Politechnika Wrocławska

Programowanie obiektowe

INP001045L

rok akademicki 2022/23

semestr zimowy

Laboratorium 6

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan

- Przeciążanie operatorów
- Przykład
  - __add__
  - __radd__
  - __iadd__
- Metody specjalne klas

Politechnika Wrocławska

Przeciążanie operatorów

- Obiekty tworzonych klas mogą wykorzystywać operatory
- Przykładowo, jeśli **`a`** oraz **`b`** są instancjami pewnej klasy, to konieczne jest przeciążenie operatora dodawania tak, aby wyrażenie

  `a + b`

  miało sensowną wartość

Przeciążanie operatorów

- Przykład – dodawanie dwóch punktów

point2d_operators.py

```
"""
Program ilustrujący działanie
operatorem dodwania.
"""
class Point2d_Operators:

    def __init__(self):
        self.__x = 0.
        self.__y = 0.

    def get_coordinates(self):
        return [ self.__x,  s

    def set_xy(self,x,y):
        self.__x = x
        self.__y = y

    # definicja metody __add_
    # przy obliczaniu wyrażen
    def __add__(self, other):
        print("__add__")
        r = Point2d_Operators()
        r.set_xy(self.__x + other.__x, self.__y + other.__y)
        return r

    def __str__(self):
        return str(self.get_coordinates())
```

point2d_21.py

```

def main():
    # zmienna p1 - punkt o współrzędnych [3., 4.]
    p1 = Point2d_Operators()
    p1.set_xy(3.,4.)

    # zmienna p2 - punkt o współrzędnych [2., 0.7]
    p2 = Point2d_Operators()
    p2.set_xy(2.,0.7)

    print( p1 + p2 )

if __name__ == '__main__':
    main()
```

```
__add__
[5.0, 4.7]
```

Przeciążanie operatorów

- Przykład – dodawanie dwóch punktów z obsługą 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
```

Politechnika Wrocławska

Przeciążanie operatorów

- Przykład – dodawanie dwóch punktów z obsługą różnych typów (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 – operator dla różnych typów w zamienionej 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      def __radd__(self, other):
36          print("__radd__")
37          return self.__add__(other)
38
```

Politechnika Wrocławska

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
```

Politechnika Wrocławska

Specjalne metody klas

- Klasy mają 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

Politechnika Wrocławska

Specjalne metody klas

- Podobnie operatory 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)
```

- Dla wielu innych operacji istnieją metody specjalne (np. :**`__abs__`**, **`__len__`**, **`__repr__`**)

Politechnika Wrocławska

Podsumowanie

- Przeciążanie operatorów
- Metody specjalne klas