



2



Python Quick Tips

Functions, Methods and Libraries



Python Quick Tips #2

Functions, Methods and Packages

Functions



In part 1, we already
used **functions**

Functions

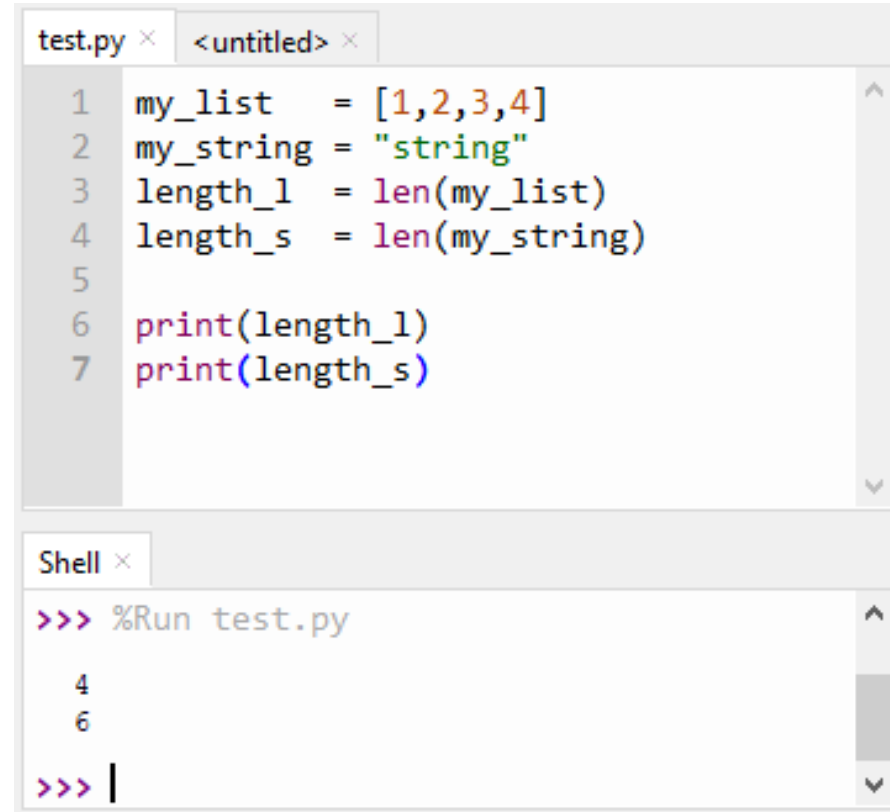
Syntax (typically)

Function(object)

Functions are **called by name**, and then passed data to operate upon

Functions

Example
`len(variable)`



The screenshot shows a Python IDE with two tabs: 'test.py' and '<untitled>'. The 'test.py' tab is active and contains the following code:

```
1 my_list = [1,2,3,4]
2 my_string = "string"
3 length_l = len(my_list)
4 length_s = len(my_string)
5
6 print(length_l)
7 print(length_s)
```

Below the code editor is a 'Shell' tab. It shows the command prompt running the script:

```
>>> %Run test.py
4
6
>>> |
```

