Python Класи та Об'єкти

Зміст

Створення класу

Клас у Python створюється за допомогою ключового слова `class`:

```
class MyClass:
    x = 5
```

Створення об'єкта

Об'єкт створюється на основі класу:

```
p1 = MyClass()
print(p1.x)
```

Функція init

Функція `__init__()` викликається автоматично при створенні об'єкта:

```
class Person:
    def __init__(self, name, age):
        self.name = name
```

```
        self.age = age

p1 = Person("John", 36)
print(p1.name)
print(p1.age)
```

Функція str

Функція `__str__()` визначає, що буде повернено при перетворенні об'єкта на рядок:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"{self.name}({self.age})"

p1 = Person("John", 36)
print(p1)
```

Методи об'єктів

Методи — це функції, які належать об'єкту:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def myfunc(self):
        print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Параметр `self`

`self` використовується для доступу до змінних класу:

```
class Person:
    def __init__(mysillyobject, name, age):
        mysillyobject.name = name
        mysillyobject.age = age
```

```
    def myfunc(abc):
        print("Hello my name is " + abc.name)

p1 = Person("John", 36)
p1.myfunc()
```

Зміна властивостей об'єкта

Властивості об'єкта можна змінювати:

```
p1.age = 40
```

Видалення властивостей та об'єктів

Властивості та об'єкти можна видаляти за допомогою ключового слова `del`:

```
del p1.age
del p1
```

Оператор `pass`

Клас не може бути порожнім, але можна використовувати `pass`:

```
class Person:
    pass
```

Наслідування в Python

Наслідування дозволяє створювати класи, які успадковують властивості та методи іншого класу:

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def printname(self):
        print(self.firstname, self.lastname)

class Student(Person):
```

```
    def __init__(self, fname, lname, year):
        super().__init__(fname, lname)
        self.graduationyear = year

    def welcome(self):
        print("Welcome", self.firstname, self.lastname, "to the class of",
self.graduationyear)

x = Student("Mike", "Olsen", 2019)
x.welcome()
```

Ітератори в Python

Ітератори — це об'єкти, які реалізують методи `__iter__()` та `__next__()`:

```
class MyNumbers:
    def __iter__(self):
        self.a = 1
        return self

    def __next__(self):
        if self.a <= 20:
            x = self.a
            self.a += 1
            return x
        else:
            raise StopIteration

myclass = MyNumbers()
myiter = iter(myclass)

for x in myiter:
    print(x)
```

Поліморфізм в Python

Поліморфізм дозволяє використовувати методи з однаковими іменами для різних класів:

```
class Vehicle:
    def move(self):
        print("Move!")

class Car(Vehicle):
    def move(self):
        print("Drive!")

class Boat(Vehicle):
```

```
    def move(self):
        print("Sail!")

class Plane(Vehicle):
    def move(self):
        print("Fly!")

for vehicle in (Car(), Boat(), Plane()):
    vehicle.move()
```