Cheatography

Coding (Python) Cheat Sheet

by bigocb via cheatography.com/97523/cs/20901/

Loops

FOR

```
list = ["geeks", "for", "geeks"]
for index in range(len(list)):
    print list[index]
list = ["geeks", "for", "geeks"]
for index in range(len(list)):
    print list[index]
else:
    print "Inside Else Block"
```

WHILE

```
while (count < 3):
    count = count + 1
    print("Hello Geek")
```

Strings

Count number of occurrences of x in str

Use counter to create a collection of letters. Pull number where str matches. Also useful in anagram

```
(Py) count = collections.Counter(str)
```

Reverse str

Useful in many problems including valid palindrome and plain reverse string

```
(Py) revs = str[::-1]
(Py) revs = str.reverse
```

Find first unique character

First create a collection of letters. THEN use enumerate() in for loop to find index of character with first 1

```
(Py) count = collections.Counter(str)
(Py) for idx,i in enumerate(str)
```

Lists

```
List = []
---------------------------------
# Addition of Elements
# in the List
```

Lists (cont)

```
---------------------------------
List.append(1)
List.append(2)
List.append(4)
for i in range(1, 4):
    List.append(i)
List2 = ['For', 'Geeks']
List.append(List2)
---------------------------------
# accessing a element from the
# list using index number
---------------------------------
print(List[0])
print(List[2])
---------------------------------
# Removing elements from List
# using Remove() method
---------------------------------
for i in range(1, 5):
    List.remove(i)
---------------------------------
# Removing element at a
# specific location from the
# Set using the pop() method
---------------------------------
List.pop(2)
---------------------------------
# Print elements from a
# pre-defined point to end
---------------------------------
Sliced_List = List[5:]
```

Intersection

```
class Solution(object):
    def intersect(self, nums1, nums2):
        from collections import Counter
        c1 = Counter(nums1)
        c2 = Counter(nums2)
        return list((c1&c2).elements())
```

Valid Anagram

```
class Solution(object):
    def isAnagram(self, s, t):
        sColl = collections.Counter(s)
        tColl = collections.Counter(t)
        if (sColl == tColl):
            return True
        else:
            return False
```

Valid Palendrone

```
class Solution(object):
    def isPalindrome(self, s):
        cleans = re.sub(r'[^\w\s]','',s).replace("
","").lower()
        revs = cleans[::-1]
        cleanr = re.sub(r'[^\w\s]','',revs).rep-
lace(" ","").lower()
        if(cleanr == cleans):
            return True
        else:
            return False
```

Max depth of Tree

```
class Solution:
    def maxDepth(self, root: TreeNode) -> int:
        if root == None:
            return 0

        left = self.maxDepth(root.left)
        right = self.maxDepth(root.right)
```

Max depth of Tree (cont)

```
        return 1 + max(left, right)
```

Shuffle an Array

```
class Solution:
    def __init__(self, nums):
        self.array = nums
        self.original = list(nums)
    def reset(self):
        self.array = self.original
        self.original = list(self.original)
        return self.array
    def shuffle(self):
        aux = list(self.array)
        for idx in range(len(self.array)):
            remove_idx = random.randrange(len(-
aux))
            self.array[idx] = aux.pop(remove_idx)
        return self.array
```

Shuffle an Array

```
class Solution:
    def __init__(self, nums):
        self.array = nums
        self.original = list(nums)
    def reset(self):
        self.array = self.original
        self.original = list(self.original)
        return self.array
    def shuffle(self):
        aux = list(self.array)
        for idx in range(len(self.array)):
            remove_idx = random.randrange(len(-
aux))
            self.array[idx] = aux.pop(remove_idx)
        return self.array
```

Cheatography

Coding (Python) Cheat Sheet

by bigocb via cheatography.com/97523/cs/20901/

Dicts

```
dict = {'Spammer' : 1}
# Deleting a key-value pair
del dict['Spammer']
# Adding a key-value pair
ab['Guido'] = 'guido@python.org'
```

import from file

```
with open('rules.json') as file:
   data = json.load(file)
```

Write to file

```
f= open("guru99.txt","w+")
f.write("This is line")
f.close
```

Linked List

```
class Node:
    def __init__(self,val):
        self.val = val
        self.next = None
        self.prev = None
    def traverse(self):
        node = self
        while node != None:
            print node.val
            node = node.next
n1 = Node(12)
n2 = Node(3)
n1.next = n2
n2.prev = n1
```

Remove Dupes from Array

```
def removeDuplicates(self, nums):
     for x in range(len(nums)-1,0,-1):
            if nums[x] == nums[x-1]:
                nums.remove(nums[x])
        print nums
```

Best time to Buy and Sell

```
class Solution(object):
    def maxProfit(self, prices):

        n = len(prices)-1
        if n <= 1:
            print 0
        maxprofit = 0

        for i in range(0, n):
            t = prices[i]
            m = prices[i+1]
            if t < m:
                maxprofit += m-t
                i++1
        return maxprofit
```

Contains Dupe

```
class Solution(object):
    def containsDuplicate(self, nums):
        nSorted = sorted(nums)
        ans = False
        l = len(nSorted)
        if len(nSorted) > 1:
            for i in range(0,len(nums)):
                if nSorted[i] == nSorted[i-1]:
                    ans = True
        return ans
```

Find non dup number

```
class Solution(object):
    def singleNumber(self, nums):

        nSort = sorted(nums)

        if(len(nums) > 1):
            for i in range(0,len(nSort)):
                if nSort[i] == nSort[i-1]:
                    nums.remove(nSort[i])
                    nums.remove(nSort[i-1])
        return nums[0]
```

Tree

```
# Definition for a binary tree node.
class TreeNode(object):
     def __init__(self, x):
         self.val = x
         self.left = None
         self.right = None
```

Shuffle an Array

```
class Solution:
    def __init__(self, nums):
        self.array = nums
        self.original = list(nums)
    def reset(self):
        self.array = self.original
        self.original = list(self.original)
        return self.array
    def shuffle(self):
        aux = list(self.array)
        for idx in range(len(self.array)):
            remove_idx = random.randrange(len(-
aux))
            self.array[idx] = aux.pop(remove_idx)
        return self.array
```

Merge Sorted Arrays

```
class Solution(object):
    def merge(self, nums1, m, nums2, n):
        n1len = len(nums1)
        n2len = len(nums2)
        if n1len > n2len:
            while n1len != m:
                t = nums1[m]
                nums1.remove(t)
                n1len -=1

        elif n2len > n1len:
            while n2len != m:
                t = nums2[m]
                nums2.remove(t)
                n2len -=1
        num1j = nums1 + nums2
        nums1 = sorted(num1j)

        return nums1
```