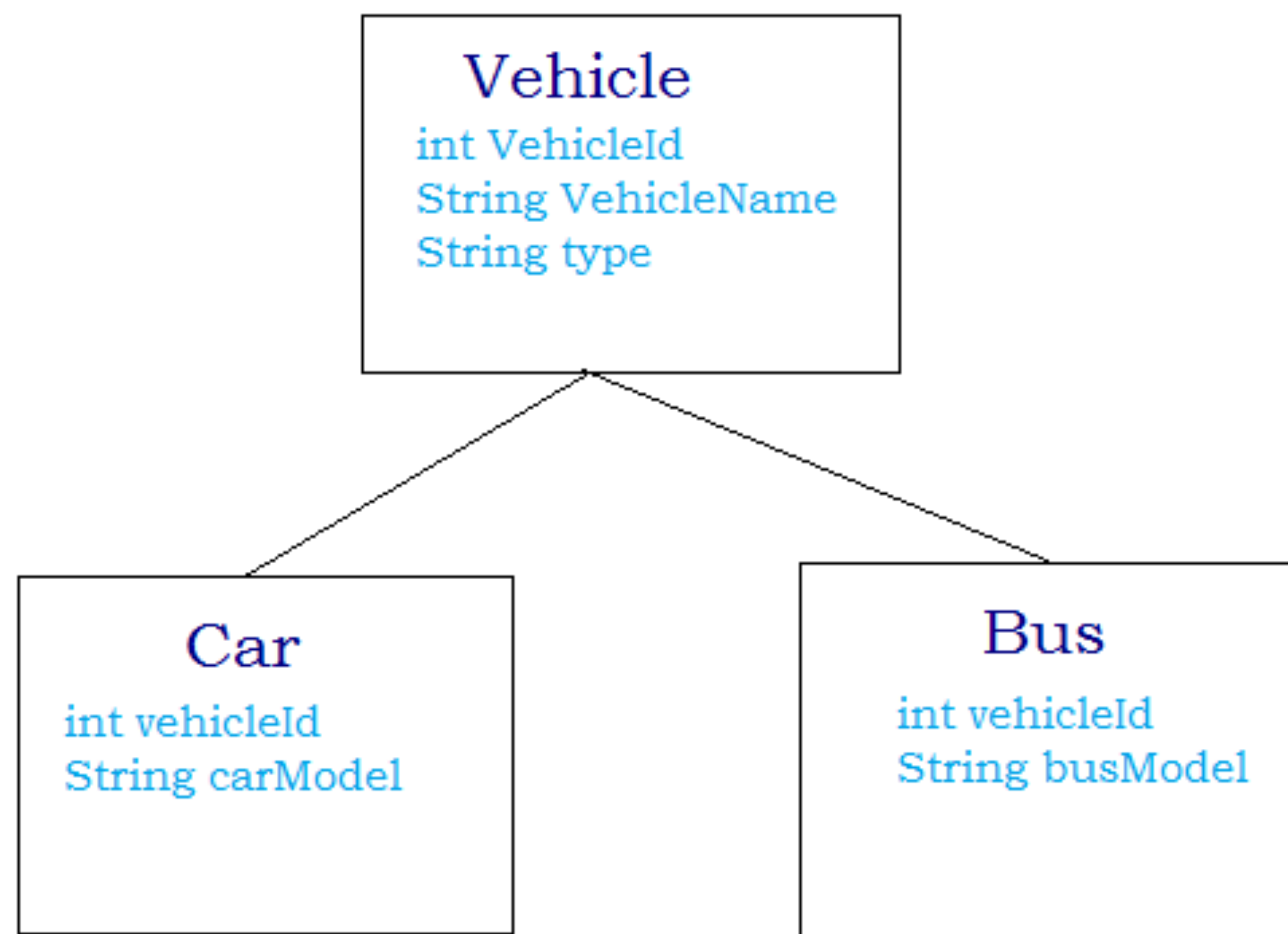
main.py x
1 usage
1 class Person:
2     def __init__(self, name, age, gender, occupation):
3         self.name = name
4         self.age = age
5         self._gender = gender # _protected
6         self.__occupation = occupation # __private
7
8     1 usage
9     def pozdrav(self):
10         print(f"Ahoj volam sa {self.name} a mam {self.age} rokov")
11
12     1 usage
13     def bydlisko(self):
14         print(f"{self.name} byva v {self.__occupation}")
15
16 patrik = Person( name: "Patrik", age: 30, gender: "muz", occupation: "Bratislava")
17 print(patrik.name)
18 print(patrik.__occupation)
19 print(patrik._gender)
20 patrik.pozdrav()
21 patrik.bydlisko()

