Python. OOP.Programowanie obiektowe. Klasy i metody.

Programowanie obiektowe to wzorzec programowania polegający na wykorzystaniu **klas i obiektów**.

Obiekt to po prostu zbiór danych (zmiennych) i metod (funkcji), które działają na tych danych. Podobnie, klasa jest planem dla tego obiektu. Klasa tworzy nowy typ, podczas gdy obiekty są instancjami (egzemplarzami) danej klasy. Każda klasa zawiera elementy dotyczące stanu (czyli dane, nazywane polami) i zachowanie (nazywane metodami(inna nazwa funkcje – w programowaniu strukturalnym).

Przykład: ”Papuga” to klasa. Każda papuga lubi latać, śpiewać to są metody klasy. Papuga ma swoje imię, wiek i wagę, to są pola klasy. Konkretna papuga, np. Ara, jest konkretnym egzemplarzem, czyli obiektem klasy, ponieważ możemy ją zindentyfikować po nazwie.

Obiekty posiadają pewne cechy i funkcje . Zadaniem obiektów w programie jest reprezentowanie wybranych, istotnych cech i funkcji rzeczywistego obiektu . Możemy stworzyć klasę reprezentującą dowolny obiekt. Obiekt ten możemy scharakteryzować później dowolnymi zmiennymi lub funkcjami.

Właściwości obiektów :

- Zachowanie obiektu – co można zrobić dzięki temu obiektowi i jakie metody (funkcje) można dla niego wywoływać?
- Stan obiektu – jak obiekt reaguje na działanie tych metod?
- Tożsamość obiektu – w jaki sposób można odróżnić ten obiekt od innych, posiadając to samo zachowanie i stan?

Deklaracja klasy odbywa się za pomocą dyrektywy class, nazwy klasy oraz ciała klasy Użytkownik deklaruje własne klasy, by na ich podstawie tworzyć konkretne obiekty dopasowane do własnych potrzeb.

Przykład : Tworzymy klasę

```
class Parrot:
  pass
```

Przykład : Tworzymy pierwszy program

```
class Parrot:

  species = "ptak"

  def __init__(self, name, age):
    self.name = name
    self.age = age

blu = Parrot("Blu", 10)
ara = Parrot("Ara", 15)
```

```
print("Blu jest {}".format(blu.__class__.species))
print("Ara jest {}".format(ara.__class__.species))

print("{} ma {} lat".format( blu.name, blu.age))
print("{} ma {} lat".format( ara.name, ara.age))
```

W powyższym programie stworzyliśmy klasę o nazwie Parrot. Następnie definiujemy atrybuty. Atrybuty są charakterystyczne dla obiektu.

Te atrybuty są zdefiniowane wewnątrz __init__metody klasy. Jest to metoda inicjowania, która jest uruchamiana po raz pierwszy zaraz po utworzeniu obiektu. Następnie tworzymy dwa obiekty – dwie papugi o imieniu Blu oraz Ara.

Możemy uzyskać dostęp do atrybutu klasy za pomocą __class__.species. Atrybuty klasy są takie same dla wszystkich instancji klasy. Podobnie uzyskujemy dostęp do atrybutów instancji za pomocą blu.name i blu.age. Jednak atrybuty instancji są różne dla każdej instancji klasy.

Self

Pierwszy argument każdej metody - "self" - przekazuje do każdej metody (funkcji zdefiniowanej wewnątrz klasy) instancję klasy na jakiej mają operować. Metody poprzedzone __ to metody specjalne, np __init__ jest wykonywana przy utworzeniu instancji klasy. Instancje powstają w wyniku wywołania obiektu klasy jako funkcji. Po utworzeniu nowej instancji klasy jej atrybuty i metody dostępne są za pomocą operatora kropki.

Konstruktor

Konstruktor jest specjalną metodą, której zadaniem jest, utworzenie obiektu w pamięci operacyjnej. Dla każdej klasy, konstruktor tworzy się w ten sam sposób, używając nazwy „_ _init _ _” z dwoma podkreślnikami po obu stronach (tzw. podłogą). Konstruktor może przyjmować parametry lub nie, ale musi posiadać przynajmniej jeden parametr „self”, który jest parametrem zarówno dla konstruktora, jak i dla każdej metody klasy.

Destruktor

Destruktor jest kolejną metodą specjalną, której zadaniem jest usuwanie z pamięci operacyjnej wszystkiego co utworzyliśmy, tworząc obiekt klasy przy pomocy konstruktora. Czyli czyścimy nim pamięć operacyjną, ze zbędnych zmiennych (danych), jakie zostają po zakończeniu działania programu. Destuktor dla każdej klasy tworzymy w ten sam sposób, używając nazwy „_ _del_ _” z dwoma podkreślnikami po obu stronach.

Metody

Metody to funkcje zdefiniowane w treści klasy. Służą do definiowania zachowań obiektu.
Przykład:
class Parrot:

#tu to co było wcześniej

```
   def sing(self, song):
      return "{} spiewa {}".format(self.name, song)

   def dance(self):
      return "{} tanczy".format(self.name)

blu = Parrot("Blu", 10)

print(blu.sing("'Happy'"))
print(blu.dance())
print(ara.dance())
```

W powyższym programie metody możemy definiować na dwa sposoby tj. sing() i dance() tak jak pokazane jest w przykładzie. Nazywane są one metodami instancji, ponieważ są wywoływane w obiekcie instancji, tj. blu.

Zapamiętać! : Klasa to typ obiektu, a nie sam obiekt! Funkcje i zmienne w klasie nazywamy ogólnie składnikami klasy.

Lista zadań:

1. Napisz klasę samochód w której zdefiniujesz 5 obiektów dotyczących samochodów np. marka, model, przebieg. Dodaj 5 metod, które będą wywoływane na obiektach z przekazywanymi parametrami np. jedzprosto(); Jeden samochód powinien mieć kolor "czerwony", rodzaj "kabriolet", wartość 60000 i nazwę "Ferrari". Zaproponuj pozostałe samochody.
2. Napisz klasę książka w której zdefiniujesz 5 obiektów dotyczących książek np. autor, tytuł Dodaj 5 metod, które będą wywoływane na obiektach z przekazywanymi parametrami np. czytaj(); Dodaj książki do list. Spróbuj utworzyć metody odpowiedzialne za sortowanie.
3. Napisz klasę smartfon w której zdefiniujesz kilka obiektów dotyczących smartfonów i dodaj ciekawe metody. Spróbuj eksperymentować z metodami oraz
4. Napisz program, w którym spróbujesz zamodelować otaczającą Cię rzeczywistość. Napisz klasę student, w której zdefiniujesz obiekty dotyczące studenta. Jakie to będą obiekty? Dodaj metody,które będą wywoływane na obiektach z przekazywanymi parametrami. Jakimi parametrami?