OOP: Metody

Podstawy programowania w języku Python

Co to jest metoda?

- jest funkcją osadzoną w klasie
- jest wywoływana na rzecz konkretnego obiektu
- ma dostęp do pól tego obiektu
- obowiązkowym parametrem jest identyfikator obiektu, dla którego wywołana jest metoda (zazwyczaj o nazwie **self**)

```
1  class MyClass:
2
3      def my_method(self):
4          print("To jest metoda")
```

Parametr **self**

```python
class MyClass:

    def my_method(self, arg):
        print("To jest metoda", arg)

obj = MyClass()
obj.my_method(1) #nie podajemy self
```

- jest używany w celu uzyskania dostępu do instancji obiektu i zmiennych klasy
- dla metod jest obowiązkowy
- musi być pominięty przy wywołaniu
- może mieć inną nazwę, ale używanie **self** to dobra praktyka

Konstruktor - metoda __init__()

- to specjalna metoda, której zadaniem jest inicjalizowanie obiektu
- jest wywoływana automatycznie gdy tworzona jest instancja obiektu
- nie może zwracać wartości
- nie może zostać wywołany bezpośrednio z obiektu lub z wewnątrz klasy

```
1  class User:
2
3      def __init__(self, name):
4          self.__name = name
5
6  user = User("Tomek")
```

Wbudowane właściwości klas

- **__name__** - zwraca nazwę klasy jako łańcuch znaków
- **__module__** - zwraca nazwę modułu przechowującego klasę
- **__bases__** - zwraca krotkę zawierającą nadklasy dla klasy

```
1   class MyClass:
2
3       def __init__(self):
4           self.x = 1
5
6   obj = MyClass()
7
8   print(MyClass.__name__) # MyClass
9   print(obj.__module__) # __main__
10  print(MyClass.__bases__) #(<class 'object'>,)
```

Refleksja i introspekcja

```python
class MyClass:

    def __init__(self):
        self.x = 1

obj = MyClass()

print(getattr(obj, "x"))
setattr(obj, "x", 2)
print(obj.x)
```

- introspekcja - zdolność programu do sprawdzenia typu lub właściwości obiektu w środowisku wykonawczym
- refleksja - zdolność programu do manipulowania wartościami, właściwościami i/lub funkcjami obiektu w środowisku wykonawczym

Reprezentacja obiektu – metoda __str__()

- reprezentacja obiektu klasy jako łańcuch znaków
- jest automatycznie wywoływana na obiekcie, gdy jest on argumentem funkcji print() i str()

```
1   class Person:
2
3       def __init__(self, name, age):
4           self.__name = name
5           self.__age = age
6
7       def __str__(self):
8           return "Jestem " + self.__name + " i mam " + str(self.__age) + " lat."
9
10  print(Person("Marta", 30)) # Jestem Marta i mam 30 lat.
```

Pytanie

Na podstawie poniższego kodu wskaż poprawne wywołanie metody:

```
a)  obj.my_method()
b)  my_method(obj, 1)
c)  obj.my_method(self, 1)
d)  obj.my_method(1)
e)  MyClass.my_method(obj, 1)
```

```
1  class MyClass:
2
3      def my_method(self, param):
4          print(param)
5
6  obj = MyClass()
```

Odpowiedź: d)

Przy wywoływaniu metody **my_method()** na obiekcie **obj**, musimy pominąć argument parametru **self**.

Pytanie

Konstruktor klasy to metoda o nazwie:

a) `__str__()`
b) `__dict__()`
c) `__init__()`
d) `__constructor__()`

Odpowiedź: c)

Pytanie

Kiedy wywoływana jest metoda __str__()?

a) gdy wywołujemy funkcję print() lub str() na obiekcie
b) gdy tworzymy obiekt
c) gdy usuwamy obiekt
d) gdy stosujemy enkapsulację

Odpowiedź: a)

Metoda wywoływana jest niejawnie dla obiektu, który przekazujemy jako argument do funkcji print() lub str().