Person > bydlisko()

Run main x
/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

1 usage

1 class Person:

2 def __init__(self, name, age, gender, occupation):

3 self.name = name

4 self.age = age

5 self._gender = gender # _protected

6 self.__occupation = occupation # __private

7

1 usage

8 def pozdrav(self):

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

10

1 usage

11 def bydlisko(self):

12 print(f"{self.name} byva v {self.__occupation}")

13

2 usages

14 class Student(Person):

15 def __init__(self, name, age, gender, occupation, score):

16 super().__init__(name, age, gender, occupation)

17 self.score = score

18

2 usages

19 def jeGenius(self):

20 if self.score > 90:

21 print(f"{self.name} je genius a {self._gender}")

22 else:

23 print(f"{self.name} nie je genius a {self._gender}")

24

25

26 patrik = Student(name: "Patrik", age: 30, gender: "muz", occupation: "Bratislava", score: 95)

27 milan = Student(name: "Milan", age: 20, gender: "muz", occupation: "Bratislava", score: 40)

28

29 print(patrik.name)

30 patrik.pozdrav()

31 patrik.bydlisko()

32 patrik.jeGenius()

33 milan.jeGenius()

5 6 ^ v

↻

⌵

:

/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

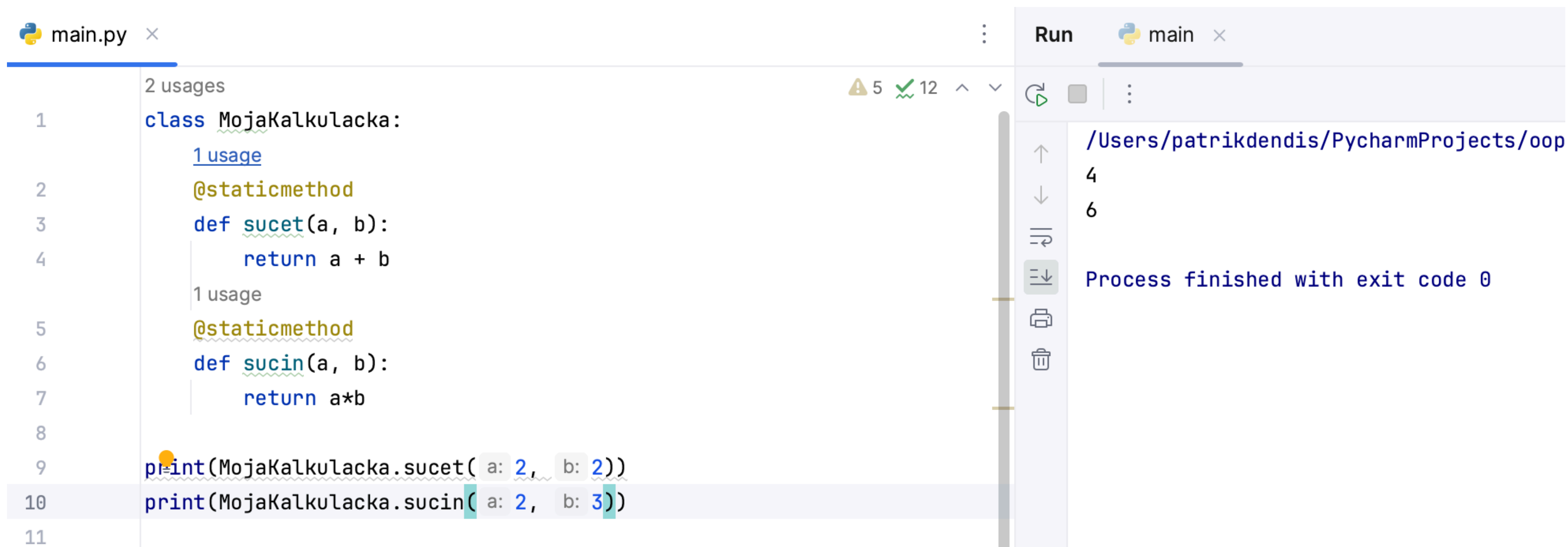
|

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 x
2 usages
1 class MojaKalkulacka:
    1 usage
2     @staticmethod
3     def sucet(a, b):
4         return a + b
5
6     @staticmethod
7     def sucin(a, b):
8         return a*b
9
10 print(MojaKalkulacka.sucet(a: 2, b: 2))
11 print(MojaKalkulacka.sucin(a: 2, b: 3))

