Metaprogramación

Conceptos generales sobre metaprogramación

Índice

¿Qué es la metaprogramación?

- Técnica consistente en la manipulación de programas desde código.
- Con la metaprogramación es posible:
  - Obtener información sobre una clase
  - Manipular el comportamiento de una clase
  - Implementar Metaclases personalizadas para la creación de clases

Atributos implícitos

- Toda clase proporciona una serie de atributos que permiten obtener información sobre la misma en tiempo de ejecución.
- Entre ellos:
  - __class__: Clase a la que pertenece un objeto
  - __dict__: Diccionario con todo el contenido de la clase
  - __module__: Módulo al que pertenece la clase
  - __bases__: Lista de clases que hereda

```
class Test:
    data=100
    def __init__(self):
        pass
    def metodo1(self):
        print("metodo1")
ob=Test()
print(ob.__class__)
print(Test.__dict__)
print(ob.__module__)
```

```
<class '__main__.Test'>
{'__module__': '__main__', 'data': 100, '__init__': <function Test.__init__ at 0x000002223FCDDA60>, 'metodo1': <function Test.metodo1 at
0x000002223FCDDB70>, '__dict__': <attribute '__dict__' of 'Test' objects>, '__weakref__': <attribute '__weakref__' of 'Test' objects>, '__doc__':
None}
__main__
```

Tipo de una clase

- Toda clase Python es creada a partir de la metaclase type
- Una clase es, por tanto, un objeto de la clase type

```
class Test:
    pass
ob=Test()
print(type(ob)) #Test
print(type(Test)) #type
```

- type es a su vez un objeto de si misma