1 Python CheatSheet   LANGUAGES

- PDF Link: cheatsheet-python-A4.pdf, Category: languages

- Blog URL: `https://cheatsheet.dennyzhang.com/cheatsheet-python-A4`

- Related posts: Golang CheatSheet, Ruby CheatSheet, #denny-cheatsheets

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

1.5   Integer

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

1.6   Dict & Set

Name	Comment
dict get first element	`m[m.keys()[0]]`
intersection	`list(set(l1).intersection(set(l2)))`
list to set	`set(list1)`
remove from set	`s.remove(2)`
remove the first from set	`s.pop()`
sort dict by values	`sorted(dict1, key=dict1.get)`
deep copy dict	`import copy; m2=copy.deepcopy(m1)`

1.7   Bit Operator

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

1.8   File

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

1.9   Math

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

1.10   Networking

Name	Comment
Start a simple HTTP server	`python -m SimpleHTTPServer <port_number>`

1.11   Queue/heapq

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

Review: Heap Problems
Link: BINARY HEAP AND HEAPQ IN PYTHON

1.11.1   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))
```

1.12 Code snippets

- Initialize Linkedlist from array

```python
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

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

- Print Trie Tree in level order

```python
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

```python
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

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

1.13 More Resources

License: Code is licensed under MIT License.