Python Programming Language
Data Structure

Sachin

PVPPCOE

December 22, 2015

Data types

- Basic objects: numbers(float, int, complex), strings, Tuples, lists, sets, & dictionaries
- Other data types: Modules, Class, Instance, Function, Method etc.
- Types
    - Mutable
        - list, dictionaries, instances, sets
    - Immutable
        - Tuples, numbers, strings, None

Number

```
n = 1   # int
n = 3.14565   # float
n = 2.4343e4   # float
```

Complex

```python
n = 1 + 2j  # complex
n.real  # 1.0
n.imag  # 2.0
b = 3.4j  # Just define an imaginary part
b.real  # 0.0
b.imag  # 3.4
n + b  # (1+5.4j)
```

String

`str` type can be defined in variety of ways:

Single quotes

```
'Desolation of the "Smug"'
```

Double quotes

```
"Desolation of the 'Smug'"
```

Triple quotes

```
'''Desolation of the Smug'''
"""Desolation of the Smug"""
```

String: additional methods

```
'hobbit desolation of the smug'.split()
# ['hobbit', 'desolation', 'of', 'the', 'smug']
'hobbit\ndesolation of the smug'.splitlines()
# ['hobbit', 'desolation of the smug']
'hobbit desolation of the smug'.title()
# 'Hobbit Desolation Of The Smug'
```

List

- Collection can be homogeneous

```
l = ['a', 5, "ford", ['apple', 'mango']]
```

- Mutable sequence

```
l.append('b')
# ['a', 5, 'ford', ['apple', 'mango'], 'b']
```

- Methods..
  - `.count()`
  - `.extend()`
  - `.index()`
  - `.pop()`
  - `.reverse()`
  - `.sort()`

List

```
a = [1, 2, 3]
b = [4, 5, 6]
a + b  # [1, 2, 3, 4, 5, 6]
a.append(b)  # ?
a[2]  # 3
a[3]  # [4, 5, 6]
```

List

- `.append` & `.pop` behaves like stack(*last in, first out*)
- What about *first in, first out* ?

List: as *collections*

```python
from collections import deque
q = deque(['Jerry'])
q.extendleft(['Tom'])
q.popleft()
```

List: *Slicing*

```
p = ['p', 'y', 't', 'h', 'o', 'n']
```

List: *Slicing*

```
+-----+-----+-----+-----+-----+-----+
|  p  |  y  |  t  |  h  |  o  |  n  |
+-----+-----+-----+-----+-----+-----+
   0     1     2     3     4     5
  -6    -5    -4    -3    -2    -1
```

List: *Slicing*

```
p[0]  # 'p'
p[5]  # 'n'
p[-1]  # 'n'
p[]  # SyntaxError
p[:]  # ['p', 'y', 't', 'h', 'o', 'n']
```

syntax

```
p[start:end:step]  # 'start', 'end' are index
```

List: *Slicing*

```
p[0:5:1] == p[:] == p[::] == p[::1] == p
p[::2]  # ['p', 't', 'o']
p[::3]  # ['p', 'h']
p[::5]  # ?
p[::6]  # ?
```

List: *Slicing*

```
p[::-1] == p.reverse()
p[:4:]  # ?
```

List comprehension

Create a list of numbers from 1 to 10

```python
for i in range(1, 11):
    print(i)
```

List comprehension

Create a list of numbers from 1 to 10

```
num = []
for i in range(1, 11):
    num.append(i)
print(num)  # [1, 2, .. 10]
```

List comprehension

Create a list of numbers from 1 to 10

```
num = [for i in range(1, 11)]
# [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

List comprehension

How about list of even number?

List comprehension

How about list of even number?

