PYTHON

Advanced OOP

Metaprogramming

METAPROGRAMMING

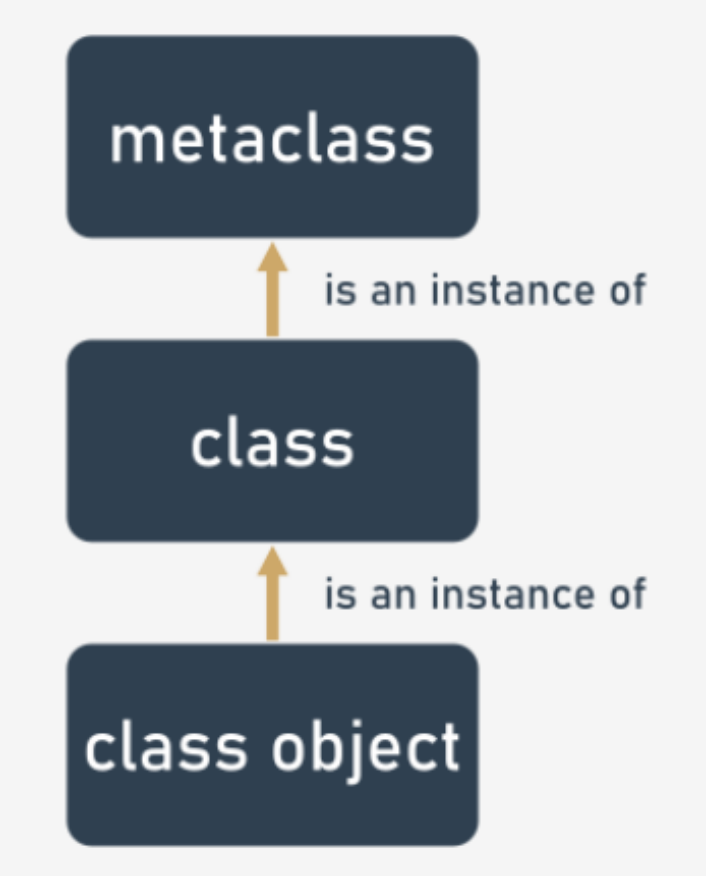
- Metaprogramación →Programas que escriben o modifican otros programas, incluídos ellos mismos.
  - Ejemplo: decoradores.
- Metaclase→Es una clase cuyas instancias son a su vez clases.
  - Útiles en la construcción de módulos o *frameworks*.
- Ejemplos:
  - Log
  - Registro de clases en el momento de la creación
  - Verificación de las interfaces
  - Adición automática de métodos
  - Adición automática de variables

METAPROGRAMMING

- La función type() permite obtener el tipo de cualquier objeto.
- La función type() aplicada a una clase siempre devuelve **type**.
  - type(3)→<class 'int'>
  - type(int)→<class 'type'>

METAPROGRAMMING

- Resumen:
  - Las metaclases se usan para crear clases;
  - Las clases se usan para crear objetos;
  - **type** es una clase.
  - Las metaclases son subclases de **type**.



METAPROGRAMMING

- Atributos especiales importantes:
  - __name__ → en una clase, contiene el nombre
  - __class__ → en un objeto o una clase, contiene información sobre la clase a la que pertenece
  - __bases__ → en una clase, contiene información sobre las clases de las que hereda directamente
  - __dict__→en un objeto o una clase, contiene un diccionario (u otro objeto de mapeo de tipos) de los atributos
  - La misma información almacenada en __class__ es proporcionada por la función **type() con un argumento**.

METAPROGRAMMING

- **type(x,y,z)** crea una nueva clase:
  - **x** especifica el nombre de la clase; este valor se convierte en el atributo __name__ de la clase;
  - **y** especifica una tupla de las clases base de las que se hereda la clase recién creada; este argumento se convierte en el atributo __bases__ de la clase;
  - **z** especifica un diccionario que contiene definiciones de métodos y variables para el cuerpo de la clase; los elementos de este argumento se convierten en el atributo __dict__ de la clase e indican el espacio de nombres de la clase.

METAPROGRAMMING

- **Ejemplo simple:**

```
Dog = type('Dog', (), {})
print('The class name is:', Dog.__name__)
print('The class is an instance of:', Dog.__class__)
print('The class is based on:', Dog.__bases__)
print('The class attributes are:', Dog.__dict__)
```

METAPROGRAMMING

- **Ejemplo completo:**

```
def bark(self):
    print('Woof, woof')

class Animal:
    def feed(self):
        print('It is feeding time!')

Dog = type('Dog', (Animal, ), {'age':0, 'bark':bark})

print('The class name is:', Dog.__name__)
print('The class is an instance of:', Dog.__class__)
print('The class is based on:', Dog.__bases__)
print('The class attributes are:', Dog.__dict__)

doggy = Dog()
doggy.feed()
doggy.bark()
```

METAPROGRAMMING

- Esta forma de crear clases, utilizando la función type, es para la forma de Python de crear clases utilizando la instrucción class:
  - después de que se ha identificado la instrucción **class** y se ha ejecutado el cuerpo de la clase, se ejecuta el código **class = type(,,)**;
  - **type** es responsable de llamar al método __**call**__ al crear la instancia de clase; este método llama a otros dos métodos:
    - __**new**__**()**, responsable de crear la instancia de clase en la memoria de la computadora; este método se ejecuta antes de __**init**__**()**;
    - __**init**__**()**, responsable de la inicialización del objeto.

- Las metaclases generalmente implementan estos dos métodos (__init__, __new__), tomando el control del procedimiento de creación e inicialización de una nueva instancia de clase. Las clases reciben una nueva capa de lógica.

METAPROGRAMMING

- Creación de una metaclase:
  - Debe heredar de **type**.
  - Implementar el método **__new__**, encargado de crear el objeto:
    - El nombre del parámetro **mcs** es una convención

```
class My_Meta(type):
    def __new__(mcs, name, bases, dictionary):
        obj = super().__new__(mcs, name, bases, dictionary)
        obj.custom_attribute = 'Added by My_Meta'
        return obj
```

METAPROGRAMMING

- Creación de una metaclase:
  - La clase que se crea a partir de la metaclase, debe heredar de esta, indicando el valor correspondiente al argumento **metaclass**.

```
class My_Object(metaclass=My_Meta):
    pass

print(My_Object.__dict__)
```

METAPROGRAMMING

- Ejemplo de metaclase:
  - Garantizar que todas las clases creadas a partir de una metaclase contienen un método concreto.
  - Si la clase creada a partir de la metaclase no contiene determinado método, este se le asigna.

METAPROGRAMMING

- Ejemplo de metaclase:

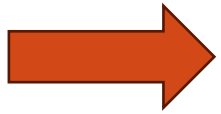
```
def metodo_obligatorio(self):
    print('Implementación por defecto')

class Mi_Metaclase(type):
    def __new__(mcs, name, bases, dictionary):
        if 'metodo_obligatorio' not in dictionary:
            dictionary['metodo_obligatorio'] = metodo_obligatorio
        obj = super().__new__(mcs, name, bases, dictionary)
        return obj

class Clase1(metaclass=Mi_Metaclase):
    pass

class Clase2(metaclass=Mi_Metaclase):
    def metodo_obligatorio(self):
        print('Implementación concreta')

objeto1 = Clase1()
objeto1.metodo_obligatorio()
objeto2 = Clase2()
objeto2.metodo_obligatorio()
```



```
Implementación por defecto
Implementación concreta
```