Run main x
/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 x
3 class Person:
4     def __init__(self, name, age, gender, occupation):
5         self.name = name
6         self.age = age
7         self._gender = gender # _protected
8         self.__occupation = occupation # __private
9
10    @staticmethod
11    def pozdrav():
12        print("Ahoj osoba")
13
14    1 usage
15    @classmethod
16    def vytvorOsobu(cls, name, year, gender, occupation):
17        return cls(name, date.today().year - year, gender, occupation)
18
19    def bydlisko(self):
20        print(f"{self.name} byva v {self.__occupation}")
21
22    patrik = Person.vytvorOsobu(name: "Patrik", year: 1993, gender: "muz", occupation: "Bratislava")
23    print(patrik.age)
24
```

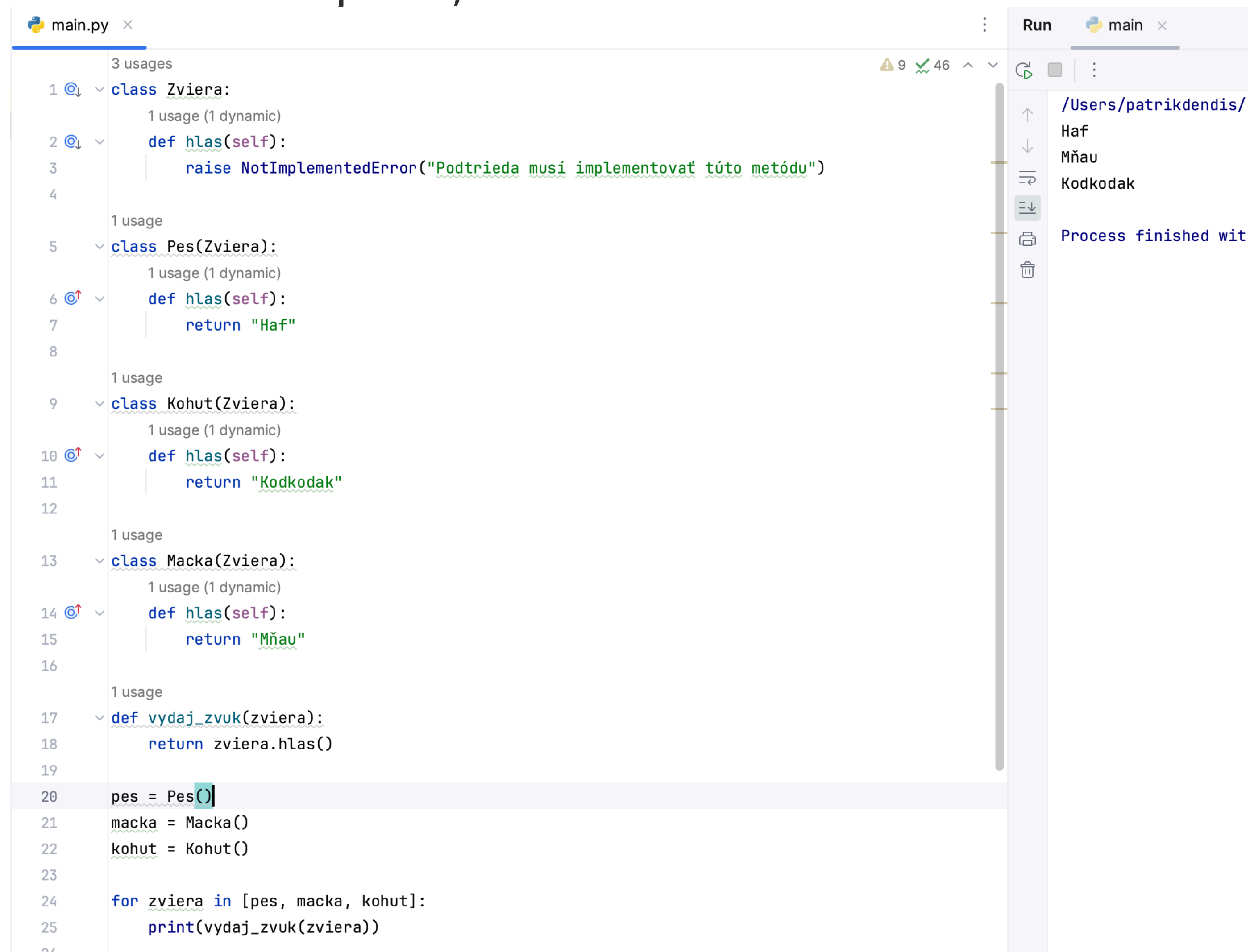
Run main x

/Users/patrikdendis/PycharmProjects/oo
31

Process finished with exit code 0

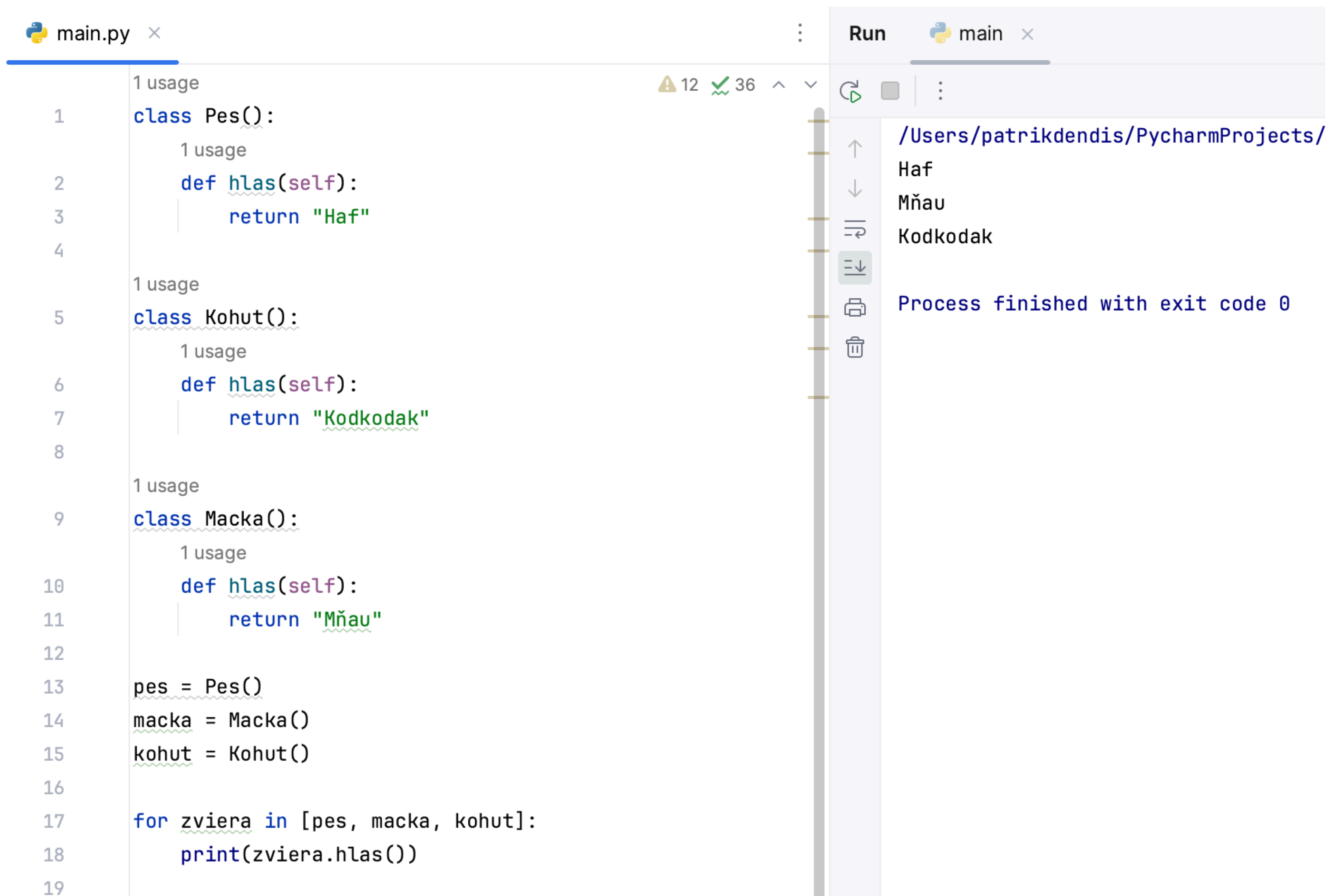
POLYMORFIZMUS

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



