



Politechnika
Wrocławska

Programowanie obiektowe

W11FTE-SI0080G (INP001129WL)

rok akademicki 2023/24

semestr letni

Wykład 4

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220



Plan

Klasa a obiekt

Przykłady

- definicja klasy
- pola i metody
- metody setter/getter
- pola prywatne
- definicja klasy w module

Klasy

- Klasa to ogólny przepis na stworzenie obiektu
- Obiekt (instancja) to pojedynczy egzemplarz danej klasy

Klasy

- Definicja klasy - przykład

```
point2d_01.py X
6  # definicja klasy Point2d
7  class Point2d:
8
9      # definicja konstruktora
10     def __init__(self):
11         self.x = 0.
12         self.y = 0.
13
14     def main():
15         # utworzenie instancji klasy
16         my_point = Point2d()
17         # bezpośredni odczyt pól klasy
18         print('współrzędne punktu: [' ,
19               my_point.x, ', ',
20               my_point.y, ']', sep = ' ')
21
22     if __name__ == '__main__':
23         main()
24
```

Klasy

- Definicja klasy - przykład

```
point2d_01.py X
6  # definicja klasy Point2d
7  class Point2d:
8
9      # definicja konstruktora
10     def __init__(self):
11         self.x = 0.
12         self.y = 0.
13
14     def main():
15         # utworzenie instancji klasy
16         my_point = Point2d()
17         # bezpośredni odczyt pól klasy
18         print('współrzędne punktu: [' ,
19               my_point.x, ', ',
20               my_point.y, ']', sep = ' ')
21
22     if __name__ == '__main__':
23         main()
24
```

inicjalizacja pól klasy

Klasy

- Klasa zawiera pola i metody

```
point2d_02.py X
5  import random
6
7  class Point2d:
8
9      def __init__(self):
10         self.x = 0.
11         self.y = 0.
12
13         #definicja metody random()
14         def random(self):
15             """
16             Meotda nadaje współrzędnym punktu
17             losowe wartości
18
19             """
20             self.x = random.uniform(0,1)
21             self.y = random.uniform(0,1)
22
```

definicja
metody

Klasy

- Wywołanie metody

```
22
23     def main():
24         my_point = Point2d()
25         print('współrzędne punktu: [' ,
26               my_point.x, ' , ' , my_point.y, ']', sep = '')
27         #wywołanie metody
28         my_point.random()
29         print('współrzędne punktu: [' ,
30               my_point.x, ' , ' , my_point.y, ']', sep = '')
31
```

Klasy

- Do nadawania wartości polom klasy wykorzystuje się odpowiednie metody („setter” / „getter”)

```
point2d_04.py X
5  import random
6
7  class Point2d:
8
9      def __init__(self):
10         self.x = 0.
11         self.y = 0.
12
13     def random(self):
14         self.x = random.uniform(0,1)
15         self.y = random.uniform(0,1)
16
17     #definicja metody
18     def get_coordinates(self):
19         return [self.x, self.y]
20
```

odczytywanie wartości
pól (getter)

Klasy

- Do nadawania wartości polom klasy wykorzystuje się odpowiednie metody („setter” / „getter”)

```
21 def main():  
22     my_point = Point2d()  
23     #wywołanie metody get_coordinates()  
24     print('współrzędne punktu: ',  
25           my_point.get_coordinates(), sep = '')
```

odczytywanie wartości
pól (getter)

Klasy

- Nazwy pól prywatnych oznaczają się dwoma podkreślnikami

```
point2d_05.py X
4  import random
5
6  class Point2d:
7
8      def __init__(self):
9          #inicjalizacja pól prywatnych
10         self.__x = 0.
11         self.__y = 0.
12
13         def random(self):
14             self.__x = random.uniform(0,1)
15             self.__y = random.uniform(0,1)
16
17         def get_coordinates(self):
18             return [self.__x, self.__y]
```

