

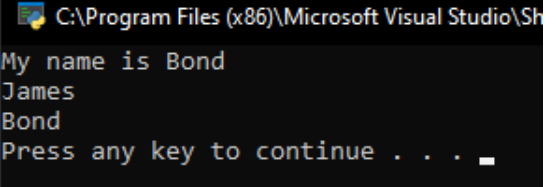
## Wstęp do programowania obiektowego (OOP – object-oriented programming)

Programowanie obiektowe w Pythonie jest możliwe dzięki klasom i instancjom. Używa się w nich atrybuty i metody. Klasa służy do tworzenia instancji, czyli obiektów. Klasy możemy tworzyć w tym samym lub innym pliku.

Utworzenie klasy w Python w pliku głównym:

```
KlasyWpython.py  ➤ ✕

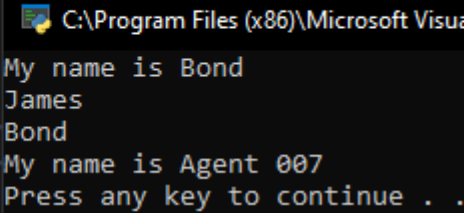
1  class Osoba: #klasu musi mieć nazwę z dużej litery
2      nazwisko = 'Bond'
3      imię = 'James'
4      #metoda tworzona w klasie jest funkcją
5      def identyfikujSie(self): # słówko 'self' odsyła nas do własnej klasy
6          nazwisko = self.nazwisko
7          return "My name is " + nazwisko
8      #utworzenie obiektu klasy Osoba
9      kumpel = Osoba() # konstruktor obiektu
10     print(kumpel.identyfikujSie())
11     print(kumpel.imię)
12     print(kumpel.nazwisko)
13
```



```
C:\Program Files (x86)\Microsoft Visual Studio\Sh
My name is Bond
James
Bond
Press any key to continue . . .
```

Możemy również stworzyć kolejny obiekt i nadać mu nowe wartości:

```
9      kumpel = Osoba() # konstruktor obiektu
10     print(kumpel.identyfikujSie())
11     print(kumpel.imię)
12     print(kumpel.nazwisko)
13     brachu = Osoba()
14     brachu.nazwisko = "Agent 007"
15     print(brachu.identyfikujSie())
```



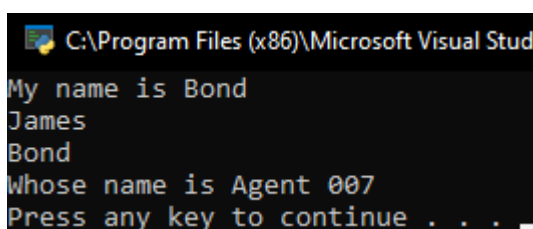
```
C:\Program Files (x86)\Microsoft Visual
My name is Bond
James
Bond
My name is Agent 007
Press any key to continue . . .
```

Do metody możemy wysłać argumenty. W tym celu należy zmodyfikować metodę:

```
5      def identyfikujSie(self, kto = "My"): #
6          nazwisko = self.nazwisko
7          return kto + " name is " + nazwisko

13     brachu = Osoba()
14     brachu.nazwisko = "Agent 007"
15     print(brachu.identyfikujSie("Whose")) # wysyłamy argument
```

Wynik zmian:



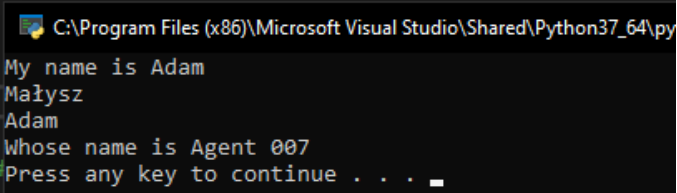
```
C:\Program Files (x86)\Microsoft Visual Stud
My name is Bond
James
Bond
Whose name is Agent 007
Press any key to continue . . .
```

Modyfikacja klasy w celu możliwości wymuszenia podawania argumentów w konstruktorze obiektu:

```
1 class Osoba: #klasa musi mieć nazwę z dużej litery
2     def __init__(self,nazwisko,imie,wiek): # __init__ wymusi wprowadzanie argumentów
3         self.nazwisko = nazwisko
4         self.imie = imie
5         self.wiek = wiek
```

c.d. (musimy podać atrybuty obiektu) :

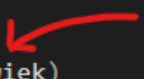
```
6 #metoda tworzona w klasie jest funkcją
7 def identyfikujSie(self, kto = "My"): # słówko 'self' odsyła nas do własnej klasy
8     nazwisko = self.nazwisko
9     return kto + " name is " + nazwisko
10 #utworzenie obiektu klasy Osoba
11 kumpel = Osoba("Adam","Małysz","55") # konstruktor obiektu
12 print(kumpel.identyfikujSie())
13 print(kumpel.imie)
14 print(kumpel.nazwisko)
15 brachu = Osoba("Olaf","Lubaszenko",60)
16 brachu.nazwisko = "Agent 007"
17 print(brachu.identyfikujSie("Whose"))
```



```
My name is Adam
Małysz
Adam
Whose name is Agent 007
Press any key to continue . . .
```

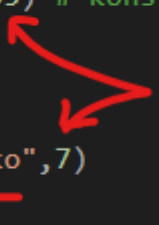
W celu obsłużenia zmiennej 'wiek' musimy zmodyfikować naszą główną metodę – potrzebne jest rzutowanie int -> str :

```
6 #metoda tworzona w klasie jest funkcją
7 def identyfikujSie(self, kto = "My"): # słówko 'self' odsyła nas do własnej klasy
8     nazwisko = self.nazwisko
9     wiek = self.wiek
10    return kto + " name is " + nazwisko + str(wiek)
```



Użycie wartości 'wiek' jako atrybutu:

```
11 #utworzenie obiektu klasy Osoba
12 kumpel = Osoba("Adam","Małysz",55) # konstruktor obiektu
13 print(kumpel.identyfikujSie())
14 print(kumpel.imie)
15 print(kumpel.nazwisko)
16 brachu = Osoba("Olaf","Lubaszenko",7)
17 brachu.nazwisko = "Agent 00"
18 print(brachu.identyfikujSie("Whose")) # wysyłamy argument
```



Podsumowując: metoda \_\_init\_\_ w sposób dynamiczny tworzy zmienne pobierane z konstruktora obiektu. Dzięki temu zawsze będziemy tworzyć obiekty ze wszystkimi potrzebnymi atrybutami.