```
main.py x
3 usages
1 class Zviera:
    1 usage (1 dynamic)
2     def hlas(self):
3         raise NotImplementedError("Podtrieda musí implementovať túto metódu")
4
5     1 usage
6     class Pes(Zviera):
7         1 usage (1 dynamic)
8         def hlas(self):
9             return "Haf"
10
11     1 usage
12     class Kohut(Zviera):
13         1 usage (1 dynamic)
14         def hlas(self):
15             return "Kodkodak"
16
17     1 usage
18     class Macka(Zviera):
19         1 usage (1 dynamic)
20         def hlas(self):
21             return "Mňau"
22
23     1 usage
24     def vydaj_zvuk(zviera):
25         return zviera.hlas()
26
27 pes = Pes()
28 macka = Macka()
29 kohut = Kohut()
30
31 for zviera in [pes, macka, kohut]:
32     print(vydaj_zvuk(zviera))
33
Run main x
/Users/patrikdendis/
Haf
Mňau
Kodkodak
Process finished with...
```

POLYMORFIZMUS



```
main.py x
1 usage
1 class Pes():
1 usage
2     def hlas(self):
3         return "Haf"
4
1 usage
5 class Kohut():
1 usage
6     def hlas(self):
7         return "Kodkodak"
8
1 usage
9 class Macka():
1 usage
10     def hlas(self):
11         return "Mňau"
12
13 pes = Pes()
14 macka = Macka()
15 kohut = Kohut()
16
17 for zviera in [pes, macka, kohut]:
18     print(zviera.hlas())
19
```

Run main x

/Users/patrikdendis/PycharmProjects/
Haf
Mňau
Kodkodak

Process finished with exit code 0

POLYMORFIZMUS

```
1 usage
1  class Pes():
2      1 usage
3      def hlas(self):
4          return "Haf"
5
6      1 usage
7      class Kohut():
8          1 usage
9          def hlas(self):
10             return "Kodkodak"
11
12      1 usage
13      class Macka():
14          def hlas22(self):
15             return "Mňau"
16
17      pes = Pes()
18      macka = Macka()
19      kohut = Kohut()
20
21      for zviera in [pes, macka, kohut]:
22          print(zviera.hlas())
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
12 36 ^ v
/Users/patrikdendis/PycharmProjects/oop/venv/bin/python /Users/patrikdendis/PycharmProjects/oop/main.py
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 dvojitém podčiarkovníkom

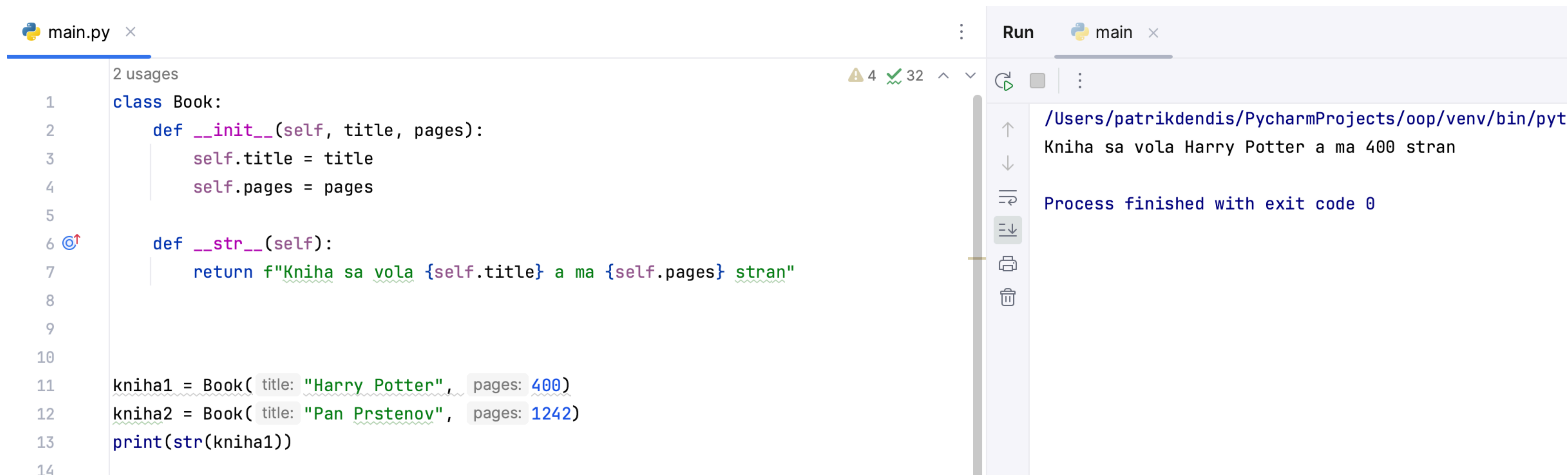
- `__init__(self, [...])` – konštruktor objektu, volaný pri vytváraní novej inštancie triedy.
- `__str__(self)` – vráti reprezentáciu objektu vo forme reťazca
- `__len__(self)` – vráti dĺžku objektu, volané, keď používate funkciu
- `__add__(self, other)` – umožňuje definovať správanie pre operátor `"+"`
- `__getitem__(self, key)` – Umožňuje pristupovať k prvkom objektu pomocou hranatých zátvoriek, napríklad **objekt[key]**

