Python Базовий

Ознайомлення з ООП: Успадкування

Introduction

Вікторія Бойчук
Python Developer, тренер CBS

Вікторія Бойчук

План курсу

1. Ознайомлення з ООП. Успадкування
2. Інкапсуляція
3. Поліморфізм
4. Абстракція
5. Робота з рекурсією
6. Структури даних
7. Модулі Python
8. Читання та запис файлів
9. PEP8 стандарти оформлення коду

Тема урока

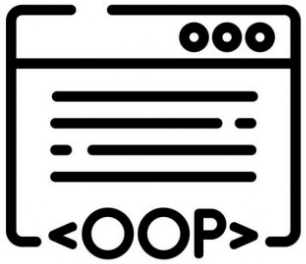
Ознайомлення з ООП: Успадкування

План уроку

1. Про що курс
2. Що таке ООП
3. Що таке наслідування
4. Приклади використання ООП та успадкування на практиці

Про що курс

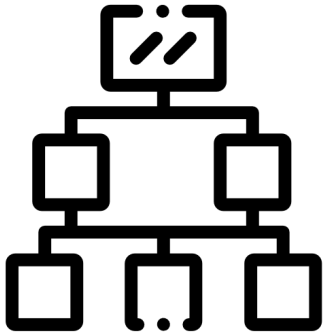
- ООП: успадкування, інкапсуляція, поліморфізм та абстракція
- Модулі Python (math, collections, itertools і т. п.)
- Правила "хорошого тону" у Python коді (PEP8)
- Робота з файлами (txt, json, yml, xml итд.)



