

1 Python CheatSheet

LANGUAGES

- PDF Link: [cheatsheet-python-A4.pdf](#), Category: languages
- Blog URL: <https://cheatsheet.dennyzhang.com/cheatsheet-python-A4>

File me Issues or star this repo.

See more CheatSheets from Denny: [#denny-cheatsheets](#)

1.1 Python Compact Coding

Name	Comment
if ... return	if k == 0: return False
if... continue	if index == icol: continue
return if.. else	return val if i>0 else 0
multiple assignment	l, r = 2, 3
assign with check of none	a = b if b else 1
assignments	l[1]=l[0]=0
swap values	left, right = right, left
list Comprehensions	[x*x for x in range(1, 1001)]
list Comprehensions	l = [2, 3, 5]; [2*x for x in l if x>2]
use zip	for a, b in zip(nums, nums[3:])
build a list	dp = [1] + [0]*3
sum a subarray	sum(nums[0:k])
sort list in descending order	sorted(nums, reverse=True)
dictionary with defaults	m = collections.defaultdict(lambda: 1)
loop with single statement	while p.left: p = p.left
print multiple values	print(x, y)
get both index and item	for i, ch in enumerate(["a", "b", "c"]): print(i, ch)
mod negative	(-2)%5

1.2 Python Common Algorithms

Name	Comment
bfs	code/tree-bfs.py
trie tree	code/tree-trie.py

1.3 List

Name	Comment
return all but last	list[:-1]
The second last item	list[-2] or list[~1]
map	map(lambda x: str(x), [1, 2, 3])
create fixed size array	l = [None] * 5
insert elements to head	array.insert(0,var)
delete element by index	del a[1]
list as stack	item = l.pop()
sort in descending	l = sorted([8, 2, 5], reverse=True)
sort by attribute	l=sorted([('ebb',12),('abc',14)], key=lambda x: x[1])
in-place sort	l.sort()
generate a-z	map(chr, range(ord('a'), ord('z')+1))
map/reduce	functools.reduce((lambda x, y: "%s %s" % (x, y)), l)
replace ith to jth	list[i:j] = otherlist
combine two list	list1 + list2
get sum	sum(list)
unique list	set(["Blah", "foo", "foo", 1, 1, 2, 3])
Insert to sorted list	bisect.insort(l, 3)
Reverse a list	l[::-1]

1.4 String

Name	Comment
reverse string	<code>'hello world'[::-1]</code>
array to string	<code>' '.join(['a', 'b'])</code>
split string to array	<code>"hello, python".split(",")</code>
string to array	<code>list('abc')</code>
format to 2 digits	<code>print "%02d" % (13)</code>
find location of substring	<code>'abc'.find('d')</code> = (returns -1)
find location of substring	<code>'abc'.index('d')</code> = (raise exception)
capitalize string	<code>'hello world'.capitalize()</code>
upper/lower string	<code>'aBc'.upper()</code> =, <code>'aBc'.lower()</code>
count substring	<code>'2-5g-3-J'.count('-')</code>
replace string	<code>'ab cd'.replace(' ', '')</code>
padd whitespace to the left	<code>'a'.ljust(10, ' ')</code>
padd whitespace to the right	<code>'a'.rjust(10, ' ')</code>
pad leading zero	<code>'101'.zfill(10)</code>
string remove tailing '0'	<code>'0023'.rstrip('0')</code>
string remove leading '0'	<code>'0023'.lstrip('0')</code>
check if string represent integer	<code>'123'.isdigit()</code>
check if string alphabetic	<code>'aBc'.isalpha()</code>
Check if string alphanumeric	<code>'a1b'.isalnum()</code>

1.5 Integer

Name	Comment
max, min	<code>sys.maxsize, -sys.maxsize-1</code>
min, max	<code>min(2, 3), max(5, 6, 2)</code>
generate range	<code>for num in range(10,20)</code>
get ascii	<code>ord('a'), chr(97)</code>
print integer in binary	<code>"{0:b}".format(10)</code>