MAGICKÉ METÓDY

```
2 usages
1 class Book:
2     def __init__(self, title, pages):
3         self.title = title
4         self.pages = pages
5
6
7 kniha1 = Book(title: "Harry Potter", pages: 400)
8 kniha2 = Book(title: "Pan Prstenov", pages: 1242)
9 print(str(kniha1))
10
```

Process finished with exit code 0

MAGICKE METODY



```
main.py x
2 usages
1 class Book:
2     def __init__(self, title, pages):
3         self.title = title
4         self.pages = pages
5
6     def __str__(self):
7         return f"Kniha sa voľa {self.title} a ma {self.pages} stran"
8
9
10
11 kniha1 = Book( title: "Harry Potter", pages: 400)
12 kniha2 = Book( title: "Pan Prstenov", pages: 1242)
13 print(str(kniha1))
14
```

Run main x

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

Process finished with exit code 0

MAGICKÉ METÓDY

[3 usages](#)

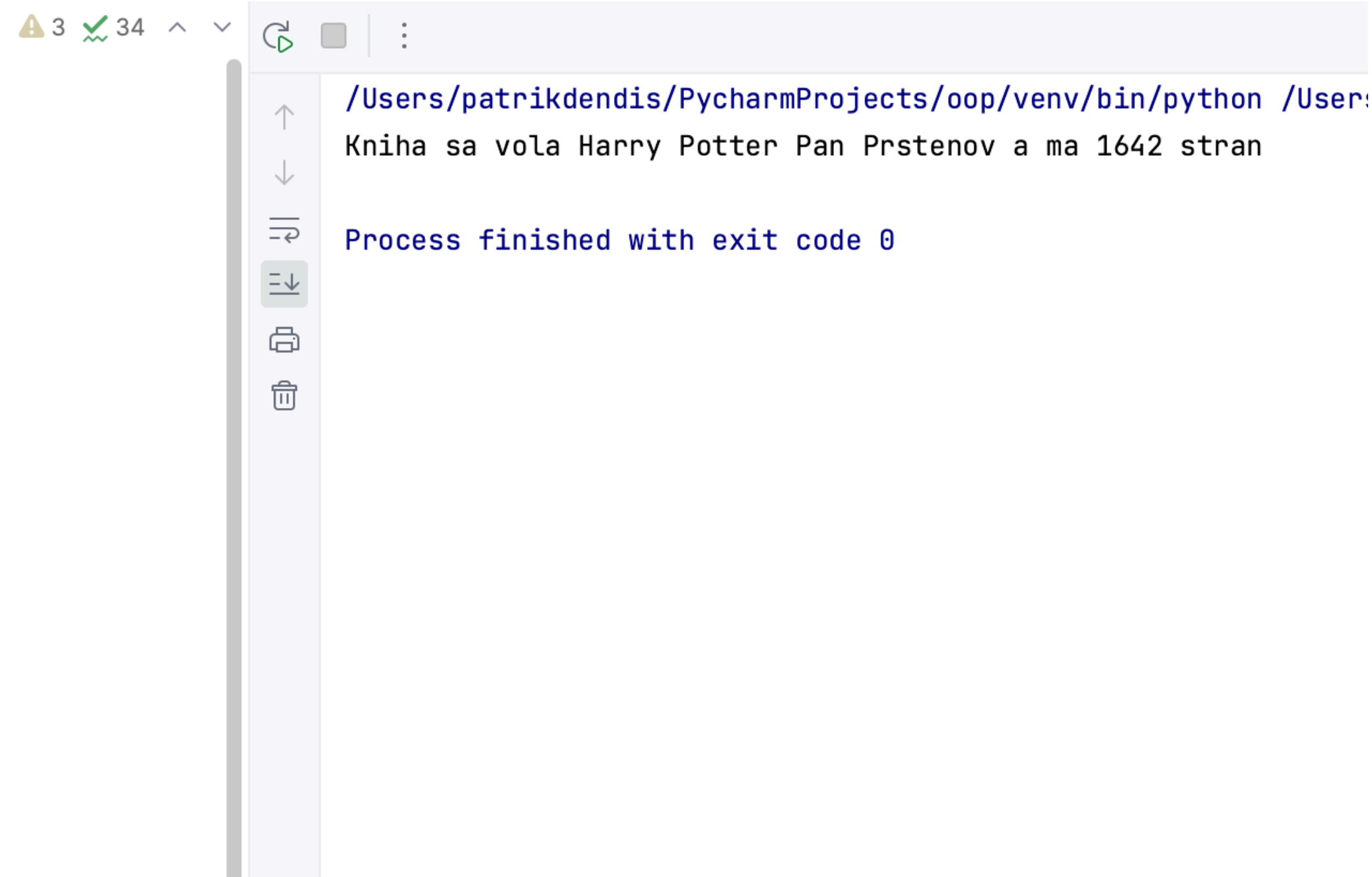
```
class Book:
    def __init__(self, title, pages):
        self.title = title
        self.pages = pages

    def __str__(self):
        return f"Kniha sa voľa {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))
```