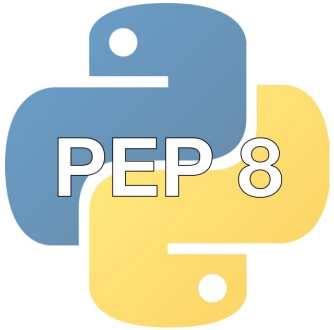
OOP

Recursion

Files



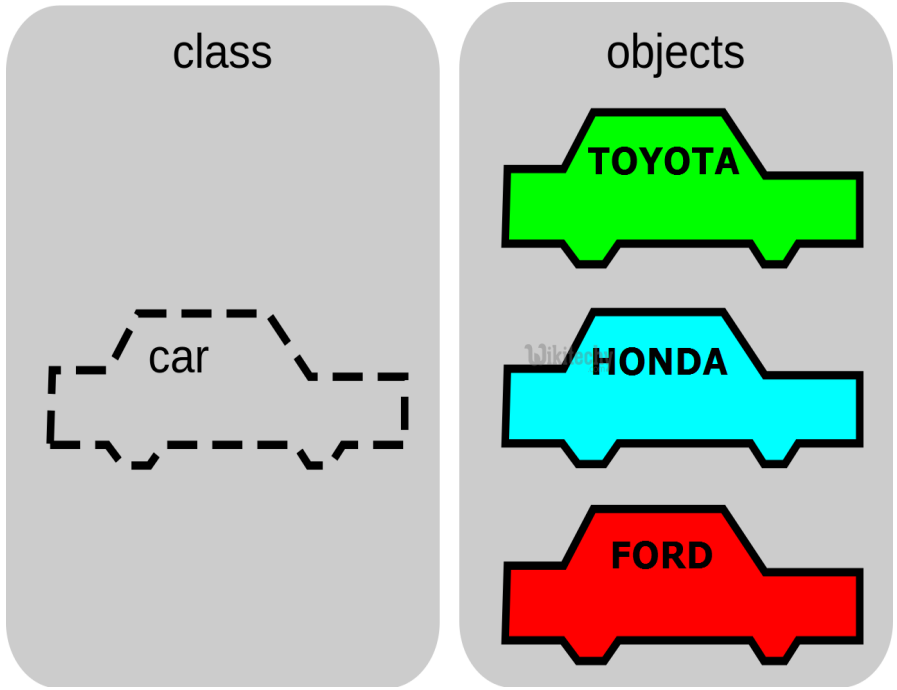
Modules



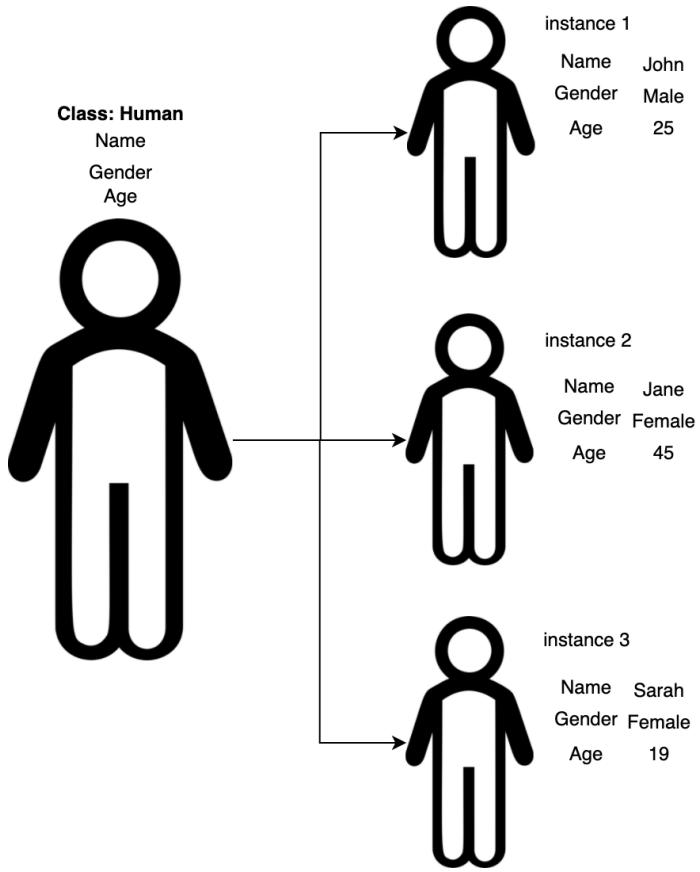
PEP8

Поняття ООП

Об'єктно-орієнтоване програмування (ООП) — методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів, кожен із яких є екземпляром певного класу, а класи утворюють ієрархію спадкування.

