Олексій Васильєв

Мова програмування Python

Київ 2020

Лекція 10. Класі і об'єкти

- Класи
- Об'єкти
- Методи і поля
- Конструктори і деструктори
- Копіювання об'єктів
- Декоратори класів

Класи і об'єкти

- **В програмі можна використовувати об'єкти, котрі об'єднують в собі данні різного типу і функції, призначені для роботи з цими даними**
- **Об'єкти створюються на основі класів. Клас являє собою шаблон, відповідно до якого реалізуються об'єкти**

```
# Описання класу:
class MyClass:
    pass
# Створення об'єктів на основі класу:
A=MyClass()
B=MyClass()
# Об'єкти:
print("Об'єкт A:",A)
print("Об'єкт B:",B)
# Тип об'єктів:
print("Клас об'єкта A:",type(A).__name__)
print("Клас об'єкта B:",type(B).__name__)
# Порівняння об'єктів:
print("A==B:",A==B)
```

Шаблон описання класу:

```
class назва:
    # Описання класу
```

Створення об'єкта:

```
змінна=Клас()
```

Якщо тіло пусте: ключ. сл. pass

Програма (Objs.py)

```
Об'єкт A: <__main__.MyClass object at 0x02174FB0>
Об'єкт B: <__main__.MyClass object at 0x021AA890>
Клас об'єкта A: MyClass
Клас об'єкта B: MyClass
A==B: False
```

Поля і методи

```
# Описання класу:
class MyClass:
    # Метод для присвоювання
    # значення полю:
    def set(self,n):
        self.number=n
    # Метод для відображення
    # значення поля:
    def show(self):
        print("Поле number =",self.number)
# Створення об'єктів:
A=MyClass()
B=MyClass()
# Присвоєння значень полям об'єктів:
A.set(100)
B.set(200)
# Перевірка значень полів об'єктів:
A.show()
B.show()
# Присвоєння значень полям об'єктів:
A.number=123
B.number=321
# Перевірка значень полів об'єктів:
A.show()
B.show()
```

- Об'єкт може містити поля і методи
- Метод описується як звичайна функція, але в тілі класу
- Перший аргумент `self` - посилання на об'єкт, з якого викликається метод
- Поле - змінна, пов'язана з об'єктом
- Поле у об'єкта з'являється, коли ми полю присвоюємо значення

Програма (Methods.py)

```
Поле number = 100
Поле number = 200
Поле number = 123
Поле number = 321
```

Поля об'єкта

```
# Описання класу:
class MyClass:
    def set(self,n):
        self.number=n
    def show(self):
        print("Поле number =",self.number)
# Створення об'єкта:
obj=MyClass()
# Перевірка наявності поля:
print("Наявність поля number:",hasattr(obj,"number"))
try:
    # Перевірка значення поля:
    obj.show()
except AttributeError:
    print("Поля number у об'єкта немає!")
# Присвоєння значення полю:
obj.number=123
# Перевірка наявності поля:
print("Наявність поля number:",hasattr(obj,"number"))
# Перевірка значення поля:
obj.show()
# Нове значення поля:
obj.set(321)
# Перевірка значення поля:
obj.show()
```

Програма (Fields.py)

```
Наявність поля number: False
Поля number у об'єкта немає!
Наявність поля number: True
Поле number = 123
Поле number = 321
```

Конструктори і деструктори

```
# Описання класу:
class MyClass:
    # Конструктор:
    def __init__(self,n="Білий"):
        self.name=n
        print("Створено об'єкт:",self.name)
    # Деструктор:
    def __del__(self):
        print("Видалено об'єкт:",self.name)
# Функція:
def create(n):
    obj=MyClass(n)
# Створення об'єктів:
A=MyClass()
B=MyClass("Червоний")
C=MyClass("Синій")
# Виклик функції:
create("Жовтий")
# Полю присвоюється нове значення:
C.name="Зелений"
# Видалення об'єктів:
del A
del B
del C
```

- **Конструктор** (метод `__init__()`) **автоматично викликається при створенні об'єкта**

- **Деструктор** (метод `__del__()`) **автоматично викликається при видаленні об'єкта з пам'яті**

Програма (ConstrDestr.py)

```
Створено об'єкт: Білий
Створено об'єкт: Червоний
Створено об'єкт: Синій
Створено об'єкт: Жовтий
Видалено об'єкт: Жовтий
Видалено об'єкт: Білий
Видалено об'єкт: Червоний
Видалено об'єкт: Зелений
```

