KURS PROGRAMOWANIA W JĘZYKU PYTHON

TYDZIEŃ 7 – DZIEDZICZENIE KLAS

1. Dziedziczenie klas

Klasy mają wbudowaną w język Python funkcję **dziedziczenia parametów i metod** w celu ich nadpisania lub dodania istniejącej już funkcjonalności w łatwy sposób do klasy, którą piszemy. Do klasy docelowej przenoszone są ***wszystkie*** metody i parametry, włącznie z metodami specjalnymi takimi jak ***__init__*** czy ***__str__***. Dziedziczenie zachodzi poprzez podanie w nawiasie do nazwy klasy którą tworzymy nazwę klasy z której chcemy aby dziedziczone były parametry i metody. Załóżmy że mamy napisaną klasę w następujący sposób:

```python
# Definiujemy klase matke
class KlasaMatka:
    def __init__(self, parametr1, parametr2, parametr3):
        # Pozwól użytkownikowi wprowadzić 3 parametry oraz miej swoje 2 przypisane z góry
        self.parametr1 = parametr1
        self.parametr2 = parametr2
        self.parametr3 = parametr3
        self.parametr_matka1 = "Matka1"
        self.parametr_matka2 = "Matka2"

    # Jakieś metody
    def metoda1(self):
        print("Wykonywanie metody 1")

    def metoda2(self):
        print("Wykonywanie metody 2")

    # Skomplikowana funkcja do wypisywania informacji o klasie
    def __str__(self):
        # Pokaż nazwe klasy
        s1 = f"Klasa {self.__class__.__name__} o wprowadzonych parametrach"
        # Wygeneruj i pokaż spis parametrów klasy
        s2 = f"{', '.join([f'{_1}: {_2, type(_2).__name__}' for _1, _2 in vars(self).items()])}"
        # Połącz wygenerowane stringi w jedną całość z odstępem jednej spacji
        return ' '.join([s1, s2])
```

Nasza klasa macierzysta przyjmuje 3 parametry przy przypisaniu do obiektu, posiada 2 parametry wbudowane, dwie metody oraz skomplikowaną funkcję wypisującą informacje o niej. Jeśli przypiszemy ją do obiektu ***obiekt1*** i wykonamy zaimplementowane metody uzyskamy:

```
obiekt1 = KlasaMatka( parametr1: 123,  parametr2: '456',  parametr3: [7, 8, 9])
print("Wykonywanie obiekt1.metoda1()")
obiekt1.metoda1()
print("Wykonywanie obiekt1.metoda2()")
obiekt1.metoda2()
print("Wypisywanie informacji o obiekcie klasy:")
print(obiekt1)
```

```
Wykonywanie obiekt1.metoda1()
Wykonywanie metody 1
Wykonywanie obiekt1.metoda2()
Wykonywanie metody 2
Wypisywanie informacji o obiekcie klasy:
Klasa KlasaMatka o wprowadzonych parametrach parametr1: (123, 'int'), parametr2: ('456', 'str'),
 parametr3: ([7, 8, 9], 'list'), parametr_wlasny: ('Coś', 'str'), parametr_wlasny2: ('Coś 2', 'str')

Process finished with exit code 0
```

Teraz, jeśli chcemy przejąć funkcjonalność naszej klasy macierzystej, musimy zdefiniować nową klasę, w której również znajdzie się zdefiniowany ***__init__***. W inicjatorze klasy na samym początku musimy przejąć wszystko z klasy macierzystej używając wbudowanej funkcji ***super().__init__(parametry_z__init__KlasyMacierzystej),*** która za nas przeniesie odpowiednie rzeczy. Dodamy do niej jeden parametr czytany od użytkownika oraz 2 parametry wbudowane, oraz nadpiszemy metodę ***metoda2***. Wygląda to następująco:

```
# Klasa Dziecko dziedzicząca parametry z klasy KlasaMatka ale pobierający 4 zamiast 3 parametrów od użytkownika
class KlasaDziecko(KlasaMatka):
    def __init__(self, param1, param2, param3, param4):
        # super() inicjuje wszystkie parametry i metody z klasy macierzystej do klasy dziedziczącej.
        super().__init__(param1, param2, param3)
        self.parametr4 = param4
        self.parametr_dziecko = "Dziecko 1"
        self.parametr_dziecko2 = "Dziecko 2"
    # nadpisanie istniejącej w klasie macierzystej metody metoda2
    def metoda2(self):
        print("Nadpisana metoda 2 wykonana!")
```

Należy zauważyć że nie przepisaliśmy metod ***__str__*** i ***metoda1*** do naszej klasy. Te metody zostały automatycznie dodane do klasy KlasaDziecko przez Python. Teraz jeśli wykonamy podane wyżej operacje na obiekcie ***obiekt2*** odwzorowującym klasę KlasaDziecko uzyskamy:

```
obiekt2 = KlasaDziecko( param1: 456,  param2: '789',  param3: [10, 11, 12],  param4: "Czwarty!")
print("Wykonywanie obiekt2.metoda1()")
obiekt2.metoda1()
print("Wykonywanie obiekt2.metoda2()")
obiekt2.metoda2()
print("Wypisywanie informacji o obiekcie klasy:")
print(obiekt2)
```

```
Wykonywanie obiekt2.metoda1()
Wykonywanie metody 1
Wykonywanie obiekt2.metoda2()
Nadpisana metoda 2 wykonana!
Wypisywanie informacji o obiekcie klasy:
Klasa KlasaDziecko o wprowadzonych parametrach parametr1: (456, 'int'), parametr2: ('789', 'str'), parametr3:
  ([10, 11, 12], 'list'), parametr_matka1: ('Matka1', 'str'), parametr_matka2: ('Matka2', 'str'), parametr4:
 ('Czwarty!', 'str'), parametr_dziecko: ('Dziecko 1', 'str'), parametr_dziecko2: ('Dziecko 2', 'str')

Process finished with exit code 0
```

Nadpisywanie metod następuje wtedy, gdy nazwa metody w klasie dziedziczącej jest taka sama jak w klasie macierzystej/dziedziczonej, funkcjonalność tej metody zostaje nadpisana kodem spod definicji w klasie dziedziczącej tylko i wyłącznie jeśli obiekt reflektuje klasę dziedziczącą nadpisywaną metodę z klasy macierzystej. Prościej – Jeśli mamy takie same metody w klasie którą dziedziczymy jak i w nowo pisanej klasie, efekt wykonania metody będzie bazował tylko i wyłącznie na metodzie z nowej, dziedziczącej klasy. Takie same zachowanie dotyczy parametrów naszych klas, czyli jeśli klasa dziedziczona ma jakiś parametr który powtarza się w klasie dziedziczącej, klasa dziedzicząca go nadpisze.