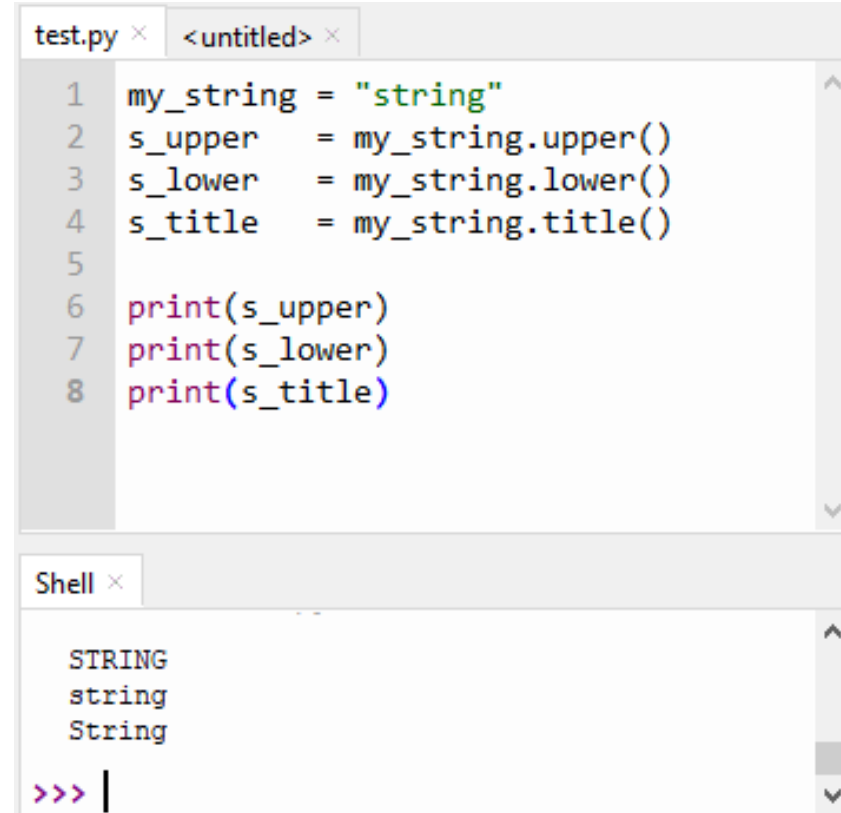
Methods

Syntax (typically)
`object.method()`

Methods are **called by name**, and associated with an object

Methods

Example string.upper()



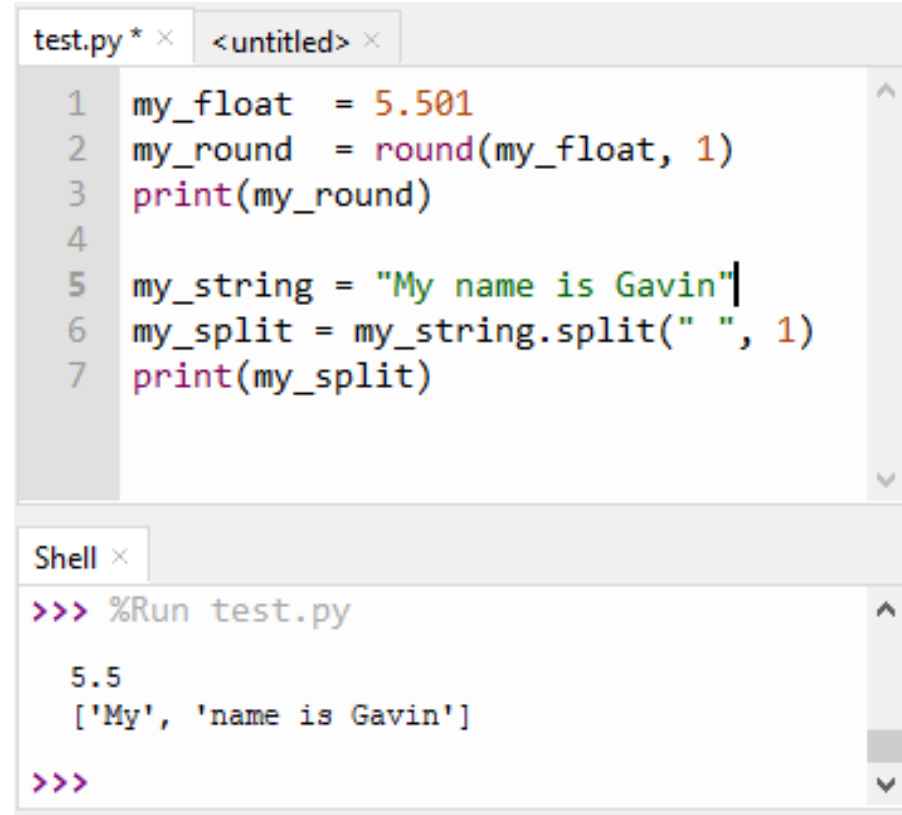
The screenshot shows a Python IDE with two panels. The top panel, titled 'test.py' and '<untitled>', contains a Python script with 8 lines of code. The script defines a variable 'my_string' with the value 'string', then uses the 'upper()', 'lower()', and 'title()' methods to create 's_upper', 's_lower', and 's_title' respectively. These three variables are then printed. The bottom panel, titled 'Shell', shows the output of the script: 'STRING', 'string', and 'String' on separate lines, followed by a prompt '>>>> |'.

```
test.py x <untitled> x
1 my_string = "string"
2 s_upper   = my_string.upper()
3 s_lower   = my_string.lower()
4 s_title   = my_string.title()
5
6 print(s_upper)
7 print(s_lower)
8 print(s_title)

Shell x
STRING
string
String
>>> |
```

(multiple, fields)

Some methods or functions require/allow multiple fields.



