Cheatography

Python Cheat Sheet

by elhamsh via cheatography.com/31327/cs/9479/

bisect (basic bisection algorithm)

import bisect	Provides support for maintaining a list in sorted order without having to sort the list after each insertion.
a=[0, 2, 3, 8]	
bisect.bisect_left(a, value)	Returns first index that value can be inserted in list
bisect.bisect_right(a, value)	Returns last index that value can be inserted in list
a.insert(index, value)	Index from previous commands
bisect.insort_left(a, value)	Insert value to the correct position
bisect.insort_right(a, value)	
bisect.insort_right(a , value, lo, hi)	For all bisect modules, we can select a sublist [lo:hi]
def grade(score, breakpoints=[60, 70, 80, 90], grades='FDCBA'): i = bisect(breakpoints, score) return grades[i]	
[grade(score) for score in [33, 99, 77, 70, 89, 90, 100]] output: ['F', 'A', 'C', 'C', 'B', 'A', 'A']	

Index might be len(a), if item will be added at the end.

Dict

d = {'a': 1, 'b': 2, 'c':3}/ d=dict()	
d['a']	Raise KeyError exception if 'a' not in dict
d.pop(key)	return value of remove from dic
for key, value in d.items():	iterate
for key in d: print key, d[key]	

Dict (cont)

d.clear()	Removes all items from d
d.items()	Returns a **list** of (key, value) pairs
d.keys()	Returns a list of keys ['a','b','c']
d.values()	Returns a list of values [1,2,3]
d.get(key, default_value)	Returns value of key in d if exists, O.W. return default_value
d.pop(key, default_value)	Return d[key] and remove key from d if exists, else return default_value
sorted(d)	Returns sorted keys

new = {}
for (key, value) in data:
group = new.setdefault (key, [])
group.append(value)
	Key might exist already
	input: data = [(1, "a"), (2, "b")]
	output: new {1: ['a'], 2: ['b']}

from collections import defaultdict	simpler with defaultdict
new = defaultdict(list) for (key, value) in data: new[key].append(value)	

Itertools

import itertools	
for i in itertools.count(5) :	i starts from 5 and go to infinity. Use break to stop.
for i in itertools.cycle([1, 2, 3]):	i iterate over the list indefinitely

Itertools (cont)

for i in itertools.izip([1, 2, 3], ['a', 'b', 'c']): for i in zip([1, 2, 3], ['a', 'b', 'c']):	(1, 'a') (2, 'b') (3, 'c')
for i in enumerate(['a', 'b', 'c']):	(0, 'a') (1, 'b') (2, 'c')
for i in itertools.compress(' ABCDEF', [1,0,1,0,1,1]):	A C E F
for i in itertools.product('AB ', 'CD'):	('A', 'C') ('A', 'D') ('B', 'C') ('B', 'D')
for i in itertools.product('ABC', repeat=2):	('A', 'A') ('A', 'B') ('A', 'C') ('B', 'A') ('B', 'B') ('B', 'C') ('C', 'A') ('C', 'B') ('C', 'C')
for i in itertools.permutations('ABC', 2):	('A', 'B') ('A', 'C') ('B', 'A') ('B', 'C') ('C', 'A') ('C', 'B')
for i in itertools.combinations('ABC', 2):	('A', 'B') ('A', 'C') ('B', 'C')
for i in itertools.combinations_with_replacement('ABC', 2):	('A', 'A') ('A', 'B') ('A', 'C') ('B', 'B') ('B', 'C') ('C', 'C')

String

chr(number)	Return string of one char. chr(65): A
ord(char)	Return int code of the char
len('asd')	3
'asd'.capitalize()	Asd
'asd'.center(width, fill_char)	'asd'.center(6, '$'): $asd$$

String (cont)

'asdas'.count('as')	2
'asdas'.find('a')	0
'asdas'.rfind('a')	3
'asdas'.rsplit(delim, maxsplits)	'asdas'.rsplit('a',1): ['asd', 's']
	'asdas'.split('a',1): ['', 'sdas']
'asdas'.endswith('as')	True
'asdas'.startswith('s')	False
'absddddsssssaaa'.strip('asd')	'b'
' a b '.replace(' ','')	'ab'
list('abc')	['a', 'b', 'c']
sorted('cba')	['a', 'b', 'c']
'ABs'.lower()	'abs'
'ABs'.upper()	'ABS'
'absdf'.index('b')/ 'absdf'.index('b', 0, 3)	1 /set start and end point
'absdfa'.rindex('a')	5
''.join(['a', 'b', 'c'])	'abc'
int('34')	34
float('34.5')	34.5
long('345')	345L
345L==345	True
'34567'.isdigit()	True
'asd'.islower()	True
'ASD'.isupper()	True

Math

divmod(5,3)	1,2
hex(int_number)	hex(15)='0xf'
import math	
math.ceil(4.1)	5.0
math.floor(4.1)	4.0
math.pow(2,3)	8.0

