**Modules in Python**

A module is a collection of functions that are grouped together

Modules = Libraries

>>> dir()

['\_\_builtins\_\_', '\_\_doc\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_']

“builtins” – modules contains

Python’s built-in **functions** are actually in a module named \_\_builtins\_\_

The double underscores before and after the name signal that it’s a part of Python

To check the list of available functions print

>>> dir(\_\_builtins\_\_)

-------------------------------------------------------------------------------------

**>>> dir(\_\_builtins\_\_)**

['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '\_', '\_\_build\_class\_\_', '\_\_debug\_\_', '\_\_doc\_\_', '\_\_import\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_',

'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']

-------------------------------------------------------------------------------------

We can get help on a single function:

**>>> help('len')**

Help on built-in function len in module builtins:

len(...)

len(object) -> integer

Return the number of items of a sequence or mapping.

or on all functions at once:

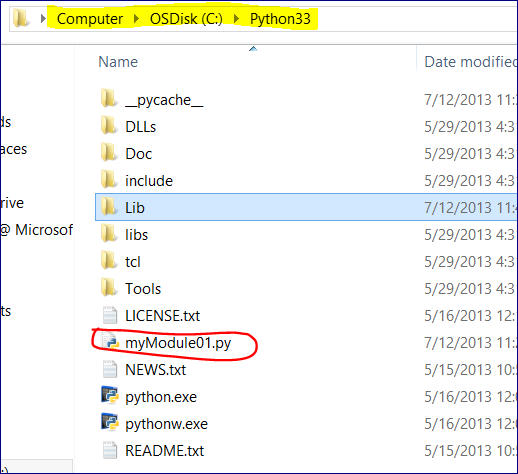
**>>> help(\_\_builtins\_\_)**

-------------------------------------------------------------------------------------

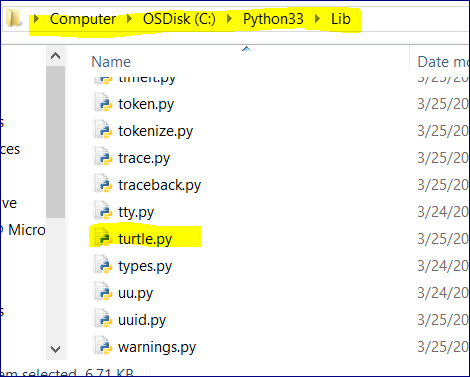
**To get the list of ALL available modules:**

Those modules can be yours, imported, or installed with Python by default.

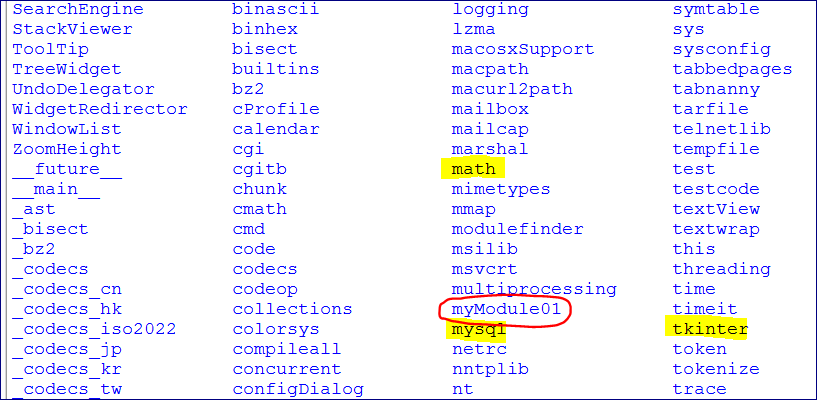
**Modules are available when installed in Python root folder: C:\Python33**



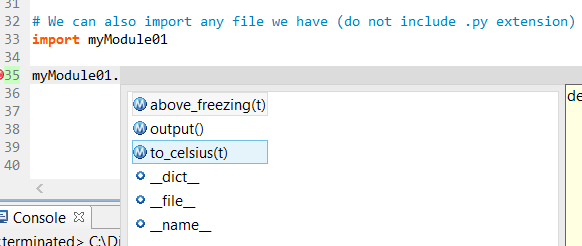
**OR in “lib” folder: C:\Python33\Lib**



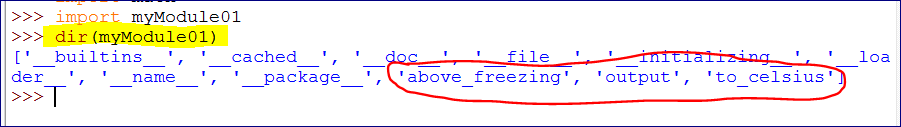
**>>> help('modules')**



---------------



We can run dir() on our own module that we just imported



The module is executed on the import.

If you import it again it won’t be executed again as it’s already imported.

Adding docstrings (Documentation Strings) when creating modules.

It will allow to access help on functions.