The screenshot shows a Python IDE with two tabs: 'test.py *' and '<untitled>'. The 'test.py' tab is active and contains the following code:

```
1 my_float = 5.501
2 my_round = round(my_float, 1)
3 print(my_round)
4
5 my_string = "My name is Gavin"
6 my_split = my_string.split(" ", 1)
7 print(my_split)
```

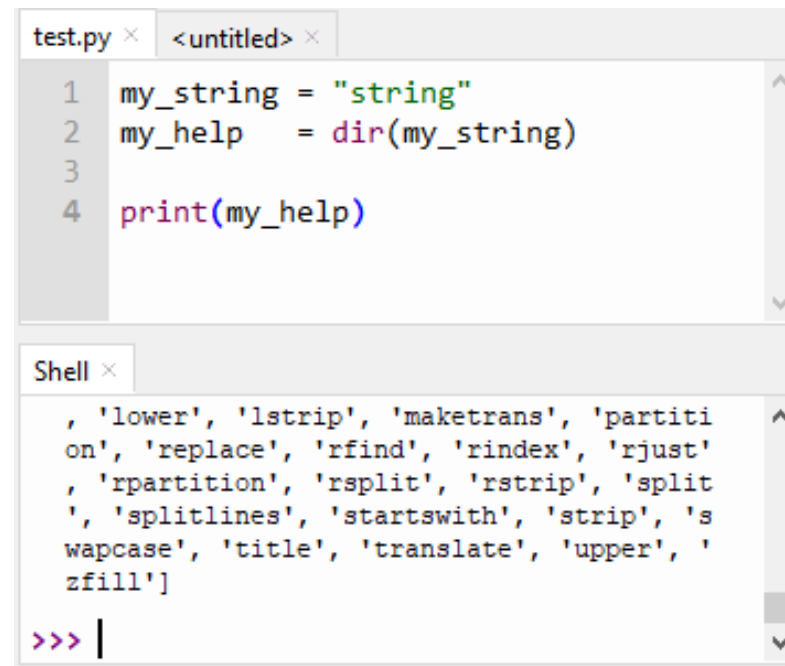
Below the code editor is a 'Shell' tab. It shows the command prompt running the script:

```
>>> %Run test.py
5.5
['My', 'name is Gavin']
>>>
```


Finding Help

Function `dir(object)`

Exposes all attributes of object



The screenshot shows a Python IDE with two panels. The top panel, titled 'test.py', contains a script with four lines of code: `1 my_string = "string"`, `2 my_help = dir(my_string)`, `3` (blank line), and `4 print(my_help)`. The bottom panel, titled 'Shell', shows the output of the script as a list of string methods: `['lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']`. The prompt `>>>` is visible at the bottom of the shell panel.

```
test.py × <untitled> ×  
1 my_string = "string"  
2 my_help = dir(my_string)  
3  
4 print(my_help)  
  
Shell ×  
, 'lower', 'lstrip', 'maketrans', 'partiti  
on', 'replace', 'rfind', 'rindex', 'rjust'  
, 'rpartition', 'rsplit', 'rstrip', 'split'  
, 'splitlines', 'startswith', 'strip', 's  
wapcase', 'title', 'translate', 'upper', '  
zfill']  
  
>>> |
```

