Klasy

- Definicja klasy oznacza definicję nowego typu danych.
- Tworząc klasy w Pythonie w znakomitej większości przypadków nie definiujemy sami konstruktora. Konstruktor jest metodą specjalną o nazwie __new__ i jej domyślna wersja jest wywoływana automatycznie w momencie tworzenia obiektów klas.
- Pojawiająca się w klasach metoda __init__, jest metodą inicjalizacyjną wywoływaną automatycznie po utworzeniu obiektu. Powinna mieć w niej miejsce inicjalizacja stanu obiektu. Jest to ważne, bo chociaż formalnie taka inicjalizacja może mieć miejsce w dowolnej metodzie klasy lub spoza niej, to jednak jej działanie może być wtedy uzależnione od kolejności wywoływania metod.
- Pierwszy parametr metod, w ramach konwencji nazywający się self, odnosi się do przetwarzanego obiektu.

Przykład I

```
class A:
    zmiennaKlasy=0

    def __init__(self, p):
        self.zmiennaInstancji=p
        A.zmiennaKlasy+=1

    def metoda(self, p):
        zmiennaMetody=p
        self.zmiennaInstancji+=p

    def __str__(self):
        return f'zmiennaKLasy={A.zmiennaKlasy},\
                zmiennaInstancji={self.zmiennaInstancji}'


a=A(8)
a.metoda(5)                 #A.metoda(a,6)
print(a)                    #zmiennaKLasy=1, zmiennaInstancji=13
b=A(3)
b.metoda(4)
print(b)                    #zmiennaKLasy=2, zmiennaInstancji=7
```

Uwaga

```
class A:
  def met(self, p):
    print(p)
```

```
a=A()

A.met(a,3)

a.met(3)
```

Sterowanie dostępem

- W Pythonie nie ma możliwości określenia modyfikatora dostępu.
- W ramach konwencji nazwa dowolnej składowej poprzedzona znakiem podkreślenia uważana jest za zmienną prywatną.
- Nazwy poprzedzone co najmniej dwoma znakami podkreślenia i co najwyżej jednym znakiem podkreślenia, jak np. __nazwa zamieniane są wewnętrznie na _nazwaKlasy__nazwa.

str

- Podkreślenia przed i po nazwie atrybutu pełnią określoną funkcję.

- Metody, których nazwy są poprzedzone oraz zakończone dwoma znakami podkreślenia są metodami specjalnymi. We wcześniejszym przykładzie pojawiły się dwie takie nazwy: __init__ oraz __str__
  Pierwsza, pozwala na określenie stanu obiektu, druga natomiast jest przesłonięciem funkcji **str** i jest wywoływana automatycznie w momencie przekazania obiektu jako parametru do funkcji **print**.

- Poprawna definicja metody __str__ wymaga przekazywania jako parametru jedynie aktualnego obiektu oraz zwracania obiektu typu string.

- Bardziej zaawansowaną metodą jest metoda __repr__, która powinna pozwolić na odtworzenie stanu obiektu. Jeśli w klasie nie jest zdefiniowana metoda __str__ będzie ona wywoływana przy wypisywaniu komunikatu na ekran.

Przeciążanie operatorów

- W Pythonie możliwe jest przeciążanie operatorów oraz funkcji wbudowanych.

- np. przeciążenie operatora dodawania będzie oznaczało zdefiniowanie w klasie metody __add__. Operator dodawania jest operatorem dwuargumentowym, przy jego wywołaniu obiekt klasy znajduje się po jego lewej stronie (metodę wywołujemy na obiekcie) a drugi parametr przekazywany jest do metody i niekoniecznie musi być on obiektem definiowanej przez nas klasy.

Przykład

```
class A:
    def __init__(self, a):
        self.a=a

    def __add__(self, p):
        return self.a+p


a=A(3)
print(a+6)        #9
print(6+a)
    #TypeError: unsupported operand type(s) for +: 'int' and 'A'
```

- Wyjątek, który się pojawił wynika z tego, że operator (odpowiednia metoda zdefiniowana w klasie) wywoływany jest na obiekcie znajdującym się po jego lewej stronie.

- Dla typu int, nie ma zdefiniowanego operatora dodawania z obiektem tworzonej przez nas klasy.

- Żeby wyjątek się nie pojawił w klasie trzeba zdefiniować dodatkowo metodę __raad__.

- Jeżeli działanie ma być „symetryczne" względem operatora wystarczy w klasie wpisać: __radd__=__add__.

Metody przeciążające podstawowe operatory

+	__add__/__radd__
+=	__iadd__
-	__sub__/__rsub__
-=	__isub__
*	__mul__/__rmul__
*=	__imul__
/	__truediv__/__rtruediv__
/=	__itruediv__
//	__floordiv__/__rfloordiv__
//=	__ifloordiv__
%	__mod__/__rmod__
%=	__imod__
**	__pow__/__rpow__
**=	__ipow__

Szczegółowe informacje o metodach specjalnych dostępne są w dokumentacji:
https://docs.python.org/3/reference/datamodel.html (rozdział 3.3)