MAGICKÉ METÓDY

The screenshot shows a PyCharm IDE with a file named `main.py`. The code defines a `Book` class with `__init__` and `__str__` methods. It then creates two `Book` objects, `kniha1` and `kniha2`, and attempts to add them together (`zrucena_kniha = kniha1 + kniha2`), which results in a `TypeError`.

```
1 class Book:
2     def __init__(self, title, pages):
3         self.title = title
4         self.pages = pages
5
6     def __str__(self):
7         return f"Kniha sa vola {self.title} a ma {self.pages} stran"
8
9
10
11 kniha1 = Book(title: "Harry Potter", pages: 400)
12 kniha2 = Book(title: "Pan Prstenov", pages: 1242)
13
14 zrucena_kniha = kniha1 + kniha2
15 print(str(zrucena_kniha))
```

The Run window shows the following traceback:

```
/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>
    zrucena_kniha = kniha1 + kniha2
TypeError: unsupported operand type(s) for +: 'Book' and 'Book'

Process finished with exit code 1
```

MAGICKE METODY

main.py x

```
3 usages
1 class Book:
2     def __init__(self, title, pages):
3         self.title = title
4         self.pages = pages
5
6     def __str__(self):
7         return f"Kniha sa vola {self.title} a ma {self.pages} stran"
8
9     def __add__(self, other):
10        return Book(title: f"{self.title} {other.title}", self.pages + other.pages)
11
12
13 kniha1 = Book(title: "Harry Potter", pages: 400)
14 kniha2 = Book(title: "Harry Potter", pages: 400)
15
16 if kniha1 == kniha2:
17     print("je to ta ista kniha")
18 else:
19     print("uplne ina kniha")
20
```

MAGICKÉ METÓDY

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

MAGICKE METODY



The screenshot displays a PyCharm IDE with a Python file named 'oop.py'. The code defines a 'Book' class with several magic methods: `__init__`, `__str__`, `__add__`, and `__eq__`. It also includes a test script that creates two 'Book' objects and checks if they are equal.

```
1 class Book:
2     def __init__(self, title, pages):
3         self.title = title
4         self.pages = pages
5
6     def __str__(self):
7         return f"Kniha sa vola {self.title} a ma {self.pages} stran"
8
9     def __add__(self, other):
10        return Book( title: f"{self.title} {other.title}", self.pages + other.pages)
11
12    def __eq__(self, other):
13        return self.title == other.title and self.pages == other.pages
14
15
16 kniha1 = Book( title: "Harry Potter", pages: 400)
17 kniha2 = Book( title: "Harry Potter", pages: 400)
18
19 if kniha1 == kniha2:
20     print("je to ta ista kniha")
21 else:
22     print("uplne ina kniha")
23
```