Finding Help

Also works for libraries



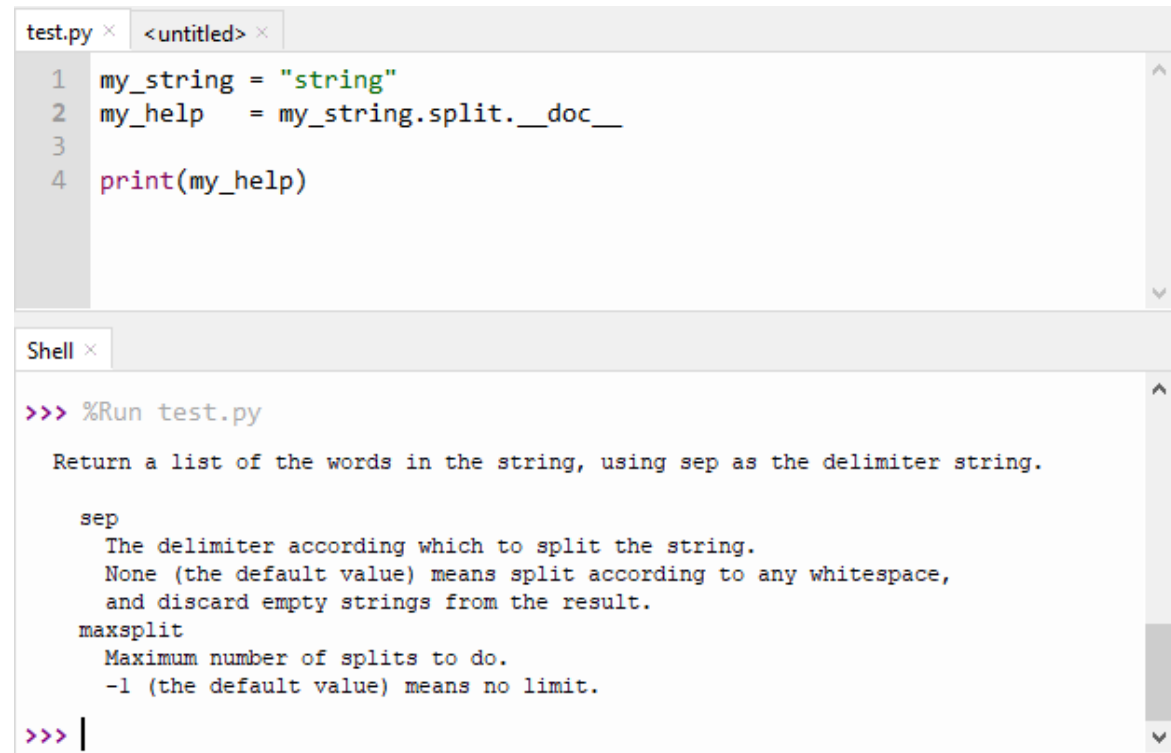
The screenshot shows a Python IDE with two panels. The top panel, titled 'test.py', contains a script that imports the pip module, uses the dir() function to list its attributes, and prints the result. The bottom panel, titled 'Shell', shows the execution of the script. The first run returns 'None', and the second run displays a list of attributes for the pip module.

```
test.py × <untitled> ×  
1 import pip  
2  
3 my_help = dir(pip)  
4  
5 print(my_help)  
  
Shell ×  
  
>>> %Run test.py  
None  
  
>>> %Run test.py  
['_builtins_', '__cached__', '__doc__', '__file__',  
'__loader__', '__name__', '__package__', '__path__',  
 '__spec__', '__version__']  
>>> |
```

