Python Module Attributes: name, doc, file, dict

Python module has its attributes that describes it. Attributes perform some tasks or contain some information about the module. Some of the important attributes are explained below:

__name__ Attribute

The `__name__` attribute returns the name of the module. By default, the name of the file (excluding the extension .py) is the value of __name__attribute.

```
>>> import math
>>> math.__name__
'math'
```

In the same way, it gives the name of your custom module.

```
>>> hello.__name__
'hello'
```

However, this can be modified by assigning different strings to this attribute. Change hello.py as shown below.

Example: hello.py

```
def SayHello(name):
    print ("Hi {}! How are you?".format(name))
    return
__name__="SayHello"
```

And check the `__name__` attribute now

```
>>> import hello
>>> hello.__name__
'SayHello'
```

The value of the `__name__` attribute is set to `__main__` when checked on the interpreter.

```
>>> __name__
'__main__'
```

When we run any Python script (i.e. a module), its `__name__` attribute is also set to `__main__`. For example, create the following welcome.py in IDLE.

Example: welcome.py

```
print ("__name__ = ", __name__)
```

Run the above welcome.py in IDLE by pressing F5. You will see the following result.

Output in IDLE:
```
>>> __name__ =  __main__
```

However, when this module is imported, its `__name__` is set to its filename. Now, import the welcome module in the new file test.py with the following content.

Example: test.py
```
import welcome

print ("__name__ = ", __name__)
```

Now run the test.py in IDLE by pressing F5. The `__name__` attribute is now "welcome".

Output in IDLE:
```
>>> __name__ =  welcome
```

This attribute allows a Python script to be used as an executable or as a module.

The following script contains a `fibo()` function which generates numbers in a Fibonacci series up to a given number. The script `fibo.py` checks whether it is executed from the interpreter by its `__name__` attribute. If it is equal to "__main__", then it means it is being executed from the interpreter and if it is fibo, then it is being used in another module using the import statement.

Example: fibo.py
```
"""Fibonacci series module"""
def fibo(n):
    FiboList=[]
    a,b=0,1
    while b<n:
        FiboList.append(b)
        a, b = b, a+b
    return FiboList
if __name__== "__main__":
    print ("Fibonacci series", fibo(100))
```

__doc__ Attribute

The **__doc__** attribute denotes the documentation string (docstring) line written in a module code.

```
>>> import math
>>> math.__doc__
'This module is always available. It provides access to the
mathematical functions defined by the C standard.'
```

Consider the the following script is saved as test.py module.

Example: test.py
```
"""This is docstring of test module"""
```

```
def SayHello(name):
    print ("Hi {}! How are you?".format(name))
    return
```

The `__doc__` attribute will return a string defined at the beginning of the module code.

```
>>> import test
>>> test.__doc__
'This is docstring of test module'
```

__file__ Attribute

`__file__` is an optional attribute which holds the name and path of the module file from which it is loaded.

