

Programowanie obiektowe

W110PA-SI0054L (INP001045L)

rok akademicki 2023/24

semestr zimowy

Laboratorium 5

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Plan

Przeciążanie operatorów

Przykład

- `__add__`
- `__radd__`
- `__iadd__`

Metody specjalne klas

Przeciążanie operatorów

- Obiekty tworzonych klas mogą wykorzystywać operatory
- Przykładowo, jeśli **a** oraz **b** są instancjami pewnej klasy, to, aby wyrażenie
a + b

miało sensowną wartość, konieczne jest przeciążenie operatora dodawania

Przeciążanie operatorów

Przykład - dodawanie dwóch punktów

```
point2d_operators.py X point2d_21.py X
1  """
2  Program ilustrujący działanie klasy z przeciążonym
3  operatorem dodawania.
4  """
5  class Point2d_Operators:
6
7      def __init__(self):
8          self.__x = 0.
9          self.__y = 0.
10
11     def get_coordinates(self):
12         return [ self.__x, self.__y ]
13
14     def set_xy(self,x,y):
15         self.__x = x
16         self.__y = y
17
18     # definicja metody __add__, która jest wywoływana
19     # przy obliczaniu wyrażenia self + other
20     def __add__(self, other):
21         print("__add__")
22         r = Point2d_Operators()
23         r.set_xy(self.__x + other.__x, self.__y + other.__y)
24         return r
25
26     def __str__(self):
27         return str(self.get_coordinates())
28
```

```
point2d_operators.py X point2d_21.py X
28
29     def main():
30         # zmienna p1 - punkt o współrzędnych [3., 4.]
31         p1 = Point2d_Operators()
32         p1.set_xy(3.,4.)
33
34         # zmienna p2 - punkt o współrzędnych [2., 0.7]
35         p2 = Point2d_Operators()
36         p2.set_xy(2.,0.7)
37
38         print( p1 + p2 )
39
40     if __name__ == '__main__':
41         main()
42
```


```
repozytorium.py
__add__
[5.0, 4.7]
```

Przeciążanie operatorów

Przykład - obsługa różnych typów

```
34 def main():
35     # zmienna p1 - punkt o współrzędnych [3., 4.]
36     p1 = Point2d_Operators()
37     p1.set_xy(3.,4.)
38
39     # zmienna p2 - punkt o współrzędnych [2., 0.7]
40     p2 = Point2d_Operators()
41     p2.set_xy(2.,0.7)
42
43     print( p1 + p2 )
44     print( p1 + 3. )
45     print( p1 + 3 )
46
```

```
# definicja metody __add__, która jest wywoływana
# przy obliczaniu wyrażenia self + other
def __add__(self, other):
    print("__add__")
    if type(other) == type(self):
        r = Point2d_Operators()
        r.set_xy(self.__x + other.__x, self.__y + other.__y)
        return r
    elif type(other) == int or type(other) == float:
        t = Point2d_Operators()
        t.set_xy(float(other), 0.0)
        return self + t
```



Przeciążanie operatorów

Przykład - zgłaszanie wyjątku

```
17
18     # definicja metody __add__, która jest wywoływana
19     # przy obliczaniu wyrażenia self + other
20     # zgłasza wyjątek TypeError gdy typ other jest inny niż
21     # Point2d_Operators, int, float
22     def __add__(self, other):
23         print("__add__")
24         if type(other) == type(self):
25             r = Point2d_Operators()
26             r.set_xy(self.__x + other.__x, self.__y + other.__y)
27             return r
28         elif type(other) == int or type(other) == float:
29             t = Point2d_Operators()
30             t.set_xy(float(other), 0.0)
31             return self + t
32         else:
33             raise TypeError
34
```

Przeciążanie operatorów

Przykład - różne typy w odwróconej kolejności

```
41
42  def main():
43      # zmienna p1 - punkt o współrzędnych [3., 4.]
44      p1 = Point2d_Operators()
45      p1.set_xy(3.,4.)
46
47      # zmienna p2 - punkt o współrzędnych [2., 0.7]
48      p2 = Point2d_Operators()
49      p2.set_xy(2.,0.7)
50
51      print( p1 + p2 )
52      print( p1 + 3. )
53      print( p1 + 3 )
54
55      print( 3. + p1 )
56
```

```
34
35
36
37
38
```

```
def __radd__(self, other):
    print("__radd__")
    return self.__add__(other)
```

Przeciążanie operatorów

Przykład - złożony operator przypisania

```
39     def __iadd__(self, other):
40         print("__iadd__")
41         if type(other) == type(self):
42             self.__x += other.__x
43             self.__y += other.__y
44             return self
45         elif type(other) == int or type(other) == float:
46             self.__x += other
47             return self
48         else:
49             raise TypeError
50
```


Przeciążanie operatorów

Metody specjalne powiązane z operatorami

```
object.__add__(self, other)
object.__sub__(self, other)
object.__mul__(self, other)
object.__matmul__(self, other)
object.__truediv__(self, other)
object.__floordiv__(self, other)
object.__mod__(self, other)
object.__divmod__(self, other)
object.__pow__(self, other[, modulo])
object.__lshift__(self, other)
object.__rshift__(self, other)
object.__and__(self, other)
object.__xor__(self, other)
object.__or__(self, other)
```

[https://docs.python.org/pl/3/reference/
datamodel.html#emulating-numeric-types](https://docs.python.org/pl/3/reference/datamodel.html#emulating-numeric-types)

Przeciążanie operatorów

Metody specjalne powiązane z operatorami relacji

```
object.__lt__(self, other)  
object.__le__(self, other)  
object.__eq__(self, other)  
object.__ne__(self, other)  
object.__gt__(self, other)  
object.__ge__(self, other)
```

Wybrane metody specjalne

```
__abs__  
__len__  
__repr__
```

Podsumowanie

Przeciążanie operatorów

Metody specjalne klas