Math (cont)

import random	
random.random()	Return a float in [0.0,1.0)
random.randint(0, 5)	Return a random integer in [0,5]
random.uniform(a,b)	Return a random floating point number N s.t. a<=N<=b for a<=b and b<=N<=a for b<a.
random.choice([" mert", "gunay", "kth"])	Return an item form the sequence
random.choice('abcdefghij')	'c'
random.sample(["mert", "gunay", "kth", "stockholm"], 2)	Return k item: ['gunay', 'kth']
random.shuffle(list)	change the order of items randomly
random.randrange(start, stop, step)	random.randrange(0, 101, 2): Even integer from 0 to 100

Queue

from collections import deque	double ended queue
d=deque('ghi')	deque(['g', 'h', 'i'])
d.append('j')	deque(['g', 'h', 'i', 'j'])
d.appendleft('f')	deque(['f', 'g', 'h', 'i', 'j'])
d.pop()	'j' deque(['f', 'g', 'h', 'i'])
d.popleft()	'f' deque(['g', 'h', 'i'])
d.extend('abc')	deque(['f', 'g', 'h', 'i', 'a', 'b', 'c'])
d.extendleft('abc')	deque([**'c', 'b', 'a'**, 'f', 'g', 'h', 'i', 'a', 'b', 'c'])

Queue (cont)

d.rotate(1)	deque(['c', 'c', 'b', 'a', 'f', 'g', 'h', 'i', 'a', 'b'])
d.rotate(-2)	deque(['b', 'a', 'f', 'g', 'h', 'i', 'a', 'b', 'c', 'c'])
a=deque(reversed(d))	a: deque(['c', 'c', 'b', 'a', 'i', 'h', 'g', 'f', 'a', 'b'])
del a[0]	deque(['c', 'b', 'a', 'i', 'h', 'g', 'f', 'a', 'b'])

Priority Queue (heap)

import heapq	
heap=[]	
heapq.heappush(heap, 2)	heap == [2]
heap=[4,2,1]	
heapq.heapify(heap)	heap = [1,2,4]
heapq.heappush(heap,3)	heap = [1, 2, 4, 3]
heapq.nlargest(3,heap)	[4, 3, 2]
heapq.nsmallest(2, heap)	[1,2]
heap[0]	min value. Access heap similar to list
heapq.heappop(heap)	Return 1, heap: [2,3,4]

Bit level

~3 ??	inverted bits of 3. -4
number<<num_bits	3<<1: 6
number>>num_bits	3>>1: 1

Set

f = frozenset([1, 2, 2, 3, 3])	frozenset([1, 2, 3]) immutable set
f = set([1,2,3,4])	
f.issubset([1,2])	False
f.issuperset([1,2])	True
f.add(5)	[1,2,3,4,5]
f.remove(1)	f: [2,3,4,5]
f.clear()	
f.intersection([3,6]) f & set([3,6])	Return [3]. Does not update f
f.union([3,6]) f|set([3,6])	Return [2,3,4,5,6]
f.difference([2]) f-set([2])	[3,4,5]
f.symmetric_difference([2,5,8]) f^set([2,5,8])	set([8, 3, 4]) **XOR**
f.update([7,6,2])	f: [2,3,4,5,6,7]

Sequences

a = ['foo', 'bar', 'baz']	
for i, j in enumerate(a): print i, j	returns 0 foo 1 bar 2 baz
a=list('abcsd')	a = ['a', 'b', 'c', 's', 'd']
if a:	Returns True if is not empty
[1] + [2]	Concatenate two list. [1,2]
[1] * 5	[1,1,1,1,1]
b = a[:]/ b = list(a)	Copy a into b. **not a=b**
a = reversed([1, 2, 3])	Returns an **iterator** through sequence in reverse order
for i in a: print i,	3 2 1

Sequences (cont)

fahrenheit = map(lambda x: (float(9)/5)*x + 32, [39.2, 36.5])	Returns a new list where ith item is fct(ith items of sequence(s))
reduce(lambda x,y: x*y, [1, 2, 3])	Returns a value: fct applied cumulatively
filter(lambda a: a % 2 == 0, [1, 3, 5, 2])	Returns a list where fct(item) is True.
a = zip([1,2,3], ["a", "b", "c"])	Returns a list of tuples, ith tuple contains ith items of each sequences
a=[0,1,2,3]	[start:stop[:step]]
a[::-1]	Reverse [3,2,1,0]
a[::-2]	[3,1]
a[::2]	[0,2]
a.insert(index, value)	
a.remove(value)	Remove first occurrence of value
a.pop(index)	Return item at this index and remove it from a
a.count(value)	
a.index(value [,start[,stop]])	Return first index of item in a. start and end can be defined
a.reverse()	Reverse items in a. Returns None!
del a[1:3]	Remove sublist a[1:3] from a. a=[0,3]
range(10,0,-1)	[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
range(0,10,-1)	[]
range(0,10,1)	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Others

import re	
re.split("[-+//]+", "3+22+63/3")	['3', '2', '2', '63', '3']
float("inf")	inf
list(itertools.product(*listoflist))	return all
import sys	
sys.maxint	Maximum integer value
-1-sys.maxint	Minimum integer value. Convert int to long automatically if its getting larger
from difflib import SequenceMatcher	
match = SequenceMatcher(None, string1, string2).find_longest_match(0, len(string1), 0, len(string2))	match: Match(a=0, b=15, size=9)