In [118…
```
# single-line comments can be done with the pound character
"""
multi-line
comments
can be done like this
"""
```
Out[118…
```
'\nmulti-line\ncomments\ncan be done like this\n'
```

Numbers

In [119…
```
type(2)
```
Out[119…
```
int
```
In [120…
```
type(2.0)
```
Out[120…
```
float
```
In [121…
```
2 + 2  # addition
```
Out[121…
```
4
```
In [122…
```
2 - 2  # subtraction
```
Out[122…
```
0
```
In [123…
```
2 * 2  # multiplication
```
Out[123…
```
4
```
In [124…
```
2 / 2  # float division
```
Out[124…
```
1.0
```
In [125…
```
2 // 2  # integer division
```
Out[125…
```
1
```
In [126…
```
2 ** 3  # exponents
```
Out[126…
```
8
```
In [127…
```
2 ** 0.5  # square root
```
Out[127…
```
1.4142135623730951
```
In [128…
```
5 % 2  # modulo operator (calculates the remainder)
```

```
Out[128… 1
```

```
In [129… int(2.0)  # conversion to an integer
```

```
Out[129… 2
```

```
In [130… int(2.1)  # this rounds down
```

```
Out[130… 2
```

```
In [131… round(2.11)  # rounds to the nearest integer
```

```
Out[131… 2
```

```
In [132… round(2.11, ndigits=1)  # round to the nearest 0.1
```

```
Out[132… 2.1
```

```
In [133… float(2)  # conversion to float
```

```
Out[133… 2.0
```

```
In [134… import math
          math.pi
```

```
Out[134… 3.141592653589793
```

```
In [135… math.lcm(2, 3, 5)  # least common multiple
```

```
Out[135… 30
```

Strings

```
In [136… 'a string'
```

```
Out[136… 'a string'
```

```
In [137… "a string"
```

```
Out[137… 'a string'
```

```
In [138… print("""
          multi-
          line
          string
          """)
```

```
multi-
line
string
```

In [139…
```
'a' + 'string'  # concatenate strings
```
Out[139…
```
'astring'
```

In [140…
```
'a' * 2  # repeat strings
```
Out[140…
```
'aa'
```

In [141…
```
str(2)  # convert a number to a string
```
Out[141…
```
'2'
```

In [142…
```
# raw string -- the last backslash must be escaped with another backslash if
r'C:\Users\Me\A folder\\'
r'C:\Users\Me\A folder\a file.txt'
```
Out[142…
```
'C:\\Users\\Me\\A folder\\a file.txt'
```

String indexing

In [143…
```
'a string'[0]  # first character of a string
```
Out[143…
```
'a'
```

In [144…
```
'a string'[-1]  # last character of a string
```
Out[144…
```
'g'
```

In [145…
```
'a string'[0:4]  # index a string to get first 4 characters
```
Out[145…
```
'a st'
```

In [146…
```
'a string'[:4]  # index a string to get first 4 characters
```
Out[146…
```
'a st'
```

In [147…
```
'a string'[::2]  # get every other letter
```
Out[147…
```
'asrn'
```

In [148…
```
'a string'[::-1]  # reverse the string
```
Out[148…
```
'gnirts a'
```

In [149…
```
'a string'[:5:2]  # every other letter in the first 5 characters
```
Out[149…
```
'asr'
```

Built-in string methods

In [150…
```
'-'.join(['this', 'is', 'a', 'test'])
```
Out[150…
```
'this-is-a-test'
```

In [151…
```
'this is a test'.split()
```
Out[151…
```
['this', 'is', 'a', 'test']
```

In [152…
```
'\t\n   - remove left'.lstrip()  # remove whitespace on the left
```
Out[152…
```
'- remove left'
```

In [153…
```
'\t\n   - remove left'.rstrip()  # remove whitespace on the right
```
Out[153…
```
'\t\n   - remove left'
```

In [154…
```
'testtest - remove left'.lstrip('test')  # remove all instances of 'test' fr
```
Out[154…
```
' - remove left'
```

In [155…
```
'testtest - remove left'.lstrip('tes')  # remove all instances of 'tes' char
```
Out[155…
```
' - remove left'
```

In [156…
```
'testtest - remove left'.removeprefix('test')  # remove one instance of 'tes
```
Out[156…
```
'test - remove left'
```

In [157…
```
'testtest - remove left'.removesuffix('left')
```
Out[157…
```
'testtest - remove '
```

In [158…
```
f'string formatting {2 + 2}'
```
Out[158…
```
'string formatting 4'
```

In [159…
```
print('tabs\tand\nnewlines')
```
```
tabs	and
newlines
```

In [160…
```
print(r'tabs\tand newlines\n')
```
```
tabs\tand newlines\n
```

In [161…
```
print('\t\n   - tabs and newlines')  # tab and newline at the beginning of a
```
```
	
   - tabs and newlines
```