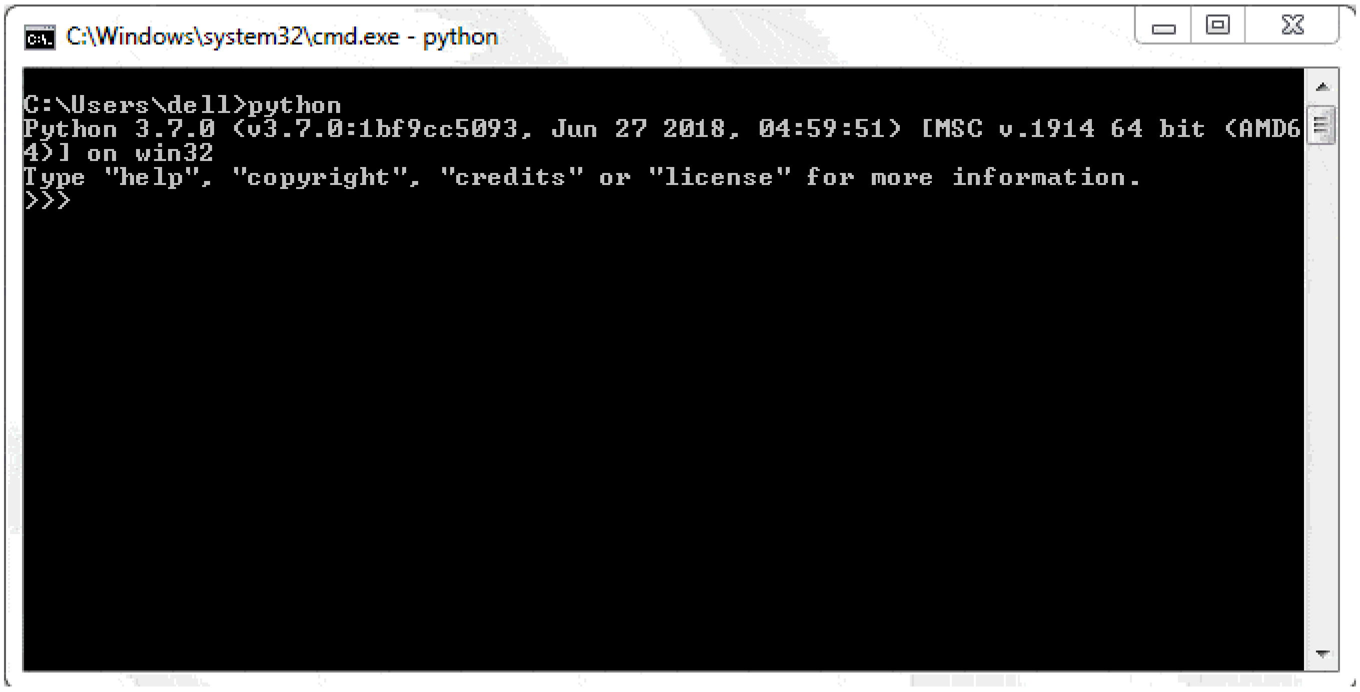
```
>>> import io
>>> io.__file__
'C:\\python37\\lib\\io.py'
```

__dict__ Attribute

The `__dict__` attribute will return a dictionary object of module attributes, functions and other definitions and their respective values.

```
>>> import math
>>> math.__dict__
{'__name__': 'math', '__doc__': 'This module is always
available. It provides a ccess to the\nmathematical functions
defined by the C standard.', '__package__': '', '__loader__':
<class '_frozen_importlib.BuiltinImporter'>, '__spec__': Modu
leSpec(name='math', loader=<class
'_frozen_importlib.BuiltinImporter'>, origin=' built-in'),
'acos': <built-in function acos>, 'acosh': <built-in function
acosh> , 'asin': <built-in function asin>, 'asinh': <built-in
function asinh>, 'atan': <built-in function atan>, 'atan2':
<built-in function atan2>, 'atanh': <built-in function atanh>,
'ceil': <built-in function ceil>, 'copysign': <built-in functi
on copysign>, 'cos': <built-in function cos>, 'cosh': <built-in
function cosh>, 'degrees': <built-in function degrees>, 'erf':
<built-in function erf>, 'erfc': <built-in function erfc>,
'exp': <built-in function exp>, 'expm1': <built-in fun ction
expm1>, 'fabs': <built-in function fabs>, 'factorial': <built-in
function factorial>, 'floor': <built-in function floor>, 'fmod':
<built-in function fmod> , 'frexp': <built-in function frexp>,
'fsum': <built-in function fsum>, 'gamma': <built-in function
gamma>, 'gcd': <built-in function gcd>, 'hypot': <built-in f
unction hypot>, 'isclose': <built-in function isclose>,
'isfinite': <built-in fu nction isfinite>, 'isinf': <built-in
function isinf>, 'isnan': <built-in functio n isnan>, 'ldexp':
<built-in function ldexp>, 'lgamma': <built-in function lgamm
a>, 'log': <built-in function log>, 'log1p': <built-in function
log1p>, 'log10': <built-in function log10>, 'log2': <built-in
function log2>, 'modf': <built-in function modf>, 'pow': <built-
in function pow>, 'radians': <built-in function ra dians>,
'remainder': <built-in function remainder>, 'sin': <built-in
function si n>, 'sinh': <built-in function sinh>, 'sqrt':
<built-in function sqrt>, 'tan': < built-in function tan>,
'tanh': <built-in function tanh>, 'trunc': <built-in fun ction
trunc>, 'pi': 3.141592653589793, 'e': 2.718281828459045, 'tau':
6.28318530 7179586, 'inf': inf, 'nan': nan}
```

dir() is a built-in function that also returns the list of all attributes and functions in a module.



```
C:\Users\dell>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```