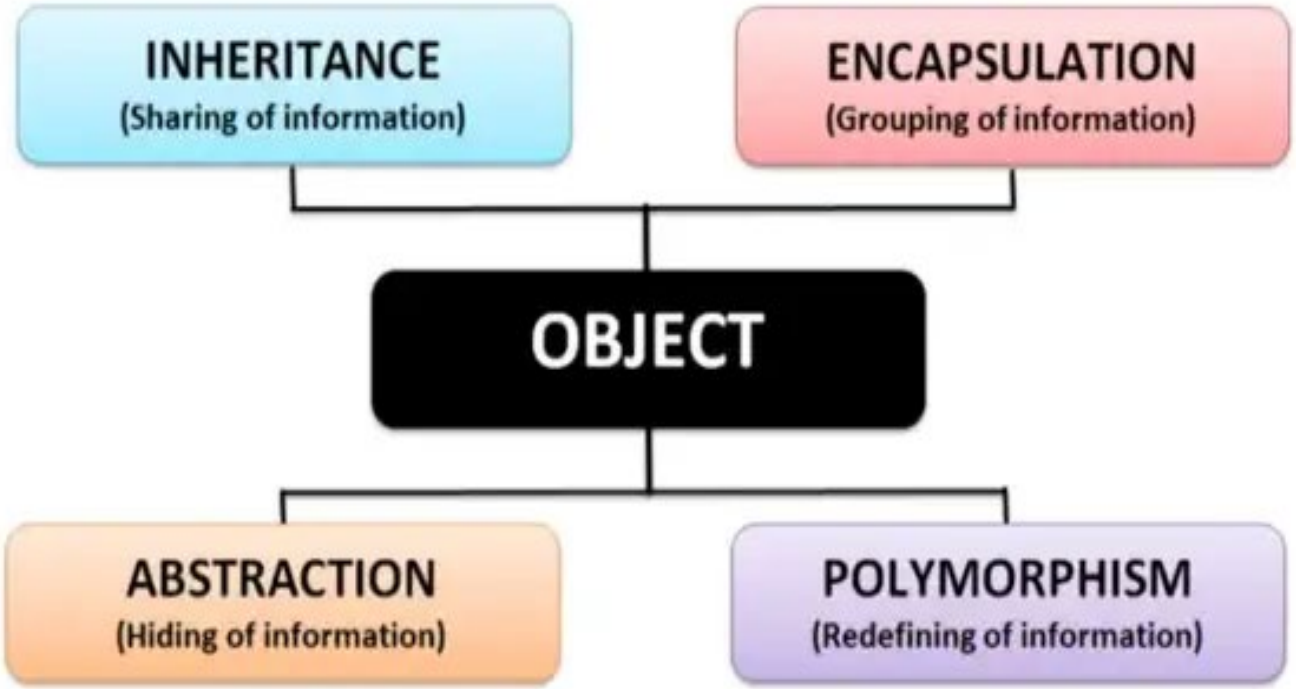


Клас та об'єкт класу



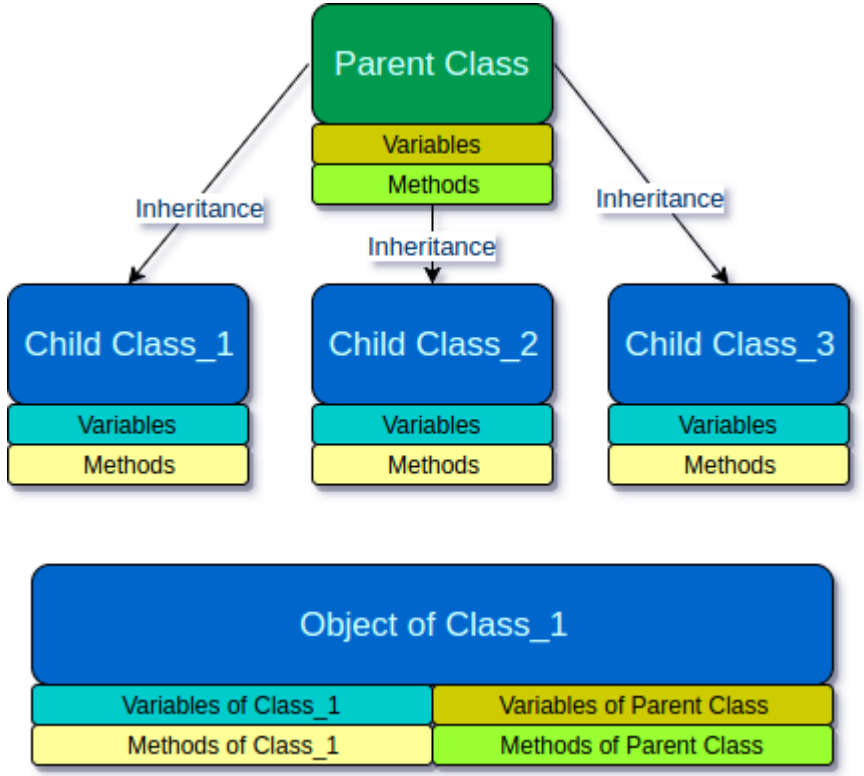

```python
class Human:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def get_name(self):
        return self.name

    def get_gender(self):
        return self.gender

    def get_age(self):
        return self.age


person = Human(name='John',
               age=34,
               gender='male')

```

Концепції ООП



Успадкування

Успадкування (англ. inheritance) — концепція об'єктно-орієнтованого програмування, згідно з якою абстрактний тип даних може успадковувати дані та функціональність деякого існуючого типу, сприяючи повторному використанню компонентів програмного забезпечення.



Види наслідування

Просте

```python
class Parent:
    def __init__(self, name):
        self.name = name

    def say_hello(self):
        print("Hello, I am {name}".format(name=self.name))


parent = Parent(name="John")
parent.say_hello()

class Child(Parent):
    def __init__(self, name, age):
        super().__init__(name)
        self.age = age

    def say_hello(self):
        print(
            "Hello, my name is {name} and I am {age} years old".format(
                name=self.name, age=self.age
            )
        )

child = Child(name="Mark", age=25)
child.say_hello()

```

Множинне

```python
class A:
    def __init__(self):
        self.a = 10


class B:
    def __init__(self):
        self.b = 25


class C(A, B):
    def __init__(self):
        A.__init__(self)
        B.__init__(self)


c = C()
print("C class has a={a_vaue} and b={b_value}".format(a_vaue=c.a, b_value=c.b))

```

Задачі

1. Написати клас автомобіля з атрибутами марки, кольору та об'єму двигуна та методами: їхати вперед та їхати назад.

2. Написати клас автомобіля, успадкованого від першого класу у пункті 1. Додати методи повороту ліворуч та праворуч.

3. Написати клас літака, що має метод злітати та атрибут модель літака.

4. Написати клас, успадкований від машини (2 пункт) та від літака (3 пункт). Подивитися що буде.

P.S. Усі методи - це просто команда друку, наприклад print(“Drive forward”) і т.д.

Рішення

```python
class Car:
    def __init__(self, brand, color, vol):
        self.brand = brand
        self.color = color
        self.vol = vol

    def drive_forward(self):
        print('Drive forward')

    def drive_backward(self):
        print('Drive backward')


class Car2(Car):
    def __init__(self, brand, color, vol):
        super().__init__(brand, color, vol)

    def turn_right(self):
        print("Turn right")

    def turn_left(self):
        print("Turn left")


class Airplane:
    def __init__(self, model):
        self.model = model

    def fly(self):
        print("Start the flight")
```

```python
class FlyingCar(Car2, Airplane):
    def __init__(self,  brand, color, vol, model):
        Car2.__init__(self, brand, color, vol)
        Airplane.__init__(self, model)

flying_car = FlyingCar(
    brand='Tesla',
    color='black',
    vol=4.5,
    model='F'
)
flying_car.fly()
```

```
Start the flight

Process finished with exit code 0
```

Інформаційний відеосервіс для розробників програмного забезпечення

TestProvider Pearson CyberBionic systematics Microsoft Partner Windows Azure EBA

Перевірка знань

TestProvider.com

TestProvider – це online сервіс перевірки знань з інформаційних технологій. За його допомогою Ви можете оцінити Ваш рівень та виявити слабкі місця. Він буде корисним як у процесі вивчення технології, так і для загальної оцінки знань IT спеціаліста.

Перевірте як Ви засвоїли даний матеріал на TestProvider.com

Успішне проходження фінального тестування дозволить Вам отримати відповідний Сертифікат.

Дякую за увагу! До нових зустрічей!

Вікторія Бойчук
Python Developer, тренер CBS