Variables

In [162…
```
books = 1
```

In [163…
```
books  # print out our variable
```

Out[163…
```
1
```

In [164…
```
books = books + 1
books
```

Out[164…
```
2
```

In [165…
```
books += 1
books
```

Out[165…
```
3
```

In [166…
```
books -= 1
books
```

Out[166…
```
2
```

In [167…
```
books *= 2
books
```

Out[167…
```
4
```

In [168…
```
books /= 2
books
```

Out[168…
```
2.0
```

In [169…
```
books **= 2
books
```

Out[169…
```
4.0
```

In [170…
```
books %= 2
books
```

Out[170…
```
0.0
```

In [171…
```
# concatenate two string variables
a = 'string 1'
b = 'another string'
a + b
```

Out[171…
```
'string 1another string'
```

In [172…
```
# check variable type
type(a)
```

Out[172…
```
str
```

In [173…
```
# don't do this!
# type = 'test'
# type(a)  # if you try this, the type() function will no longer work
```

Lists, Tuples, Sets, and Dictionaries

In [174…
```
# a basic list
[1, 2, 3]
```

Out[174…
```
[1, 2, 3]
```

In [175…
```
# lists can contain different data types
[1, 'a', 3]
```

Out[175…
```
[1, 'a', 3]
```

In [176…
```
# lists can contain other lists
[1, [1, 2, 3], 3]
```

Out[176…
```
[1, [1, 2, 3], 3]
```

In [177…
```
# join lists
[1, 2, 3] + [4, 5]
```

Out[177…
```
[1, 2, 3, 4, 5]
```

In [178…
```
# repeat a list
[1, 2, 3] * 2
```

Out[178…
```
[1, 2, 3, 1, 2, 3]
```

In [179…
```
# get the length of a list
len([1, 2, 3])
```

Out[179…
```
3
```

In [180…
```
# make a blank list and add the element '1' to it
a_list = []
a_list.append(1)
a_list
```

Out[180…
```
[1]
```

In [181…
```
# sort in-place
a_list = [1, 3, 2]
a_list.sort()
a_list
```

Out[181…
```
[1, 2, 3]
```

In [182…
```
# sort
a_list = [1, 3, 2]
sorted(a_list)
```
Out[182…
```
[1, 2, 3]
```
In [183…
```
# indexing: [start:stop:step]
a_list = [1, 2, 3, 4, 5]
a_list[0]
```
Out[183…
```
1
```
In [184…
```
a_list[-1]
```
Out[184…
```
5
```
In [185…
```
a_list[0:3]
```
Out[185…
```
[1, 2, 3]
```
In [186…
```
a_list[:3]
```
Out[186…
```
[1, 2, 3]
```
In [187…
```
a_list[::2]
```
Out[187…
```
[1, 3, 5]
```
In [188…
```
a_list[0:3:2]
```
Out[188…
```
[1, 3]
```
In [189…
```
# reverse a list
a_list[::-1]
```
Out[189…
```
[5, 4, 3, 2, 1]
```

Tuples

In [190…
```
a_tuple = (2, 3)
a_tuple
```
Out[190…
```
(2, 3)
```
In [191…
```
tuple(a_list)
```
Out[191…
```
(1, 2, 3, 4, 5)
```

Sets

```
In [192…  set(a_list)

Out[192…  {1, 2, 3, 4, 5}

In [193…  a_set = {1, 2, 3, 3}
          a_set

Out[193…  {1, 2, 3}

In [194…  set_1 = {1, 2, 3}
          set_2 = {2, 3, 4}
          set_1.union(set_2)

Out[194…  {1, 2, 3, 4}

In [195…  set_1 | set_2

Out[195…  {1, 2, 3, 4}

In [196…  set_1.difference(set_2)

Out[196…  {1}

In [197…  # shorthand for different operator
          set_1 - set_2

Out[197…  {1}
```

Dictionaries

```
In [198…  a_dict = {'books': 1, 'magazines': 2, 'articles': 7}
          a_dict

Out[198…  {'books': 1, 'magazines': 2, 'articles': 7}

In [199…  a_dict['books']

Out[199…  1

In [200…  another_dict = {'movies': 4}
          a_dict | another_dict

Out[200…  {'books': 1, 'magazines': 2, 'articles': 7, 'movies': 4}

In [201…  a_dict['shows'] = 12

In [202…  a_dict

Out[202…  {'books': 1, 'magazines': 2, 'articles': 7, 'shows': 12}
```

Loops and Comprehensions

In [203…
```
a_list = [1, 2, 3]
for element in a_list:
    print(element)
```

```
1
2
3
```

In [204…
```
a_list = [1, 2, 3]
for index in range(len(a_list)):
    print(index)
```

```
0
1
2
```

This brings up the documentation for a function.