Klasy

- Nazwy pól prywatnych oznaczają się dwoma podkreślnikami

```
point2d_05.py X point2d_06.py X
20 def main():
21     # dane w klasie obsługiwane
22     # są tylko za pomocą metod
23     my_point = Point2d()
24     print('współrzędne punktu: ',
25           my_point.get_coordinates(), sep = '')
26     my_point.random()
27     print('współrzędne punktu: ',
28           my_point.get_coordinates(), sep = '')
29     # poniższa instrukcja powoduje błąd
30     # print(my_point.__x)
31
32
33 if __name__ == '__main__':
34     main()
35
```

Klasy

- Nazwy pól prywatnych oznaczają się dwoma podkreślnikami

```
point2d_06.py X
19 def main():
20     my_point = Point2d()
21     print('współrzędne punktu: ',
22           my_point.get_coordinates(), sep = '')
23     my_point.random()
24     # poniższe instrukcje nie dają dostępu do
25     # prywatnych pól klasy
26     my_point.__x = .5
27     my_point.__y = .5
28     print('współrzędne punktu: ',
29           my_point.get_coordinates(), sep = '')
30
```

Klasy

- Przykład metody nadającej wartości polom klasy

```
point2d_07.py X
5  import random
6
7  class Point2d:
8
9      def __init__(self):
10         self.__x = 0.
11         self.__y = 0.
12
13     def random(self):
14         self.__x = random.uniform(0,1)
15         self.__y = random.uniform(0,1)
16
17     def get_coordinates(self):
18         return [self.__x, self.__y]
19
20     #definicja metody
21     def set_xy(self,x,y):
22         self.__x = x
23         self.__y = y
```

nadawanie wartości pól
(setter)

Klasy

- Przykład metody nadającej wartości polom klasy

```
point2d_07.py X
24
25 def main():
26     my_point = Point2d()
27     print('współrzędne punktu: ',
28           my_point.get_coordinates(), sep = '')
29     #wywołanie metody
30     my_point.set_xy(.5, .5)
31     print('współrzędne punktu: ',
32           my_point.get_coordinates(), sep = '')
```

nadawanie wartości pól
(setter)

Klasy

- Wykorzystanie metody klasy przez inne metody klasy

```
point2d_08.py X
5  import random
6
7  class Point2d:
8
9      def __init__(self):
10         self.__x = 0.
11         self.__y = 0.
12
13         #metoda random wywołuje metodę set_xy
14     def random(self):
15         self.set_xy(random.uniform(0,1), random.uniform(0,1))
16
17     def get_coordinates(self):
18         return [self.__x, self.__y]
19
20     def set_xy(self,x, y):
21         self.__x = x
22         self.__y = y
23
```

Klasy

- Definicja klasy może być umieszczona w osobnym module

```
point2d.py X point2d_09.py X
4 import random
5
6 class Point2d:
7
8     def __init__(self):
9         self.__x = 0.
10        self.__y = 0.
11
12    def random(self):
13        self.set_xy(random.uniform(0,1),
14
15    def get_coordinates(self):
16        return [ self.__x, self.__y ]
17
18    def set_xy(self,x,y):
19        self.__x = x
20        self.__y = y
21
22    #funkcja implementuje konwersję obiektu na string
23    def __str__(self):
24        return str(self.get_coordinates())
25
```

```
point2d.py X point2d_09.py X
5 import point2d
6
7 def main():
8     my_point = point2d.Point2d()
9     # obiekt klasy jako argument funkcji print
10    # do wyświetlenia wykorzystywana jest metoda __str__()
11    print('współrzędne punktu: ', my_point)
12    my_point.set_xy(.5, .5)
13    print('współrzędne punktu: ', my_point)
14
15
16 if __name__ == '__main__':
17     main()
18
19
```


Cechy programowania obiektowego

abstrakcja

hermetyzacja

polimorfizm

dziedziczenie

Podsumowanie

Klasa a obiekt

Przykłady

- Definicja klasy
- Pola i metody
- Metody setter/getter
- Pola prywatne
- Definicja klasy w module