Laboratorium 8 — Ćwiczenia w programowaniu obiektowym. Tworzenie własnych typów danych.

Klasy, metody, pola

Python jako język obiektowy pozwala na tworzenie klas, przy czym występują tu znaczne różnice w porównaniu do języków takich jak $C++$. W celu zdefiniowania nowej klasy posługujemy się słówkiem `class`:

```
class Rectangle:
    pass
```

Pierwsza różnica pojawią się przy konstruktorze, funkcją specjalną najbardziej swoim działaniem przypominającą konstruktor jest `__init__`. Przy czym metoda ta wywoływana jest już po utworzeniu obiektu. Każda metoda (funkcja klasy), w tym odpowiednik konstruktora posiada minimalnie jeden parametr będący referencją na obiekt (typowo parametr ten nazywa się `self`). Kolejną różnicą jest to, że metoda `__init__` może być tylko jedna i nie ma możliwości jej przeciążania.

```
class Rectangle:
    def __init__(self):
        pass

rectangle = Rectangle() # utworzenie obiektu
```

Aby utworzyć pole klasy posługujemy się przekazaną referencją `self`. Domyślnie pola są publiczne. Jeśli chcemy, aby jakaś zmienna była prywatną, należy jej nazwę poprzedzić `__`, przy czym prywatność ta jest tylko umowna, nadal można uzyskać dostęp do takiej zmiennej z zewnątrz klasy, wystarczy wiedzieć do jakiej postaci jest ona konwertowana.

```
class Rectangle:
    def __init__(self):
        self.var1 = 5
        self.__var2 = 10

rectangle = Rectangle() # utworzenie obiektu
print(rectangle.var1)  # wyswietlnie publicznej zmiennej
print(rectangle._Rectangle__var2)  # obejscie prywatnosci zmiennej
```

Aby utworzyć metodę statyczną należy posłużyć się dekoratorem `@staticmethod`, a jeśli chcemy, aby zmienna istniała na poziomie klasy (pole statyczne) musi zostać utworzona poza konstruktorem.

```
class Rectangle:
    pole_statyczne = 0

    def __init__(self):
        self.var1 = 5

    @staticmethod
    def metoda2():
        Rectangle.pole_statyczne += 1  # odwolanie do pola statycznego
```

Oprócz metod statycznych i metod działających na poziomie obiektu, Python posiada trzeci rodzaj metod. Metody klasy oznaczone dekoratorem `@classmethod` jako pierwszy argument zawsze przyjmują klasę. Metody te pozwalają np. na obejście limitu konstruktorów.

```
class Rectangle:
    def __init__(self, a, b):
        pass

    def metoda1(self):
        pass
```

```
    @staticmethod
    def metoda2():
        pass

     @classmethod
     def createFromStr(cls, a, b):
         return cls(int(a), int(b))

rectangle = Rectangle.createFromStr('3', '5')
```

Jak widać w przykładzie, zastosowanie metody klasowej pozwoliło na obejście problemu utworzenia konstruktorów dla różnych typów danych. Zalety wykorzystania tych metod do tworzenia funkcji „fabryk" pojawią się głównie przy dziedziczeniu, ponieważ w funkcji statycznej musielibyśmy podać konkretną klasę, tutaj natomiast przekazana klasa będzie zgodna z klasą, która ją wywołuje, nie ma więc potrzeby tworzenia nowej metody dla każdego potomka.

Zadanie 1 Utwórz klasę `Ulamek` o dwóch polach prywatnych: `mianownik`, `licznik`. Przeciąż cztery podstawowe operatory arytmetyczne: $*$ (`__mul__`), $+$ (`__add__`), $-$ (`__sub__`), / (`__truediv__`).

Aby ułatwić wyświetlanie dodaj poniższe przeciążenie funkcji specjalnej, pamiętając o zmianie nazw zmiennych:

```
def __str__(self):
     return str(self.__nominator) + "/" + str(self.__denominator)
```

Zadanie 2 Utwórz klasę `Osoba` o polach `pesel` (str), `imie` (str), `nazwisko` (str), `wzrost` (float — m), `waga` (float — kg), `rok_urodzenia` (int). Rok urodzenia powinien być obliczany na podstawie numeru pesel. Wszystkie pola powinny być ukryte/prywatne. Klasa powinna posiadać konstruktor 5 argumentowy z dwoma polami opcjonalnymi (pesel, imię, nazwisko, wzrost = 0.0, waga = 0.0).

Zadanie 3 Dodaj pole informujące o liczbie osób (statyczne, prywatne), pole to musi być zwiększane przy wywołaniu konstruktora i zmniejsza przy wywołaniu destruktora.

```
def __del__(self): # naglowek dsestruktora
del obiekt # usuniecie obiektu
```

Zadanie 4 Dodaj do klasy metody specjalne `__hash__` zwracająca numer pesel (ponieważ numer pesel powinien być unikatową wartością, może być zastosowany w tej roli) oraz `__str__` zwracające napis „PESEL, Imię Nazwisko", aby utworzyć nieoficjalną (`str(obiekt)`) reprezentacje klasy. Dodatkowo dodaj do klasy metodę `__repr__`, która zwraca oficjalną reprezentację klasy, czyli taką , która pozwala na odtworzenie obiektu przez interpreter (`eval(repr(obiekt))`).

Zadanie 5 Utwórz metody do: pobrania/zmiany wagi, pobrania/zmiany wzrostu, pobrania wieku (obliczanego jako różnica między aktualną datą, a rokiem urodzenia). Następnie dodaj statyczną metodą wyświetlającą liczbę instancji klasy. Na koniec dodaj metodę klasy tworzącą nowy obiekt na podstawie 5 argumentów, ale w odróżnieniu od konstruktora powinna ona oczekiwać wzrostu w cm i dokonywać odpowiedniej konwersji przy tworzeniu obiektu.