PYTHON

Advanced OOP

OOP Foundations

OOP FOUNDATIONS

Classes, Instances, Attributes, Methods

Introduction

OOP FOUNDATIONS

- Conceptos básicos:
  - Clase
  - Instancia
  - Objeto
  - Atributo
  - Método
  - Tipo

OOP FOUNDATIONS

- Conceptos básicos:
  - self
  - super()

OOP FOUNDATIONS

- Definición de clase:

```
class Duck:
    def __init__(self, height, weight, sex):
        self.height = height
        self.weight = weight
        self.sex = sex

    def walk(self):
        pass

    def quack(self):
        return print('Quack')
```

OOP FOUNDATIONS

- Instanciación:

```
duckling = Duck(height=10, weight=3.4, sex="male")
drake = Duck(height=25, weight=3.7, sex="male")
hen = Duck(height=20, weight=3.4, sex="female")
```

OOP FOUNDATIONS

- Atributos:
  - Se consideran atributos de un objeto:
    - Variables.
    - Métodos (atributos *callables*)
  - Se pueden añadir nuevos atributos a los objetos ya creados.
  - El acceso a los atributos se realiza utilizando la notación *objeto.atributo* o las funciones ***getattr()*** y ***setattr()***:

```
getattr(drake, 'height')
```

  - Una invocación a un método no existente en un objeto provoca un **AttributeError**:
    - drake.eat()→*AttributeError: 'Duck' object has no attribute 'eat'*

OOP FOUNDATIONS

- Type:
  - La información sobre el tipo de una clase (siempre será type), o de las instancias, atributos o métodos se encuentra en el atributo especial __class__.

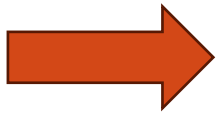
```
class Duck:
    def __init__(self, height, weight, sex):
        self.height = height
        self.weight = weight
        self.sex = sex

    def walk(self):
        pass

    def quack(self):
        return print('Quack')

duckling = Duck(height=10, weight=3.4, sex="male")
drake = Duck(height=25, weight=3.7, sex="male")
hen = Duck(height=20, weight=3.4, sex="female")

print(Duck.__class__)
print(duckling.__class__)
print(duckling.sex.__class__)
print(duckling.quack.__class__)
```



```
<class 'type'>
<class '__main__.Duck'>
<class 'str'>
<class 'method'>
```

OOP FOUNDATIONS

Working with class and instance data – instance variables

OOP FOUNDATIONS

- **Variables de instancia vs variables de clase.**
- **Variables de instancia:**
  - Se pueden crear en el constructor __init__
  - Se pueden crear durante la vida del objeto.
  - Se pueden eliminar durante la vida del objeto.
  - Acceso *nombre_objeto.nombre_variable* o con *self.nombre_variable* desde dentro de la clase.
  - El atributo especial __dict__ aplicado a un objeto muestra las variables de instancia del mismo, tanto las construidas en el método __init__ como las construidas durante la vida del objeto (no muestra las de clase).

```
class Demo:
    def __init__(self, value):
        self.instance_var = value
```

OOP FOUNDATIONS

- **Variables de instancia vs variables de clase.**
- **Variables de clase:**
  - Se crean fuera del constructor.
  - Son independientes de las instancias.
  - Se acceden a través del nombre de la clase (*Clase.variable*) o a través del nombre del objeto (*instancia.variable*).
  - El atributo __dict__ aplicado a una clase muestra las variables de clase (no las de instancia)
  - Se pueden utilizar para almacenar metadatos comunes a todas las instancias.

```
class Demo:
    class_var = 'shared variable'

print(Demo.class_var)
print(Demo.__dict__)
```

OOP FOUNDATIONS

- **Variables de instancia vs variables de clase.**
- **Variables de clase:**
  - Aunque el acceso de lectura a una variable de clase se puede realizar a través del nombre de la clase o de una instancia, no ocurre igual para asignar el valor → Debe realizarse obligatoriamente a través del nombre de la clase (a través del nombre del objeto se crea una variable de instancia).

```
class Demo:
    class_var = 'shared variable'

print(Demo.class_var)
print(Demo.__dict__)
```