Finding Help

Method
object.__doc__

Exposes documentation of object



```
test.py x <untitled> x
1 my_string = "string"
2 my_help   = my_string.split.__doc__
3
4 print(my_help)

Shell x
>>> %Run test.py

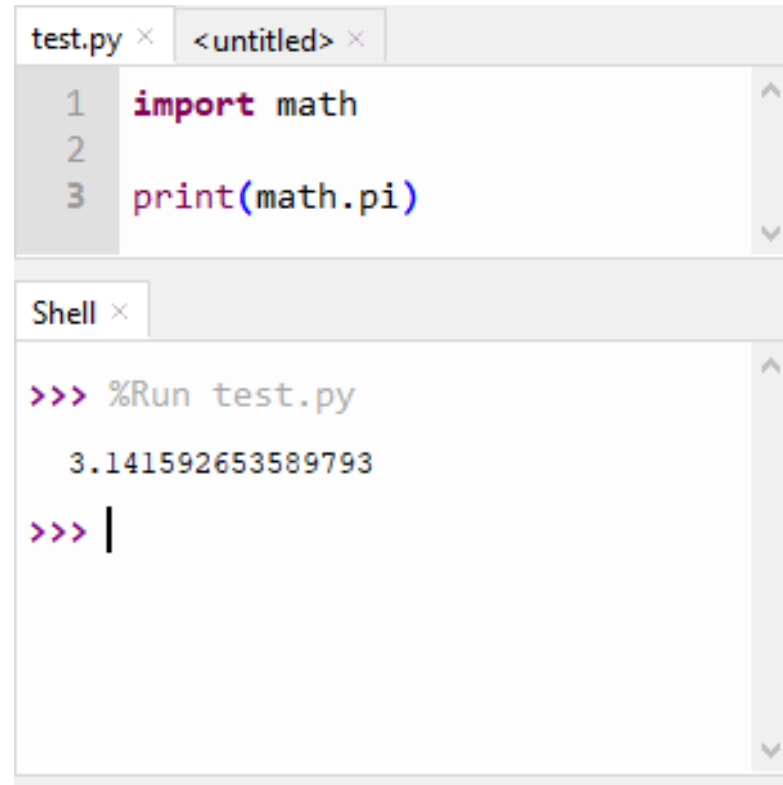
Return a list of the words in the string, using sep as the delimiter string.

sep
The delimiter according which to split the string.
None (the default value) means split according to any whitespace,
and discard empty strings from the result.
maxsplit
Maximum number of splits to do.
-1 (the default value) means no limit.

>>> |
```

Importing Libraries

Syntax
import library

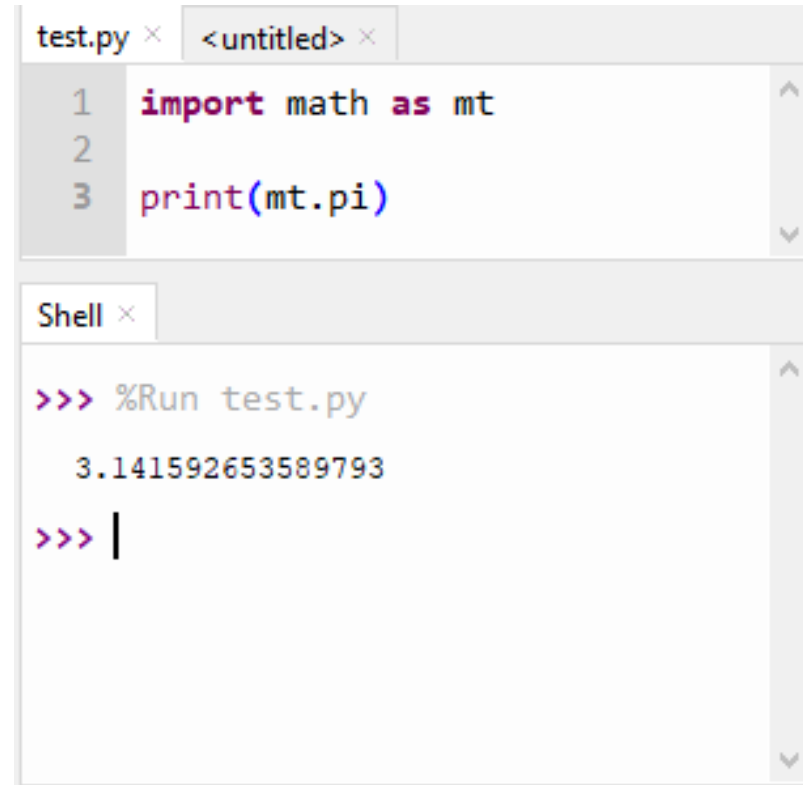


The screenshot shows a Python IDE interface. At the top, there are two tabs: 'test.py' and '<untitled>'. The 'test.py' tab is active, displaying a script with three lines of code: `1 import math`, `2` (blank line), and `3 print(math.pi)`. Below the code editor is a 'Shell' tab, which shows the output of running the script. The shell prompt is `>>>`, followed by the command `%Run test.py`. The output is `3.141592653589793`. The shell prompt is now `>>> |`, indicating it is ready for the next command.

```
test.py × <untitled> ×  
1 import math  
2  
3 print(math.pi)  
  
Shell ×  
  
>>> %Run test.py  
3.141592653589793  
>>> |
```

