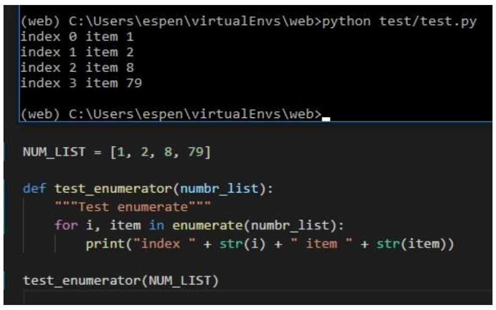
Python 3.4.7 Built-in Functions (subset)

https://docs.python.org/3.4/library/functions.html

The Python Standard Library

https://docs.python.org/3.4/library/index.html

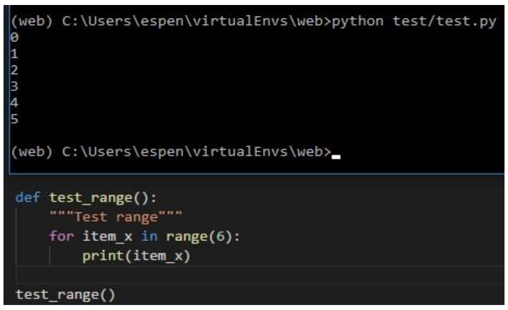
Enumerate

- 

```
(web) C:\Users\espen\virtualEnvs\web>python test/test.py
index 0 item 1
index 1 item 2
index 2 item 8
index 3 item 79

(web) C:\Users\espen\virtualEnvs\web>_
```

```python
NUM_LIST = [1, 2, 8, 79]

def test_enumerator(numbr_list):
    """Test enumerate"""
    for i, item in enumerate(numbr_list):
        print("index " + str(i) + " item " + str(item))

test_enumerator(NUM_LIST)
```

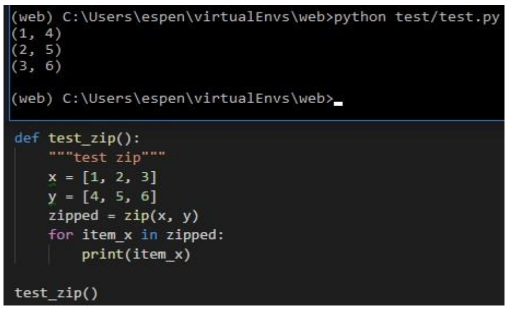
Range

- 

```
(web) C:\Users\espen\virtualEnvs\web>python test/test.py
0
1
2
3
4
5

(web) C:\Users\espen\virtualEnvs\web>_
```

```python
def test_range():
    """Test range"""
    for item_x in range(6):
        print(item_x)

test_range()
```

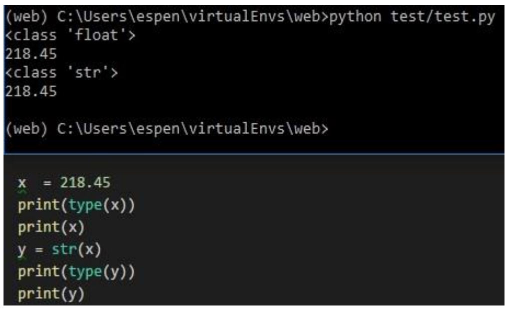
Zip

- 

```
(web) C:\Users\espen\virtualEnvs\web>python test/test.py
(1, 4)
(2, 5)
(3, 6)

(web) C:\Users\espen\virtualEnvs\web>_
```

```python
def test_zip():
    """test zip"""
    x = [1, 2, 3]
    y = [4, 5, 6]
    zipped = zip(x, y)
    for item_x in zipped:
        print(item_x)

test_zip()
```

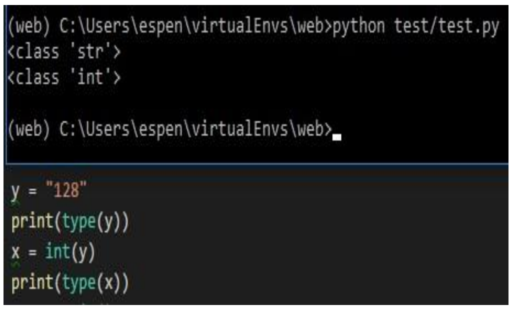
Str

- 

```
(web) C:\Users\espen\virtualEnvs\web>python test/test.py
<class 'float'>
218.45
<class 'str'>
218.45

(web) C:\Users\espen\virtualEnvs\web>
```

```python
x  = 218.45
print(type(x))
print(x)
y = str(x)
print(type(y))
print(y)
```

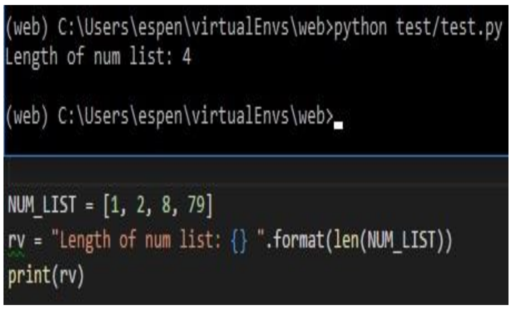
Convert

- 

```
(web) C:\Users\espen\virtualEnvs\web>python test/test.py
<class 'str'>
<class 'int'>

(web) C:\Users\espen\virtualEnvs\web>
```

```
y = "128"
print(type(y))
x = int(y)
print(type(x))
```