Реалізація класу

Програма (ClassObj.py)

```
# Описання класів:
class Alpha:
    pass
class Bravo:
    pass
# Назва класу як значення:
MyClass=Alpha
# Створення об'єкта:
A=MyClass()
# Назва класу як значення:
MyClass=Bravo
# Створення об'єкта:
B=MyClass()
# Присвоєння класів:
Alpha=Bravo
# Створення об'єкта:
C=Alpha()
# Посилання на об'єкт
# реалізації класу:
MyClass=A.__class__
# Створення об'єкта:
D=MyClass()
# Перевірка типу об'єктів:
print("Об'єкт A:",type(A).__name__)
print("Об'єкт B:",type(B).__name__)
print("Об'єкт C:",type(C).__name__)
print("Об'єкт D:",type(D).__name__)
# Зміна назв класів:
MyClass.__name__="First"
Bravo.__name__="Second"
# Перевірка типу об'єктів:
print("Об'єкт A:",type(A).__name__)
print("Об'єкт B:",type(B).__name__)
print("Об'єкт C:",type(C).__name__)
print("Об'єкт D:",type(D).__name__)
```

```
Об'єкт A: Alpha
Об'єкт B: Bravo
Об'єкт C: Bravo
Об'єкт D: Alpha
Об'єкт A: First
Об'єкт B: Second
Об'єкт C: Second
Об'єкт D: First
```

Клас реалізується за допомогою об'єкта (об'єкт реалізації класу)

Поля і методи класу

```
# Описання класу:
class MyClass:
    # Поле класу:
    color="Червоний"
    # Методи класу:
    def set(txt):
        MyClass.color=txt
    def show():
        print(MyClass.color)
# Виклик методів класу:
MyClass.show()
MyClass.set("Зелений")
# Відображення значення поля класу:
print(MyClass.color)
# Нове значення для поля класу:
MyClass.color="Синій"
# Виклик методу класу:
MyClass.show()
# Створення об'єктів класу:
A=MyClass()
B=MyClass()
# Перевірка значення поля:
print("Клас:",MyClass.color)
print("Об'єкт A:",A.color)
print("Об'єкт B:",B.color)
```

```
Червоний
Зелений
Синій
Клас: Синій
Об'єкт A: Синій
Об'єкт B: Синій
Клас: Синій
Об'єкт A: Білий
Об'єкт B: Синій
Клас: Жовтий
Об'єкт A: Білий
Об'єкт B: Жовтий
Клас: Жовтий
Об'єкт A: Жовтий
Об'єкт B: Жовтий
```

```
# Присвоєння значення полю:
A.color="Білий"
# Перевірка значення поля:
print("Клас:",MyClass.color)
print("Об'єкт A:",A.color)
print("Об'єкт B:",B.color)
# Присвоєння значення полю:
MyClass.color="Жовтий"
# Перевірка значення поля:
print("Клас:",MyClass.color)
print("Об'єкт A:",A.color)
print("Об'єкт B:",B.color)
# Видалення поля з об'єкта A:
del A.color
# Перевірка значення поля:
print("Клас:",MyClass.color)
print("Об'єкт A:",A.color)
print("Об'єкт B:",B.color)
```

Програма (ClassMethods.py)

Клас і об'єкт

Програма (ClassAndObject.py)

```
# Описання класу:
class MyClass:
    # Конструктор:
    def __init__(self):
        self.value=123
        print("Створюється об'єкт:",self.value)
    # Деструктор:
    def __del__(self):
        print("Видаляється об'єкт:",self.value)
    # Присвоєння значення полю:
    def set(self,n):
        self.value=n
    # Відображення значення поля:
    def show(self):
        print("Поле об'єкта:",self.value)
# Створення об'єкта:
obj=MyClass()
# Виклик методів з об'єкта:
obj.show()
obj.set(100)
```

```
Створюється об'єкт: 123
Поле об'єкта: 123
Поле об'єкта: 100
Поле об'єкта: 200
Створюється об'єкт: 123
Видаляється об'єкт: 123
Поле об'єкта: 123
Поле об'єкта: 321
Створюється об'єкт: 123
Видаляється об'єкт: 123
Поле об'єкта: 123
```

```
# Виклик методів з класу:
MyClass.show(obj)
MyClass.set(obj,200)
# Перевірка значення поля:
obj.show()
# Явний вклик конструктора:
MyClass.__init__(obj)
# Явний виклик деструктора:
MyClass.__del__(obj)
# Перевірка значення поля:
obj.show()
# Зміна значення поля:
obj.value=321
obj.show()
# Явний виклик конструктора:
obj.__init__()
# Явний виклик деструктора:
obj.__del__()
# Перевірка значення поля:
obj.show()
```

