Coding Interview Python Language Essentials

---

1. Hash - backed maps

```
#Define Dictionary
thisdict = {
  'bob': 7387,
   'alice': 3719,
   'jack': 7052,
}

#Get value by key
x = thisdict["bob"]

#Set value by key
thisdict["alice"] = 2456

#Print all keys
for x in thisdict:
  print(x)

#Print all values
for x in thisdict:
  print(thisdict[x])
```

2. Queue

```
from queue import Queue

# Initializing a queue
q = Queue(maxsize = 3)

# qsize() give the maxsize of the Queue
q.qsize()

# Adding of element to queue
q.put('a')

# Return Boolean for Full Queue
q.full()

# Removing element from queue
q.get()

# Return Boolean for Empty Queue
q.empty()
```

3. Stack

```
# Approach 1
stack = [3, 4, 5]
stack.append(6)     # [3 ,4, 5, 6]
stack.pop()         # [3 , 4, 5]

# Approach 2
class Stack:
  def __init__(self):
    self.stack = []

    # check if empty
  def isEmpty(self):
    return len(self.stack) == 0

  def push(self,p):
    self.stack.append(p)

  def pop(self):
    return self.stack.pop()
```

4. Exceptions

```
try:
   fh = open("testfile", "r")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
#No exception run this code
else:
   print "Written content in the file successfully"
finally:
   print "Error: can\'t find file or read data"

# Raise an exception
x = 10
if x > 5:
    raise Exception('x should not exceed 5. The value of x was: {}'.format(x))

#Assert an error
import sys
assert ('linux' in sys.platform), "This code runs on Linux only."
```

5. String

> Python strings are "immutable" which means they cannot be changed after they are created. Since strings can't be changed, we construct *new* strings as we go to represent computed values. So for example the expression ('hello' + 'there') takes in the 2 strings 'hello' and 'there' and builds a new string 'hellothere'.

```
s = 'hi'
print s[1]          ## i
print len(s)        ## 2
print s + ' there'  ## hi there
```

6. Casting

```
# string to int
int("12")

#int to string
str(number_to_convert)
```

7. Arithmetic

```
# Modulus
5 % 2    ## return 1

# Division
5 / 2    ## return 2.5

# Division round off
5 // 2   ## returns 2

# Round examples
round(51.6) ## return 52
round(51.5) ## return 52
round(51.4) ## return 51
round(2.665, 2) ## return 2.67
round(2.676, 2) ## return 2.68

# Floor and Ceil
import math

math.floor(300.16) ## return 300
math.floor(300.76) ## return 300

math.ceil(300.16) ## return 301
math.ceil(300.16) ## return 301
```

8. 2-D Array

```
# Approach 1
matrix = []
for i in range(rows):
  row = []
  for j in range(cols):
    row.append(0)
  matrix.append(row)

# Approach 2
matrix = [[0 for i in range(5)] for j in range(5)]
```

9. Sorting

```
# Approach 1
sorted([5, 2, 3, 1, 4])

# Approach 2
a = [5, 2, 3, 1, 4]
a.sort()
```

10. Switch

```
def numbers_to_strings(argument):
    switcher = {
        0: "zero",
        1: "one",
        2: "two",
    }

    # get() method of dictionary data type returns
    # value of passed argument if it is present
    # in dictionary otherwise second argument will
    # be assigned as default value of passed argument
    return switcher.get(argument, "nothing")

# Driver program
if __name__ == "__main__":
    argument=0
    print numbers_to_strings(argument)
```

11. Array Enumerate (with index)

```
# iterate over array
ints = ["a", "b", "c"]

for idx, val in enumerate(ints):
    print(idx, val)
```

12. Bit Manipulation

```
a = 60
b = 13

# AND
c = a & b       ## c = 12

# OR
c = a | b       ## c = 61

# Binary XOR
c = a ^ b       ## c = 49

# Binary ones complement / NOT
c = ~a          ## c = -61

# Binary Left Shift
c = a << 2      ## c = 240

# Binary Right Shift
c = a >> 2      ## c = 15
```