In [205…
```
?range
```

```
Init signature: range(self, /, *args, **kwargs)
Docstring:
range(stop) -> range object
range(start, stop[, step]) -> range object

Return an object that produces a sequence of integers from start (inclusive)
to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, ..., j-1.
start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
These are exactly the valid indices for a list of 4 elements.
When step is given, it specifies the increment (or decrement).
Type:           type
Subclasses:
```

In [206…
```
a_list = [1, 2, 3]
for index, element in enumerate(a_list):
    print(index, element)
```

```
0 1
1 2
2 3
```

In [207…
```
a_list = []
for i in range(3):
    a_list.append(i)

a_list
```

Out[207…
```
[0, 1, 2]
```

In [208…
```
# a list comprehension
a_list = [i for i in range(3)]
a_list
```

```
Out[208… [0, 1, 2]
```

```
In [209… a_dict = {'books': 1, 'magazines': 2, 'articles': 7}
         for key, value in a_dict.items():
             print(f'{key}:{value}')
```

```
books:1
magazines:2
articles:7
```

```
In [210… # a dictionary comprehension
         a_dict = {i: i ** 2 for i in range(1, 4)}
         a_dict
```

```
Out[210… {1: 1, 2: 4, 3: 9}
```

Booleans and Conditionals

```
In [211… books_read = 11
         books_read > 10
```

```
Out[211… True
```

```
In [212… none_var = None
         none_var is None
```

```
Out[212… True
```

```
In [213… books_read = 12
         if books_read < 10:
             print("You have only read a few books.")
         elif books_read >= 12:
             print("You've read lots of books!")
         else:
             print("You've read 10 or 11 books.")
```

```
You've read lots of books!
```

```
In [214… a = 'test'
         type(a) is str
```

```
Out[214… True
```

```
In [215… type(a) is not str
```

```
Out[215… False
```

```
In [216… 'st' in 'a string'  # check for a substring in a string
```

```
Out[216… True
```

```
In [217… a_set = {1, 2, 3}
         1 in a_set
```

```
Out[217… True
```

```
In [218… a_list = [1, 2, 3]
         1 in a_list
```

```
Out[218… True
```

```
In [219… a_dict = {1: 'val1', 2: 'val2', 3: 'val3'}
         1 in a_dict
```

```
Out[219… True
```

```
In [220… if 1 in a_set:
             print('1 is in there')
```

```
1 is in there
```

```
In [221… condition = False
         if condition != False:
             print('not false')
         elif condition == False:
             print('is false')
```

```
is false
```

Libraries and Imports

```
In [222… import time
         time.time()
```

```
Out[222… 1752353223.052392
```

```
In [223… import time as t
         t.time()
```

```
Out[223… 1752353223.0738618
```

```
In [224… import urllib.request
         urllib.request.urlopen('https://www.pypi.org')
```

```
Out[224… <http.client.HTTPResponse at 0x7f5f541b0460>
```

```
In [225… from urllib.request import urlopen
         urlopen('https://www.pypi.org')
```

```
Out[225… <http.client.HTTPResponse at 0x7f5f54172800>
```

```
In [226… # importing a function from a subpackage of a library, and aliasing it
         from urllib.request import urlopen as uo
         uo('https://www.pypi.org')
```

```
Out[226… <http.client.HTTPResponse at 0x7f5f541af700>
```

Functions

In [227…
```
def test_function(doPrint, printAdd='more'):
    """
    A demo function.
    """
    if doPrint:
        print('test' + printAdd)
    return printAdd
```

In [228…
```
value = test_function(True)
print(value)
```

```
testmore
more
```

In [229…
```
# brings up documentation for sorted()
?sorted
```

```
Signature: sorted(iterable, /, *, key=None, reverse=False)
Docstring:
Return a new list containing all items from the iterable in ascending order.

A custom key function can be supplied to customize the sort order, and the
reverse flag can be set to request the result in descending order.
Type:      builtin_function_or_method
```

In [230…
```
a_list = [2, 4, 1]
sorted(a_list, reverse=True)
```

Out[230…
```
[4, 2, 1]
```

In [231…
```
def test_function():
    """
    A demo function.
    """
    func_var = 'testing'
    print(func_var)
```

In [232…
```
test_function()
```

```
testing
```

In [233…
```
test_function
```

Out[233…
```
<function __main__.test_function()>
```

In [234…
```
add10 = lambda x, y: x + y + 10
add10(10, 3)
```

Out[234…
```
23
```

Classes

In [235…
```
class testObject:
    def __init__(self, attr):
        self.test_attribute = attr


    def test_function(self):
        print('testing123')
        print(f'testing{self.test_attribute}')
```

In [236…
```
to = testObject(123)
to.test_attribute
```

Out[236…
```
123
```

In [237…
```
to.test_function()
```

```
testing123
testing123
```

Here is another module from core Python.

In [238…
```
import calendar
# creates a new instance of a Calendar object
c = calendar.Calendar()
type(c)
```

Out[238…
```
calendar.Calendar
```

In [239…
```
# an attribute
c.firstweekday
```

Out[239…
```
0
```

In [240…
```
# a method/function
list(c.iterweekdays())
```

Out[240…
```
[0, 1, 2, 3, 4, 5, 6]
```

Multithreading and Multiprocessing

The multiprocessing and threading libraries are ways you will see many people recommend, but I prefer the concurrent.futures library myself. See the multiprocessing_demo.py file for more. Note that you should run the file like `python multiprocessing_demo.py`, and not in Jupyter notebook or IPython.