The Run console on the right shows the output of the script:

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

Process finished with exit code 0
```


DEKORÁTOR

```
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"

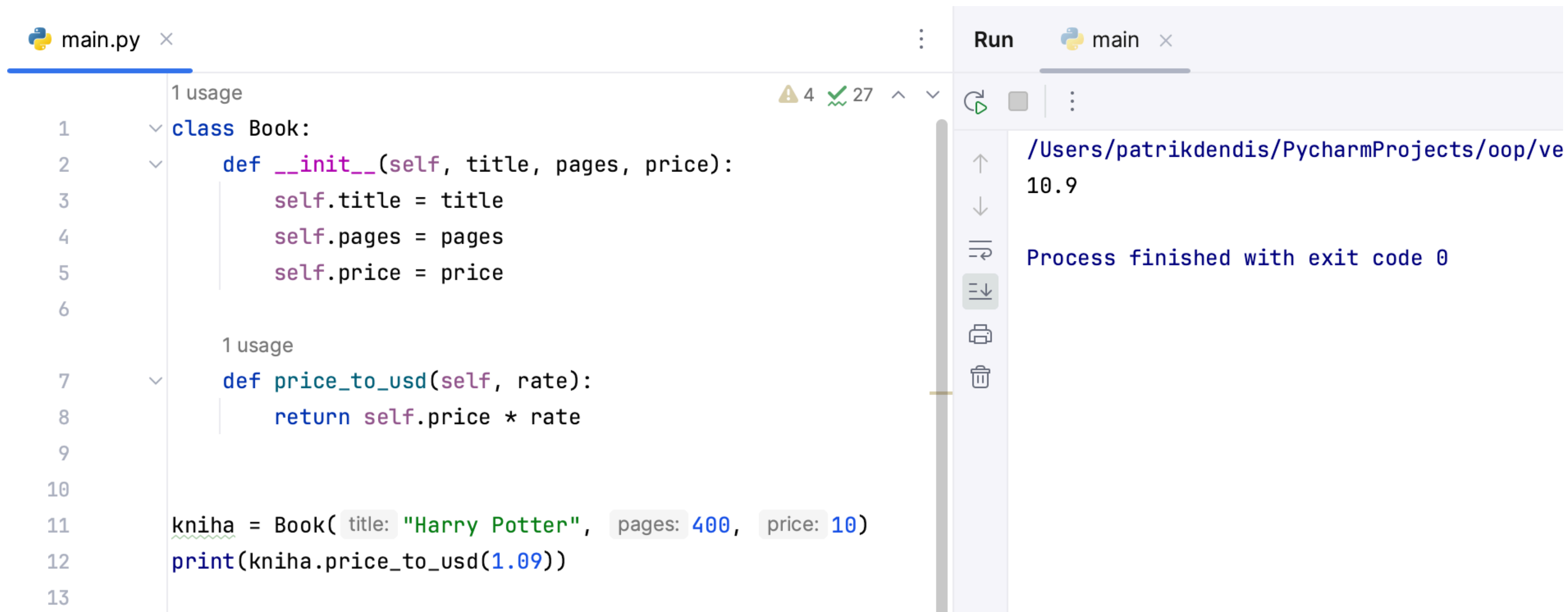
kniha = Book(title="Harry Potter", pages=400)
film = Movie(title="Avatar", minutes=230)
print(kniha.show_info())
print(film.show_info())
```

2 4 33 ^ v

```
/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



The screenshot shows a PyCharm IDE with a Python file named `main.py`. The code defines a `Book` class with an `__init__` method and a `price_to_usd` method. An instance of the class is created and the `price_to_usd` method is called with a rate of 1.09. The Run window on the right shows the output of the program.

```
1 usage
2 class Book:
3     def __init__(self, title, pages, price):
4         self.title = title
5         self.pages = pages
6         self.price = price
7
8     1 usage
9     def price_to_usd(self, rate):
10         return self.price * rate
11
12 kniha = Book(title: "Harry Potter", pages: 400, price: 10)
13 print(kniha.price_to_usd(1.09))
```

Run main ×

↑ /Users/patrikdendis/PycharmProjects/oop/ve
↓ 10.9
↺ Process finished with exit code 0
≡
🖨
🗑

DESCRIPTORS

```
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

```
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/Py
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 POZORNOST