**Python 进阶 - Python类的内置属性**

内置属性：当创建一个类后，类所自带的属性，叫类的内置属性。

" 注意：有的属性是只有类有，有的属性是类和对象都有！"

|  |  |
| --- | --- |
| 属性 | 含义 |
| \_\_name\_\_ | 当前定义的【类】的名字 |
| \_\_module\_\_ | 【类或对象】所属的模块名 |
| \_\_dict\_\_ | 【类或对象】的属性（包含一个字典，由类的数据属性组成） |
| \_\_doc\_\_ | 【类或对象】的文档字符串 - 一般写在class 类下面 |
| \_\_base\_\_ | 当前【类】的父类 |
| \_\_bases\_\_ | 当前【类】所有父类构成的元组 |
| \_\_file\_\_ | \_\_file\_\_会返回当前文件的绝对路径，而在终端直接运行时，则返回的是文件本身 |
| \_\_class\_\_ | 返回所在的类。用法：实例或者函数.\_\_class\_\_ |
| \_\_mro\_\_ | 返回继承顺序：  class B(object):  pass    class A(B):  pass    print(A.\_\_mro\_\_) # (<class '\_\_main\_\_.A'>, <class '\_\_main\_\_.B'>, <class 'object'>)。  如果只是单继承，继承层次不深的话很好理解，但其实并不是单纯的从下到上的顺序，感兴趣的同学可以去了解一下，这里就不展开了。 |
| \_\_metaclass\_\_ | 元类 |

示例：

class Person:

"""A class name Person"""

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

p1 = Person('Aaron', 20)

1. **\_\_name\_\_**: 返回当前定义的【类】的名字

print(Person.\_\_name\_\_) *# 输出：Person*

 2.**\_\_module\_\_**: 返回当前【类或对象】所在模块的名字

1. print(Person.\_\_module\_\_) *# 输出：\_\_main\_\_*
3. print(p1.\_\_module\_\_) *# 输出：\_\_main\_\_*

 3. **\_\_dict\_\_**: 【类或对象】的属性（包含一个字典，由类的数据属性组成）

print(Person.\_\_dict\_\_)

# 输出

{

'\_\_module\_\_': '\_\_main\_\_',

'\_\_doc\_\_': 'A class name Person',

'\_\_init\_\_': <function Person.\_\_init\_\_ at 0x7f46262ff700>,

'\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'Person' objects>,

'\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'Person' objects>

}

print(p1.\_\_dict\_\_)

# 输出

{

'name': 'Aaron',

'age': 20

}

4. **\_\_doc\_\_**：【类或对象】的文档字符串 。一般写在class 类下面，对该类进行一些注释和说明，但是一般只会类来引用这个属性，很少有用对象来引用的。

print(Person.\_\_doc\_\_) # 输出：A class name Person

print(p1.\_\_doc\_\_) # 输出：A class name Person

5. **\_\_base\_\_**：当前【类】的父类

print(Person.\_\_base\_\_) *# 输出：<class 'object'>*

6. **\_\_bases\_\_**：当前【类】的所有父类构成的**元组**

print(Person.\_\_bases\_\_) *# 输出：(<class 'object'>,)*

**文**