Атрибути

- Об'єкты і класи технічно зберігаються у вигляді словників

- Доступ до словника, через який реалізується об'єкт/клас, можна отримати за допомогою поля `__dict__`

- Зв'язок між об'єктом і класом, на основі якого об'єкт створено, встановлюється на рівні послідовності пошуку атрибутів: спочатку запитуваний атрибут шукається в об'єкті, а якщо він там не знайдений — то в класі (об'єкті, на основі якого реалізований клас)

Атрибути

Програма (Atributes.py)

```
# Перший клас:
class Alpha:
    pass
# Другий клас:
class Bravo:
    name="Bravo"
    def display():
        print("Поле name:",Bravo.name)
    def show(self):
        print("Поле value:",self.value)
    def __init__(self):
        self.value=123
# Створення об'єктів:
A=Alpha()
B=Bravo()
# Атрибути першого класу:
print("Класс Alpha")
n=1
for s in Alpha.__dict__:
    print("["+str(n)+"] "+s+":",Alpha.__dict__[s])
    n+=1
# Атрибути другого класу:
print("Класс Bravo")
n=1
for s in Bravo.__dict__:
    print("["+str(n)+"] "+s+":",Bravo.__dict__[s])
    n+=1
# Атрибути об'єктів:
print("Об'єкт A:",A.__dict__)
print("Об'єкт B:",B.__dict__)
```

```
# Виклик методу класу:
Bravo.display()
# Створення атрибуту класу:
Alpha.display=Bravo.display
# Видалення атрибуту класу:
del Bravo.display
# Виклик методу з об'єкта:
B.show()
# Створення атрибуту класу:
A.color="Червоний"
# Створення атрибуту об'єкта:
B.show=lambda: print("Об'єкт B:",B.value)
# Атрибути першого класу:
print("Клас Alpha")
n=1
for s in Alpha.__dict__:
    print("["+str(n)+"] "+s+":",Alpha.__dict__[s])
    n+=1
# Атрибути другого класу:
print("Клас Bravo")
n=1
for s in Bravo.__dict__:
    print("["+str(n)+"] "+s+":",Bravo.__dict__[s])
    n+=1
# Атрибути об'єктів:
print("Об'єкт A:",A.__dict__)
print("Об'єкт B:",B.__dict__)
# Виклик методів:
Alpha.display()
Bravo.show(B)
B.show()
```

Атрибути

Програма (Atributes.py)
Результат виконання

```
Клас Alpha
[1] __module__: __main__
[2] __dict__: <attribute '__dict__' of 'Alpha' objects>
[3] __weakref__: <attribute '__weakref__' of 'Alpha' objects>
[4] __doc__: None
Клас Bravo
[1] __module__: __main__
[2] name: Bravo
[3] display: <function Bravo.display at 0x02646078>
[4] show: <function Bravo.show at 0x026460C0>
[5] __init__: <function Bravo.__init__ at 0x02646108>
[6] __dict__: <attribute '__dict__' of 'Bravo' objects>
[7] __weakref__: <attribute '__weakref__' of 'Bravo' objects>
[8] __doc__: None
Об'єкт A: {}
Об'єкт B: {'value': 123}
Поле name: Bravo
Поле value: 123
Клас Alpha
[1] __module__: __main__
[2] __dict__: <attribute '__dict__' of 'Alpha' objects>
[3] __weakref__: <attribute '__weakref__' of 'Alpha' objects>
[4] __doc__: None
[5] display: <function Bravo.display at 0x02646078>
Клас Bravo
[1] __module__: __main__
[2] name: Bravo
[3] show: <function Bravo.show at 0x026460C0>
[4] __init__: <function Bravo.__init__ at 0x02646108>
[5] __dict__: <attribute '__dict__' of 'Bravo' objects>
[6] __weakref__: <attribute '__weakref__' of 'Bravo' objects>
[7] __doc__: None
Об'єкт A: {'color': 'Червоний'}
Об'єкт B: {'value': 123, 'show': <function <lambda> at 0x02646030>}
Поле name: Bravo
Поле value: 123
Об'єкт B: 123
```