```
[i for i in range(1, 11) if i%2 == 0]
# [2, 4, 6, 8, 10]
```

Nested list comprehension

```
[[i for n in range(1)] for i in range(1, 11) \
 if i%2 == 0]
# ?
```

Nested list comprehension

```
[[i for n in range(1)] for i in range(1, 11) \
 if i%2 == 0]
# [[2], [4], [6], [8], [10]]
```

Tuple

- Immutable sequence
- No methods like `.insert()`, `.append()` etc.
- but slicing works..

Tuple

```
t = ("a", "b", "mpilgrim", "z", "example")
t[0]
t[-1]
t[-3]
```

Set

- Unordered unique elements

Set

```python
apple = ['a', 'p', 'p', 'l', 'e']
apple = set(apple)
apple
# set(['a', 'p', 'e', 'l'])
mango = ['m', 'a', 'n', 'g', 'o']
mango = set(mango)
mango
# set(['a', 'm', 'o', 'g', 'n'])
```

Set

```
apple & mango
# set(['a'])
apple | mango
# set(['a', 'e', 'g', 'm', 'l', 'o', 'n', 'p'])
apple - mango
# set(['p', 'e', 'l'])
apple ^ mango
# set(['e', 'g', 'm', 'l', 'o', 'n', 'p'])
```

Dictionary

```python
empty_dict = {} # Empty dictionary
status = {
    'stdout': 'Hello',
    'stderr': None,
    'exit': 0,
}

print status['exit'] # 0
print status['stdout'] # 'Hello'

print status.keys() # ['stdout', 'stderr', 'exit']
print status.values() # ['Hello', None, 0]
```

Dictionary

```python
status = {
    'stdout': 'Hello',
    'stderr': None,
    'exit': 0,
}

# Change value
status['exit'] = 1 # 0 to 1
```

Dictionary

Delete key

```
d = {'one': 1}
d['two'] = 2
d
# {'two': 2, 'one': 1}
del d['one']
d
# {'two': 2}
```

Dictionary

Clear

```
d.clear()
```

Dictionary

run for-loop over a dictionary

```python
numbers = {
    'one': 1,
    'two': 2,
    'three': 3,
    'four': 4
}

for k, v in numbers.items():
    print k,v

# four 4
# three 3
# two 2
# one 1
```

print

```
print("Sum is", 23+6)
print("sum is %d" % (1+2))
print("Class objects support %s & %s" % \
      ("attribute references", "instantiation."))
```

format

This is a string method

```
print("{} + {} = {}".format(2, 3, 2+3))
print("{0} + {1} = {2}".format(2, 3, 2+3))
print("{1} + {0} = {2}".format(2, 3, 2+3))
# 2 + 3 = 5
# 2 + 3 = 5
# 3 + 2 = 5
```

Operators

```
>>> 5 + 3.4
>>> 5 - 3.4
>>> 5 * 3.4
>>> 5 / 3.4
>>> 5 // 3.4
>>> int(5 / 3.4)
>>> float(89)
>>> abs(3.4 - 5)
>>> divmod(5, 3.4)  # //, %
>>> pow(2,3)  # 2**3
```

Operators

```
# additional operators for int and float
>>> import math

>>> math.trunc(2.3)  # nearest integral toward 0
>>> round(3.1423, 2)  # 3.14
>>> math.floor(2.3)  # 2
>>> math.ceil(3.14)  # 4
```

Operations

Operations	Meaning
<	less than
<=	less than equal to
>	greater than
>=	greater than equal to
==	equal to
!=	not equal to
is	is True, is None
is not	is not None ..

if

```python
if True:
        # do this..
elif condition:
        # or do this..
else:
        # else do this
```

for

```python
for n in seq:
    print(n)
```

for

```python
# example
for i in [1, 2, 3, 4, 5]:
    print(i << 1)
```

while

```python
is_fighter = True
while is_fighter:
    # fight a match
```

Exercise 1

`a = [1, 2, 3, 4..10]`

Generate a new list from 'a' such that if element in 'a' is even, return square else return cube

like:

```
[1, 4, 27, 16, 125, 36, 343, 64, 729, 100]
```

Solution

1.
```
[x*x if x%2==0 else x*x*x for x in range(1,11)]
```

End

Email

`iclcoolster@gmail.com`

Blog

`http://psachin.github.io`