Len + format

- 

```
(web) C:\Users\espen\virtualEnvs\web>python test/test.py
Length of num list: 4

(web) C:\Users\espen\virtualEnvs\web>
```

```
NUM_LIST = [1, 2, 8, 79]
rv = "Length of num list: {} ".format(len(NUM_LIST))
print(rv)
```

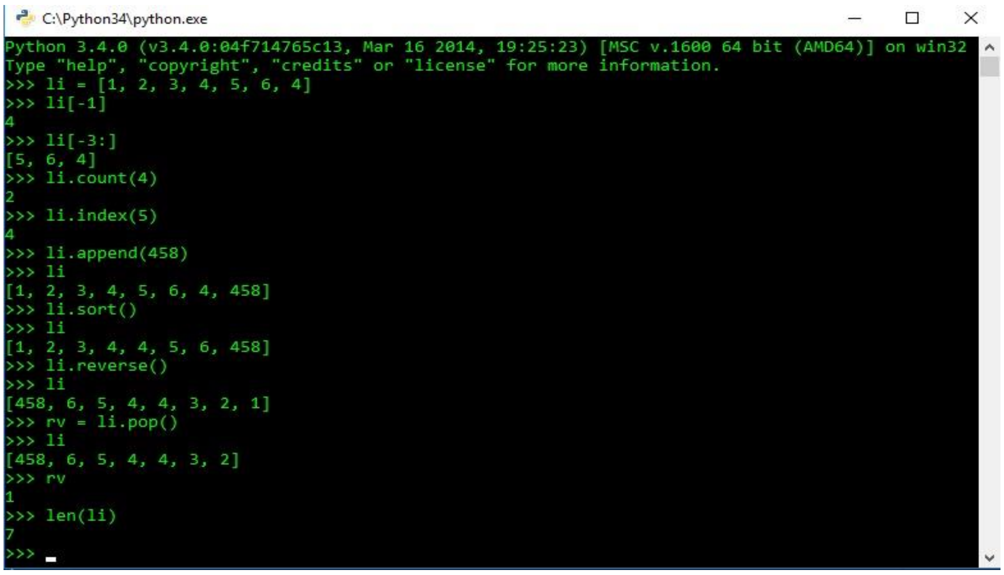
Data structures

https://docs.python.org/3.5/tutorial/index.html

More on list:

A list of comma-separated values (items) between square brackets, lists are mutable

- Like strings (and all other built-in sequence type), lists can be indexed and sliced
- list.append(x)
- list.extend(iterable)
- list.insert(i, x)
- list.remove(x)
- list.pop([i])
- list.clear()
- list.index(x[, start[, end]])
- list.count(x)
- list.sort(key=None, reverse=False)
- list.reverse()
- list.copy()

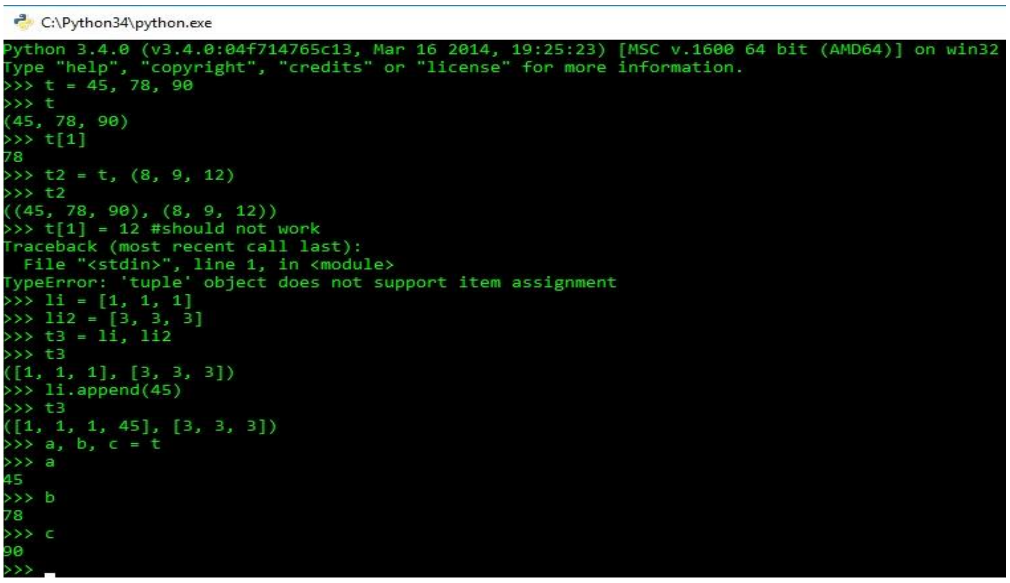


```
Python 3.4.0 (v3.4.0:04f714765c13, Mar 16 2014, 19:25:23) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> li = [1, 2, 3, 4, 5, 6, 4]
>>> li[-1]
4
>>> li[-3:]
[5, 6, 4]
>>> li.count(4)
2
>>> li.index(5)
4
>>> li.append(458)
>>> li
[1, 2, 3, 4, 5, 6, 4, 458]
>>> li.sort()
>>> li
[1, 2, 3, 4, 4, 5, 6, 458]
>>> li.reverse()
>>> li
[458, 6, 5, 4, 4, 3, 2, 1]
>>> rv = li.pop()
>>> li
[458, 6, 5, 4, 4, 3, 2]
>>> rv
1
>>> len(li)
7
>>> _
```