Копіювання

Програма (ObjCopy.py)

```
# Імпорт функцій:
from copy import *
# Описання класу:
class MyClass:
    pass
# Створення об'єкта:
A=MyClass()
# Поля об'єкта:
A.value=100
A.nums=[1,2,3]
# Присвоєння посилання на об'єкт:
B=A
# Копія об'єкта:
C=copy(A)
# Повна копія об'єкта:
D=deepcopy(A)
print("Створені об'єкти")
# Поля об'єктов:
print("A:",A.value,"i",A.nums)
print("B:",B.value,"i",B.nums)
print("C:",C.value,"i",C.nums)
print("D:",D.value,"i",D.nums)
print("A.value=200 i A.nums[1]=0")
# Нові значення для полів:
A.value=200
A.nums[1]=0
# Поля об'єктів:
print("A:",A.value,"i",A.nums)
print("B:",B.value,"i",B.nums)
print("C:",C.value,"i",C.nums)
print("D:",D.value,"i",D.nums)
print("Видаляється A")
# Видалення змінної:
del A
print("B.value=300 i B.nums[2]=4")
# Нові значення для полів:
B.value=300
B.nums[2]=4
# Поля об'єктів:
print("B:",B.value,"i",B.nums)
print("C:",C.value,"i",C.nums)
print("D:",D.value,"i",D.nums)
```

Копіювання

Програма (ObjCopy.py)
Результат виконання

```
Створені об'єкти
A: 100 і [1, 2, 3]
B: 100 і [1, 2, 3]
C: 100 і [1, 2, 3]
D: 100 і [1, 2, 3]
A.value=200 і A.nums[1]=0
A: 200 і [1, 0, 3]
B: 200 і [1, 0, 3]
C: 100 і [1, 0, 3]
D: 100 і [1, 2, 3]
Видаляється A
B.value=300 і B.nums[2]=4
B: 300 і [1, 0, 4]
C: 100 і [1, 0, 4]
D: 100 і [1, 2, 3]
```

Документування

Програма (Document.py)

```
# Перший клас:
class Alpha:
    "Це клас Alpha"
    pass
# Другий клас:
class Bravo:
    "Це клас Bravo"
    pass
# Інформація про класи:
print(Alpha.__doc__)
print(Bravo.__doc__)
# Об'єкти класів:
A=Alpha()
B=Bravo()
# Зміна рядка документування:
Alpha.__doc__="Перший клас"
B.__class__.__doc__="Другий клас"
# Інформація про класи:
print(A.__class__.__doc__)
print(B.__doc__)
```

- **Якщо в описанні класу відразу після рядка з ключовим словом `class` і назвою класу вказати текстовий літерал, то він буде визначати рядок документування класу**
- **Рядок документування доступний через спеціальну властивість `__doc__`**

```
Це клас Alpha
Це клас Bravo
Перший клас
Другий клас
```

Клас як результат

```
# Функція аргументом отримує посилання на клас
# і результатом повертає посилання на клас:
def F(Alpha):
    # Внутрішній клас:
    class Bravo:
        value=Alpha()
    Bravo.__name__="My"+Alpha.__name__
    return Bravo
# Описання класу:
class Charlie:
    # Конструктор:
    def __init__(self):
        self.number=123
    # Метод для відображення значення поля:
    def show(self):
        print("Поле number:",self.number)
# Створення об'єкта на основі класу,
# отриманого при виклику функції:
obj=F(Charlie)()
# Перевірка результату:
obj.value.show()
print("Клас об'єкта obj:",obj.__class__.__name__)
print("Клас поля value:",obj.value.__class__.__name__)
```

Програма (ClassResult.py)

```
Поле number: 123
Клас об'єкта obj: MyCharlie
Клас поля value: Charlie
```

Декоратор класу

Програма (Decorator.py)

```
# Функція для декоратора:
def F(Alpha):
    class Bravo:
        value=Alpha()
    Bravo.__name__="My"+Alpha.__name__
    return Bravo
# Клас з декоратором:
@F
class Charlie:
    def __init__(self):
        self.number=123
    def show(self):
        print("Поле number:",self.number)
# Створення об'єкта:
obj=Charlie()
# Перевірка результату:
obj.value.show()
print("Клас об'єкта obj:",obj.__class__.__name__)
print("Клас поля value:",obj.value.__class__.__name__)
```

Якщо перед описанням класу (наприклад `Charlie`) вказаний декоратор (наприклад, `@F`), то до класу застосовується перетворення, яке можна описати інструкцією `Charlie=F(Charlie)`

```
Поле number: 123
Клас об'єкта obj: MyCharlie
Клас поля value: Charlie
```

Завдання

Напишіть програму, в якій описується клас з наступними характеристиками. У класу є конструктор, якому (крім посилання на об'єкт виклику) передаються два значення. Ці значення присвоюються полям об'єкта класу. В класі має бути описаний метод, при виклику якого відображаються значення полів класу. Перевірте функціональність класу, створивши на його основі декілька об'єктів

Домашнє завдання

Напишіть програму, в якій створюється ланцюжок об'єктов. Для створення ланцюжка об'єктів запропонуйте функцію, при виклику якої аргументом передається ціле число, яке визначає кількість об'єктів у ланцюжку