CYFROWA SZKOŁA
Wielkopolsk@ 2020

Dziedziczenie i polimorfizm

Wprowadzenie do języka Python (XVIII)

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
# Kiedy tworzymy klasę, możemy dziedziczyć wszystkie pola i metody
z innej klasy

# To się nazywa dziedziczenie

# Klasa, która dziedziczy nazywa się podklasą, a klasa, z której
dziedziczymy, jest superklasą

# To będzie nasza super klasa: klasa Zwierze:

class Zwierze:

    def __init__(self, urodzenie="Unknown", wyglad="Unknown",
uklad_krwionosny="Unknown"):

        self.__urodzenie = urodzenie

        self.__wyglad = wyglad

        self.__uklad_krwionosny = uklad_krwionosny
```

```
# użycie gettera


    @property
    def urodzenie(self):
        # Przy użyciu getterów i setterów nie należy zapominać o
znaku  '__' w definicji zmiennych prywatnych
        return self.__urodzenie


    @urodzenie.setter
    def urodzenie(self, urodzenie):
        self.__urodzenie = urodzenie
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
@property
    def wyglad(self):
        return self.__wyglad

    @wyglad.setter
    def wyglad(self, wyglad):
        self.__wyglad = wyglad

    @property
    def uklad_krwionosny(self):
        return self.__uklad_krwionosny
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
    @uklad_krwionosny.setter
    def uklad_krwionosny(self, uklad_krwionosny):
        self.__uklad_krwionosny = uklad_krwionosny
        # Metoda może być użyta w celu przedstawienia obiektu jako
ciągu znakowego
        # type(self).__name__ zwraca nazwę klasy
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
def __str__(self):
        return "Gatunek {} jest {} posiada {} oraz jest " \
                "{}".format(type(self).__name__,
                            self.urodzenie,
                            self.wyglad,
                            self.uklad_krwionosny)
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
# Klasa Sssak, która dziedziczy z klasy Zwierze
# Można dziedziczyć z wielu klas używając przecinka, a
# nazwy dziedziczonych klas umieszczając w nawiasach okrągłych
class Ssak(Zwierze):
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
def __init__(self, urodzenie="urodzony żywy",
                   wyglad="włosy lub futro",
                   uklad_krwionosny="stałocieplny",
                   opieka_nad_mlodymi=True):
        # Wywołanie super klasy do inicjalizacji pól obiektu
        Zwierze.__init__(self, urodzenie,
                               wyglad,
                               uklad_krwionosny)
        self.__opieka_nad_mlodymi = opieka_nad_mlodymi
```

```
# Możemy rozszerzyć superklasę
    @property
    def opieka_nad_mlodymi(self):
        return self.__opieka_nad_mlodymi

    @opieka_nad_mlodymi.setter
    def opieka_nad_mlodymi(self, opieka_nad_mlodymi):
        self.__opieka_nad_mlodymi = opieka_nad_mlodymi
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
    # Nadpisanie metody __str__
        # Można używać wywołania super() aby odwołać się do super klasy
        def __str__(self):
            return super().__str__() + " i stąd {} opiekują się " \
                                       "swoimi
młodymi".format(self.opieka_nad_mlodymi)
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
class Gad(Zwierze):
    def __init__(self, urodzenie="urodzony z jajka lub urodzony
żywy",
                 wyglad="suche łuski",
                 uklad_krwionosny="zimnocieplny"):
        # Wywołanie super klasy do inicjalizacji pól obiektu
        Zwierze.__init__(self, urodzenie,
                         wyglad,
                         uklad_krwionosny)
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
def main():
    zwierze1 = Zwierze("urodzony żywy")
    print(zwierze1.urodzenie)
    # Call __str__()
    print(zwierze1)
    print()
    ssak1 = Ssak()
    print(ssak1)
    print(ssak1.urodzenie)
```

CYFROWA SZKOŁA
Wielkopolsk@ 2020

```
print(ssak1.wyglad)
    print(ssak1.uklad_krwionosny)
    print(ssak1.opieka_nad_mlodymi)
    print()
    gad1 = Gad()
    print(gad1.urodzenie)
    print(gad1.wyglad)
    print(gad1.uklad_krwionosny)


main()
```