Importing Libraries (Alias)

Syntax
import library **as** alias



The screenshot shows a code editor with two tabs: 'test.py' and '<untitled>'. The 'test.py' tab is active and contains the following code:

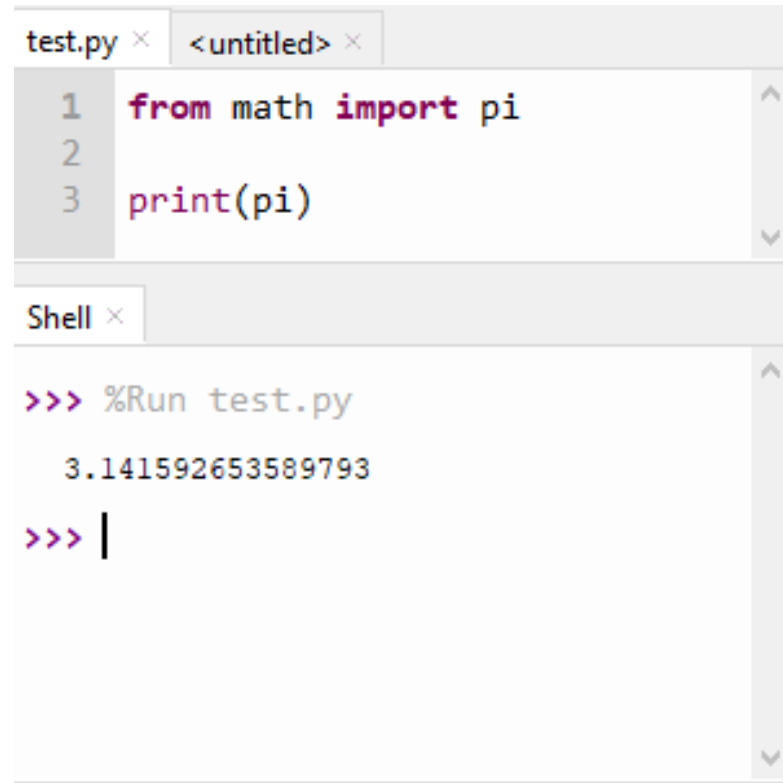
```
1 import math as mt
2
3 print(mt.pi)
```

Below the code editor is a 'Shell' window. It shows the command prompt running the script:

```
>>> %Run test.py
      3.141592653589793
>>> |
```

Importing Attributes

Syntax
from library **import** attribute



The screenshot shows a Python IDE with two tabs: 'test.py' and '<untitled>'. The 'test.py' tab is active and contains the following code:

```
1 from math import pi
2
3 print(pi)
```

Below the code editor is a 'Shell' tab, which is also active. It displays the output of running the script:

```
>>> %Run test.py
3.141592653589793
>>> |
```

Importing Libraries (Limited)

Note
You lose context



The screenshot shows a Python IDE with two tabs: 'test.py' and '<untitled>'. The 'test.py' tab is active and contains the following code:

```
1 from math import pi
2
3 print(math.pi)
```

Below the code editor is a 'Shell' window. It shows the command prompt running the script, followed by a traceback error:

```
>>> %Run test.py
Traceback (most recent call last):
  File "C:\Users\Gavin\Desktop\test.py", line 3, in <module>
    print(math.pi)
NameError: name 'math' is not defined
>>> |
```

The error message indicates that the 'math' module is not defined, which is the result of the limited import statement used in the script.

Common Packages

Library Name

Reference

Regular expressions

re

Maths

math

DateTime

datetime

Operating System

os

PIP

pip

Iteration tools

itertools

Pillow

pil

Matplotlib

matplotlib

Numerical Python

Numpy (np)

Pandas

Pandas (pd)

More Specific Packages

Library Name

Reference

Keras

keras

Tensor Flow

tensorflow (tf)

PyTorch

PyTorch

NLTK

nltk

Delorean

delorean

SciPy

scipy

Seaborn

seaborn (sb)



Next on #3
Working with Lists



2



Python Quick Tips

Functions, Methods and Libraries