**Python Cheat Sheet**

**Reading input from command prompt:**

* **Read integer from prompt :** int(input())
* **Read space separated values from prompt:** input().split()
* **Intilizating or reading in nested lists :**

# Creates a list containing 5 lists, each of 8 items, all set to 0

w, h = 8, 5.

Matrix = [[0 for x in range(w)] for y in range(h)]

* **Creating 2d list with 2 diemsion:**

marksheet = []

for \_ in range(0,int(input())):

marksheet.append([input(), float(input())])

for i in range(t):

k = int(sys.stdin.readline().strip())

A = [int(x) for x in sys.stdin.readline().strip().split()]

**List Comprehensions:**

* If/Else condition in List comprehension ([**http://stackoverflow.com/questions/4406389/if-else-in-a-list-comprehension**](http://stackoverflow.com/questions/4406389/if-else-in-a-list-comprehension))

>>> l = [22, 13, 45, 50, 98, 69, 43, 44, 1]

>>> [x+1 if x >= 45 else x+5 for x in l]

[27, 18, 46, 51, 99, 70, 48, 49, 6]

**Conversions:**

**From List :**

* **From List to String:**

list1 = ['1', '2', '3']

str1 = ''.join(list1)

**Misc Fucntions**

Map function :

Yield :

Set: Sets are lists with no duplicate entries

[**http://www.learnpython.org/en/Sets**](http://www.learnpython.org/en/Sets)

**Data Structures:**

**Q. Alogrithm to find GCD?**

1. Euclid’s Algorithm : This algorithm helps us to find the greatest common divisor for 2 integers.

<https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/the-euclidean-algorithm>

**Q. What is self in python?**

1. Self is always a reference to the current instance (Object) of the class

**Q. How can we compute execution time in python?**

A. Use python timeit module to measure computation time

List big O Efficiency code (<https://www.ics.uci.edu/~pattis/ICS-33/lectures/complexitypython.txt>)

from timeit import Timer

from random import \*

for i in range(1000,1000001,1000):

x=list(range(i))

t1 = Timer("x.sort()","from \_\_main\_\_ import x")

print ("Number of iteration: %d"%i,t1.timeit(number = 1000))

**Q. Calculate time complexity of List index operation in Python?**

A, The time increases as the i value increases because this code uses the list.index(x) method instead of the list index operator list[x].   
  
The list.index(x) method searches linearly through a list until it finds the item x, then it returns the index of that item. If list.index(x) doesn't find an item x, then it returns an error.   
  
The list index operator list[x] returns an item at index x of the list, and runs in constant time.   
  
Below is a solution that yields time complexity of O(1):

for i in range(10000, 1000000, 20000):

    t = timeit.Timer("x[(random.randrange(%d))]"%i, "from \_\_main\_\_ import random, x")   
  
x = list(range(i))

index\_time = t.timeit(number=1000)   
print("%d, %10.4f" % (i, index\_time))

**Q. Devise an experiment to verify that get item and set item are O(1) for dictionaries.**

A. from timeit import Timer

from random import \*

for i in range(1000,1000001,1000):

dic = {i:None for i in range(i+1)}

x = Timer("dic[randint(1,%d)]=randint(1,10)"%i,"from \_\_main\_\_ import randint,dic")

indtime = x.timeit(number = 1000)

print i,"-",indtime

**Q. Compariason of Dict and list del operation efficiency**

A. from timeit import Timer

from random import \*

for i in range(1000,1001,1000):

d = [dict.fromkeys(range(i)) for \_ in range(1000)]

l = [list(range(i+2)) for i in range(1000)]

liter = iter(l)

diter = iter(d)

x1 = Timer("del next(diter)[10]","from \_\_main\_\_ import randint,diter")

x2 = Timer("del next(liter)[0]","from \_\_main\_\_ import randint,liter")

lsttime = x2.timeit(number = 1000)

indtime = x1.timeit(number = 1000)

print i,"-",lsttime,"-",indtime

#x1 = Timer("del dic[0]"%i,"from \_\_main\_\_ im#port randint,dic")

#x2 = Timer("del l[randint(%d,%d)]"%i,"from \_\_main\_\_ import randint,l")

#indtime = x1.timeit(number = 1)

#lsttime = x2.timeit(number = 1)

#print i,"-",indtime,"-",lsttime

**Q. Big O Notion references**

A. <http://stackoverflow.com/questions/487258/what-is-a-plain-english-explanation-of-big-o-notation>