```
$ python3

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.

>>> # numbers

>>> type(5)
<class 'int'>

>>> type(5.0)
<class 'float'>

>>> # powers

>>> 2**8
256

>>> #  past value, '_' means the last printed expression

>>> 5 + 11
16

>>> _
16

>>> 5 + 11
16

>>> _ * 2
32

>>> # complex numbers

>>> a = 5 + 2j

>>> type(a)
<class 'complex'>

>>> a + 1
(6+2j)
```

```
>>> a - 5j
(5-3j)

>>> a.real
5.0

>>> a.imag
2.0

>>> a.conjugate()
(5-2j)

>>> a * a.conjugate()
(29+0j)

>>> # Strings

>>> s = 'goodbye CAT'

>>> s[0]
'g'

>>> s[1]
'o'

>>> len(s)
11

>>> s.upper()
'GOODBYE CAT'

>>> s.lower()
'goodbye cat'

>>> type(s)
<class 'str'>


>>> # SLICING - indexing starts with 0

>>> s[0:4]# first four elements
'good'

>>> s[1:4]
'ood'
```

```
>>> s[1:]# removes first element
'oodbye CAT'

>>> s[:]# all elements
'goodbye CAT'

>>> s[-1]       # last element
'T'

>>> s[4:-1]# removes first three elements and last element
'bye CA'

>>> s[-3:]# last three elements
'CAT'

>>> s[:-1]# removes last elements
'goodbye CA'

>>> s[:-2]              # removes last two elements
'goodbye C'

>>> s[:4] + s[4:]# complete string
'goodbye CAT'

>>> # STRINGS ARE IMMUTABLE - the elements of a string can not be
changed

>>> s[2]
'o'

>>> s[2] = 'b'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment


>>> # CONCATENATION

>>> 'abc' + 'xyz'
'abcxyz'

>>> s = 'abc'

>>> s + s
'abcabc'
```

```
>>> 2*s
'abcabc'

>>> 50*s
'abcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcab
cabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcab
cabcabcabcabcabcabc'

>>> # FUNCTIONS

>>> import my_functions

>>> # help(my_functions)

>>> # press 'q' to exit help screen

>>> dir(my_functions)
['__builtins__', '__cached__', '__doc__', '__file__',
'__loader__', '__name__', '__package__', '__spec__', 'funA',
'funB', 'my_fun']

>>> my_functions.funA(3.2)
6.4

>>> my_functions.funB(3)
7

>>> # To import just one function:

>>> from my_functions import funA

>>> funA(3.2)
6.4

>>> # If a file has been edited, then reload it via:

>>> import importlib

>>> importlib.reload(my_functions)
<module 'my_functions' from '/Users/selesi/MyDocuments/teaching/
DSP Lab (Python)/demos/demos 2018/00 - learning python/
my_functions.py'>

>>> # Lists

>>> v = [10, 5, 1, 200, 'abc']
```

```
>>> type(v)
<class 'list'>

>>> v[0]
10

>>> v[1]
5

>>> v[-1]
'abc'

>>> # SLICING

>>> v[0:3]

[10, 5, 1]

>>> v[1:]
[5, 1, 200, 'abc']

>>> v[:-1]
[10, 5, 1, 200]

>>> v[:]
[10, 5, 1, 200, 'abc']

>>> v[3:4]
[200]

>>> # compare with

>>> v[3]
200

>>> v = [10, 5, 1, 200, 'abc']

>>> # Compare:
... v[1] = []

>>> v
[10, [], 1, 200, 'abc']

>>> v[2:3] = []
```

```
>>> v
[10, [], 200, 'abc']

>>> # LIST ARE MUTABLE, the values of a list can be changed

>>> v[1] = 9999

>>> v
[10, 9999, 200, 'abc']

>>> v.append(13)
>>> v
[10, 9999, 200, 'abc', 13]

>>> v.reverse()
>>> v
[13, 'abc', 200, 9999, 10]

>>> v.append('dog')
>>> v
[13, 'abc', 200, 9999, 10, 'dog']

>>> v.append('cat')
>>> v
[13, 'abc', 200, 9999, 10, 'dog', 'cat']

>>> v[0]
13

>>> v[-1]
'cat'

>>> # replace an element
... v[3:4] = ['hello']

>>> # LOOPS

>>> for i in v:
...    print(i)

13
abc
200
hello
10
dog
```

```
cat

>>> for i in v:
...    print(i, 2*i)
...
13 26
abc abcabc
200 400
hello hellohello
10 20
dog dogdog
cat catcat

>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9

>>> for i in range(5, 10):
...     print(i)
...
5
6
7
8
9

>>> for i in range(0, 10, 2):
...     print(i)
...
0
2
4
6
8

>>> range(0, 10)    # iterable object
```

```
range(0, 10)

>>> list(range(0, 10))    # convert to list
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> # delete an element
... v = [10, 5, 1, 200, 'abc']

>>> del v[1]

>>> v
[10, 1, 200, 'abc']

>>> # TUPLES, like a list, but immutable
... # Define a tuple using parentheses ( and )
... a = (10, 20, 'hello', 'bye')

>>> type(a)
<class 'tuple'>

>>> len(a)
4

>>> a[0]
10

>>> a[1]
20

>>> a[-2:]   # last two elements
('hello', 'bye')

>>> a[:2]    # first two elements
(10, 20)

>>> a[1] = 12     # error because tuples are immutable
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment

>>> b = a[2:]

>>> b
('hello', 'bye')

>>> len(b)
```

```
2

>>> type(b)
<class 'tuple'>

>>> b = a[2:3]

>>> b
('hello',)

>>> len(b)
1

>>> type(b)
<class 'tuple'>

>>> # b is a tuple of length 1.  The comma indicates that it is a
tuple...
... # To create a tuple of length 1, use a comma:

>>> c = ('cat',)

>>> type(c)
<class 'tuple'>

>>> g = (10,)

>>> g
(10,)

>>> type(g)
<class 'tuple'>

>>> # short-cut to create tuples: omit parentheses

>>> a = (10, 20, 'hello', 'bye')
>>> a
(10, 20, 'hello', 'bye')

>>> a = 10, 20, 'hello', 'bye'
>>> a
(10, 20, 'hello', 'bye')

>>> # create a tuple of one element without parenthesis

>>> f = 10,
```

```
>>> f
(10,)

>>> type(f)
<class 'tuple'>

>>> len(f)
1

>>> # unpacking a sequence

>>> a = (10, 20)    # tuple

>>> x, y = a

>>> x
10

>>> y
20

>>> a = [10, 20]    # list

>>> x, y = a

>>> x
10

>>> y
20

>>> # Must have correct number of variables on left-hand-side

>>> a = [10, 20, 'hello']    # list

>>> x, y = a            # ERROR
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: too many values to unpack (expected 2)

>>> x, y, z = a         # it works

>>> # MATH

>>> import math
```

```
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__',
'__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf',
'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose',
'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10',
'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians',
'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

>>> math.pi
3.141592653589793

>>> math.cos(math.pi)
-1.0

>>> quit()
```