

# Hakersi:

*programowanie obiektowe  
w pythonie*



# Czym jest programowanie obiektowe

Programowanie obiektowe (OOP) to sposób projektowania programów, który pozwala na łatwiejsze tworzenie skomplikowanych systemów. Jest to technika programowania, która umożliwia programistom myślenie o programie w kategoriach obiektów, a nie tylko w kategoriach danych i funkcji.

Każdy obiekt jest instancją klasy, która jest modelem dla danego typu obiektu. Klasy mają atrybuty, które reprezentują dane, jakie posiada dany obiekt, oraz metody, które reprezentują czynności, jakie można wykonać na tym obiekcie.

# Klasy

Można sobie wyobrazić klasę jako zestaw cech, jakie posiada określona rzecz - na przykład rower może mieć kolor, markę, rozmiar i stan, a samochód może mieć markę, model, rok produkcji i prędkość maksymalną. W OOP tworzymy modele, które opisują, jakie cechy i zachowania posiadają nasze obiekty.

# Klasa: Rower

Atrybuty (attributes):

- kolor
- marka
- rozmiar
- Stan

Metody (methods):

- Jedzie do przodu
- Hamuje

```
python Copy code

class Rower:
    def __init__(self, kolor, marka, rozmiar, stan):
        self.kolor = kolor
        self.marka = marka
        self.rozmiar = rozmiar
        self.stan = stan


    def jedz_do_przodu(self):
        print("Jadę do przodu!")

    def hamuj(self):
        print("Hamuję!")
```

## Elementy w pythonie:

1. Słowo kluczowe: class
2. Konstruktor: \_\_init\_\_
3. Odwołanie do samego siebie: self

python

 Copy code

```
class Rower:
    def __init__(self, kolor, marka, rozmiar, stan):
        self.kolor = kolor
        self.marka = marka
        self.rozmiar = rozmiar
        self.stan = stan

    def jedz_do_przodu(self):
        print("Jadę do przodu!")

    def hamuj(self):
        print("Hamuję!")
```

# Obiekt - “instancja klasy”

Dzięki temu, że opisujemy modele obiektów w postaci klas, a obiekty tworzymy na podstawie tych klas, możemy łatwo tworzyć wiele instancji obiektów o tych samych cechach i zachowaniach, jednocześnie zapewniając ich bezpieczeństwo i abstrakcję.

# Obiekt: Rower Joanny

## Atrybuty (attributes):

- kolor: czerwony
- marka: quechua
- rozmiar: mały
- stan: dobry

## Metody (methods):

- Jedzie do przodu
- Hamuje

```
1 class Rower:
2     def __init__(self, kolor, marka, rozmiar, stan):
3         self.kolor = kolor
4         self.marka = marka
5         self.rozmiar = rozmiar
6         self.stan = stan
7
8     def jedz_do_przodu(self):
9         print("Jadę do przodu!")
10
11    def hamuj(self):
12        print("Hamuję!")
13
14
15 rower_joanny = Rower(kolor='czerwony', marka='quechua', rozmiar='mały', stan='dobry')
16
```

# Zadanie

Korzystając z klasy po prawej, stwórz obiekt typu "Bicycle" i zmień jego prędkość do 10000 mph (10km / k).

Stwórz drugi obiekt z innymi atrybutami i spróbuj sprawdzić czy są takie same.

```
class Bicycle:
    def __init__(self, brand, model, color):
        self.brand = brand
        self.model = model
        self.color = color
        self.current_speed = 0

    def pedal(self, speed_increase):
        self.current_speed += speed_increase

    def brake(self, speed_decrease):
        self.current_speed = max(0, self.current_speed - speed_decrease)

    def describe(self):
        print(f"This is a {self.color} {self.brand} {self.model} bicycle going {self.current_speed} mph." )
```



# Specjalne metody

Metoda `__str__` jest specjalną metodą w Pythonie, która zwraca reprezentację obiektu w postaci łańcucha znaków. W tym przypadku zwraca łańcuch znaków, który reprezentuje kolor, markę i model roweru.

Metoda `__eq__` to kolejna specjalna metoda w Pythonie, która definiuje, w jaki sposób dwa obiekty tej samej klasy mogą być porównywane pod względem równości. W tym przypadku porównuje kolor, markę i model dwóch rowerów, aby określić, czy są one takie same.

```
class Bicycle:

    def __init__(self, brand, model, color):

        self.brand = brand

        self.model = model

        self.color = color

        self.current_speed = 0

    def pedal(self, speed_increase):

        self.current_speed += speed_increase

    def brake(self, speed_decrease):

        self.current_speed = max(0, self.current_speed - speed_decrease)

    def describe(self):

        print(f"This is a {self.color} {self.brand} {self.model} bicycle going {self.current_speed} mph." )

    def __str__(self):

        return f"{self.color} {self.brand} {self.model}"

    def __eq__(self, other):

        if isinstance(other, Bicycle):

            return (self.color == other.color and

                    self.brand == other.brand and

                    self.model == other.model)

        return False
```

# Zadanie

1. Zamień poprzednia klasę `Bicycle` na nowa i wypróbuj jak działają specjalne metody.
2. Dodaj nowy atrybut: height i specjalna metodę która sprawdza czy jeden rower jest większy od drugiego.

```
class Bicycle:

    def __init__(self, brand, model, color):

        self.brand = brand

        self.model = model

        self.color = color

        self.current_speed = 0

    def pedal(self, speed_increase):

        self.current_speed += speed_increase

    def brake(self, speed_decrease):

        self.current_speed = max(0, self.current_speed - speed_decrease)

    def describe(self):

        print(f"This is a {self.color} {self.brand} {self.model} bicycle going {self.current_speed} mph." )

    def __str__(self):

        return f"{self.color} {self.brand} {self.model}"

    def __eq__(self, other):

        if isinstance(other, Bike):

            return (self.color == other.color and

                    self.brand == other.brand and

                    self.model == other.model)

        return False
```