1.6 Dict & Set

Name	Comment
dict get first element	<code>m[m.keys()[0]]</code>
intersection	<code>list(set(11).intersection(set(12)))</code>
list to set	<code>set(list1)</code>
remove from set	<code>s.remove(2)</code>
remove the first from set	<code>s.pop()</code>
sort dict by values	<code>sorted(dict1, key=dict1.get)</code>
deep copy dict	<code>import copy; m2=copy.deepcopy(m1)</code>

1.7 Bit Operator

Name	Comment
mod	<code>x % 2</code>
shift left	<code>x << 1 ; a << 2=</code>
shift righ	<code>x >> 2</code>
and	<code>x & y</code>
complement	<code>~x</code>
xor	<code>x ^ y</code>
power	<code>2 ** 3</code>
bool complement	<code>not x</code>
binary format	<code>bin(5)</code> (get 101)
count 1 inside binary	<code>bin(5).count('1')</code>

1.8 File

Name	Comment
Append file	<code>open("/tmp/test.txt", "ab").write("\ntest:")</code>
Write file	<code>open("/tmp/test.txt", "wab").write("\ntest:")</code>
Read files	<code>f.readlines()</code>
Check file	<code>os.path.exists("/tmp/test.txt")</code>

1.9 Math

Name	Comment
<code>sqrt</code>	<code>import math; math.sqrt(5)</code>
<code>power</code>	<code>import math; math.pow(2, 3)</code>
<code>random</code>	<code>random.randint(1, 10)</code> 1 and 10 included
<code>eval string</code>	<code>eval("2-11*2")</code>

1.10 Queue/heapq

Name	Comment
Initialize min heap	<code>heapq.heapify(q)</code>
heappush a tuple	<code>q[]; heapq.heappush(q, (5, 'ab'))=</code>
pop	<code>print (heapq.heappop(q))</code>
first item	<code>q[0]</code>
print heapq	<code>print list(q)</code>
create a queue	<code>from collections import deque; queue = deque([1,5,8,9])</code>
append queue	<code>queue.append(7)</code>
pop queue from head	<code>element = queue.popleft()</code>

Review: Heap Problems

Link: [BINARY HEAP AND HEAPQ IN PYTHON](#)

1.10.1 minheap & maxheap

```
import heapq

# initializing list
li = [5, 7, 9, 1, 3]

# using heapify to convert list into heap
heapq.heapify(li) # a minheap
heapq._heapify_max(li) # for a maxheap!

# printing created heap
print (list(li))

# using heappush() to push elements into heap
# pushes 4
heapq.heappush(li,4)

# printing modified heap
print (list(li))

# using heappop() to pop smallest element
print (heapq.heappop(li))

print (list(li))
```

1.11 Code snippets

- Initialize Linkedlist from array

```
def initListNodeFromArray(self, nums):
    head = ListNode(None)
    prev, p = head, head
    for num in nums:
        pre = p
        p.val = num
        q = ListNode(None)
        p.next = q
        p = p.next
    pre.next = None
    return head
```

- Print linkedlist

```
def printListNode(self, head):
    print("printListnode")
    while head:
        print("%d" % (head.val))
        head = head.next
```

- Print Trie Tree in level order

```
def printTrieTreeLevelOrder(self, node):
    print("printTrieTreeLevelOrder")
    if node.is_word:
        print("Node is a word")
    queue = []
    queue.append(node)
    while len(queue) != 0:
        s = ''
        for i in range(len(queue)):
            node = queue[0]
            del queue[0]
            for child_key in node.children:
                s = '%s %s' % (s, child_key)
                queue.append(node.children[child_key])
        if s != '':
            print 'print level children: %s' % (s)
```

- python sort with customized cmp function: -1 first

```
nums = [3, 2, 6]
def myCompare(v1, v2):
    return -1
sorted_nums = sorted(nums, cmp=myCompare)
print nums # [3, 2, 6]
print sorted_nums # [6, 3, 2]
```

- Initialize m*n matrix

```
col_count, row_count = 3, 2
matrix = [[None for j in range(col_count)] for i in range(row_count)]
print matrix
```

1.12 More Resources

License: Code is licensed under MIT License.