Cheatography

Python Coding Interview Cheat Sheet

by Ptero via cheatography.com/190551/cs/39752/

Type built-in

bool(x=False)

bytearray(source=b'')
bytearray(source, encoding, errors)

bytes(source=b'')
bytes(source, encoding, errors)

complex(real=0, imag=0)
complex(string)

dict(**kwarg)
dict(mapping, **kwarg)
dict(iterable, **kwarg)

float(x=0.0)

int(x=0)
int(x, base=10)

set(iterable)

str(object='')
str(object=b'', encoding='utf-8', errors='strict')

tuple(iterable)

Built-in math functions

abs(x)

divmod(a, b)

max(iterable, *, default, key=None)
max(arg1, arg2, *args, key=None)

min(iterable, *, default, key=None)
min(arg1, arg2, *args, key=None)

pow(base, exp, mod=None)

round(number, ndigits=None)

sum(iterable, /, start=0)

String built-in functions

bin(x)

chr(i)

hex(x)

oct(x)

ord(c)

print(*objects, sep=' ', end='\n', file=None, flush=False)

Functional programming

filter(function, iterable)

map(function, iterable, *iterables)

func = lambda a, b, c : a + b + c

Other built-in functions

all(iterable)

any(iterable)

enumerate(iterable, start=0)

sorted(iterable, /, *, key=None, reverse=False)

zip(*iterables, strict=False)

Misc

occurrence	import collections d = defaultdict(int) for w in words: d[w] += 1

Min Max

min int/float	float('-inf') // or -math.inf
max int/float	float('inf') // or math.inf
min	m = min(a, b, c, ...)
min in array	m = min(arr)
min in dict key	m = min({1:4, 2:3}) # 1
dict key w/ min value	d = {1:4, 2:3} m = min(d, key = lamba k : d[k]) # 2

String

new	s = "str"
get	c = s[i]
check empty	if not s:
length	length = len(s)
compare	if s1 == s2:
reverse	r = s[::-1]
to lower	s = s.**lower**()

String (cont)

to upper	s = s.**upper**()
get sub string	sub = s[from : to]
check sub string	if sub in s:
split	arr = s.**split**(sep)
join	s = sep.**join**(arr)

List

new	arr = [1, 2, 3]
get	num = arr[i]
check empty	if not arr:
length	length = len(arr)
reverse	r = arr[::-1]
get sub array	sub = arr[from : to]
fill	arr = [i for i in range(5)]
traverse	for num in arr:
sort asscending (in-place)	arr.**sort**()
sort descending (in-place)	arr.**sort**(reverse=True)
find index	index = arr.**index**(num, start, end)

Stack

new	stack = []
push	stack.**append**(num)
pop	popped = stack.**pop**()
top	top = stack[-1]
check empty	if not stack:
length	length = len(stack)

Queue

new	queue = []
push	queue.**append**(num)
pop	popped = queue.**pop**(0)
front	front = queue[0]
check empty	if not queue:
length	length = len(queue)

By **Ptero**
cheatography.com/ptero/

Not published yet.
Last updated 10th August, 2023.

Heap	
import	import heapq
new	heapq = []
heapify	heapq.**heapify**(heap)
push	heapq.**heappush**(heap, item)
pop	smallest = heapq.**heappop**(heap)
push & pop	smallest = heapq.**heappushpop**(heap, item)
pop & push	smallest = heapq.**heapreplace**(heap, item)
peek	smallest = heap[0]
check empty	if not queue:
length	length = len(queue)

Set	

new	s = set() # for empty set
	# or
	s = {1, 2, 3}

add	s.**add**(item)
add list	s.**update**(arr)
delete	s.**remove**(item) # throw error # or s.**discard**(item) # no error
clear	s.**clear**()
check empty	if not s:
check in set	if item in s:
size	length = len(s)
to list	arr = list(s)
merge	s = s1 **|** s2
intersection	s = s1 **&** s2
difference	s = s1 **-** s2
symetric difference	s = s1 **^** s2

Dict	

new	d = {} # for empty dict
	# or
	d = {1:2, 3:4}

set	d[key] = val

get	val = d[key] # throws error
	# or
	val = d.get(key) # None if not exist

delete	val = s.**pop**(key) # or **del** d[key]
delete last	key, val = s.**popitem**()
clear	**del** d
check empty	if not d:
check key in dict	if key in d.**keys**():
check value in dict	if val in d.**values**():
size	length = len(s)
iterate	for key, val in d.items():

Queue (thread safe)	
import	from queue import SimpleQueue
new	queue = SImpleQueue()
push	queue.**put**(num)
pop	popped = queue.**get**()
check empty	if queue.**empty**():
length	length = queue.**qsize**()

PriorityQueue (thread safe)	
import	from queue import PriorityQueue
new	pq = PriorityQueue()
push	pq.**put**(num)
pop	popped = pq.**get**()
check empty	if pq.**empty**():
length	length = pq.**qsize**()