Tuples and Sequences:

A tuple consists of a number of values separated by commas, Tuples are immutable, and usually contain a heterogeneous sequence of elements that are accessed via unpacking

- Index
- Tuples may be nested
- TypeError: 'tuple' object does not support item assignment
- Can contain mutable objects
- t = 45, 78, 90, is an example of tuple packing
- The reverse operation is also possible, a, b, c = t



```
Python 3.4.0 (v3.4.0:04f714765c13, Mar 16 2014, 19:25:23) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> t = 45, 78, 90
>>> t
(45, 78, 90)
>>> t[1]
78
>>> t2 = t, (8, 9, 12)
>>> t2
((45, 78, 90), (8, 9, 12))
>>> t[1] = 12 #should not work
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> li = [1, 1, 1]
>>> li2 = [3, 3, 3]
>>> t3 = li, li2
>>> t3
([1, 1, 1], [3, 3, 3])
>>> li.append(45)
>>> t3
([1, 1, 1, 45], [3, 3, 3])
>>> a, b, c = t
>>> a
45
>>> b
78
>>> c
90
>>> _
```

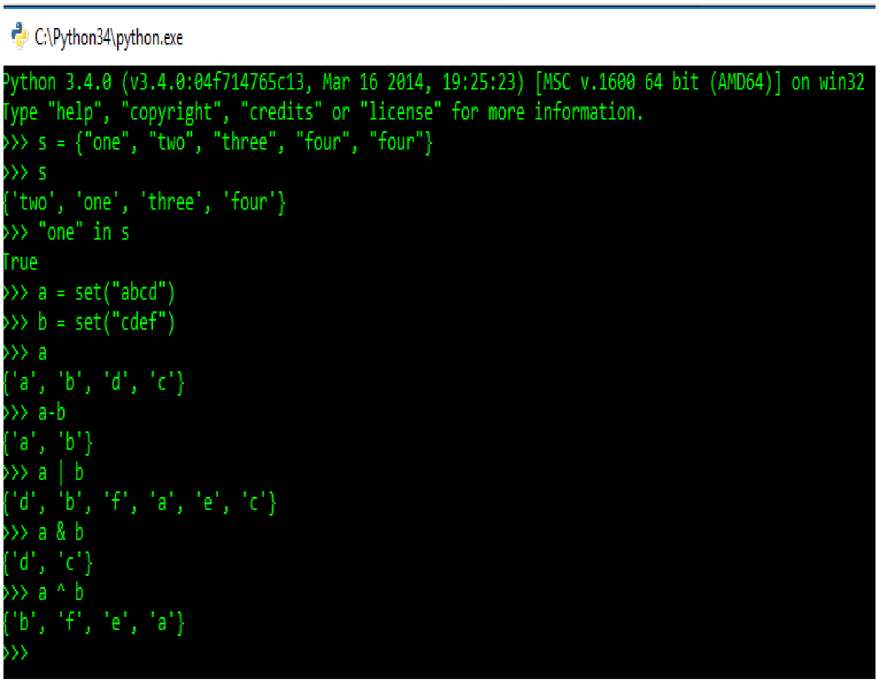
Sets

A set is an unordered collection with no duplicate elements

Curly braces or the set() function can be used to create sets. Note: to create an empty set you have to use set(), not {}; the latter creates an empty dictionary

Set objects also support mathematical operations like union, intersection, difference, and symmetric difference.

- Duplicates will be removed
- Membership testing
- Set operations:
  - , in a but not b
- | , in a or b
- & , in a and b
- ^ , in a or b, but not both



```
Python 3.4.0 (v3.4.0:04f714765c13, Mar 16 2014, 19:25:23) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> s = {"one", "two", "three", "four", "four"}
>>> s
{'two', 'one', 'three', 'four'}
>>> "one" in s
True
>>> a = set("abcd")
>>> b = set("cdef")
>>> a
{'a', 'b', 'd', 'c'}
>>> a-b
{'a', 'b'}
>>> a | b
{'d', 'b', 'f', 'a', 'e', 'c'}
>>> a & b
{'d', 'c'}
>>> a ^ b
{'b', 'f', 'e', 'a'}
>>>
```

Dictionaries

Unlike sequences, which are indexed by a range of numbers, dictionaries are indexed by keys, which can be any immutable type; strings and numbers can always be keys

Dictionaries can be seen as an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}. Placing a comma-separated list of key:value pairs within the braces adds initial key:value pairs to the dictionary