Cheatography

Python Cheat Sheet

by yunshu (xys) via cheatography.com/130138/cs/25798/

Basics

abs()	absolute value
hash()	hash value
set()	creat a Set object
all()	True if all elements are true
any()	True if any element is true
min()	return minimum
max()	return maximum
divmod(a,b)	return (a//b,a%b)
hex()	hexadecimal
oct()	octal
bin()	binary
dir()	return all attributes of obj
sorted(*iter*)	return a new sorted list from iter
open(p-ath,mode)	open a file
int()	creat an Int object
str()	return the string form
float()	creat a float obj
list()	creat a List obj
isinstance(o-bj,class)	check if obj belongs to class
ord(c)	return ASCII of c
chr(n)	return char of ASCII
sum(iter)	return sum of iter
filter(pred,iter)	return list of elements meeting pred
pow(a,b)	return a^b
callable(obj)	True if obj is callable
type()	return type of obj
zip()	zip('ab','12') -> a1,b2
map(f,xs)	return ys = f(xs)
round()	rounded number

thread

import threading	
t = threading.Thread(t-arget=fun, args = iter,-name=thread name)	creat a thread
t.start()	start thread t
t.join()	join thread t, other threads waiting until t finishes
lock = threading.Lock()	create a lock
lock.acquire()	current thread acquires the lock
lock.release()	current thead release lock

str-library

s1+s2	string concatenation
s*5	repeating 5 times of s
[0] or [:]	subscription and slice
in/not in	member test
r/R	no escape: r'\n' = '\n' (no new line)
%	string formatting (%d <=> integer))
s.capitalize()	capitalize the first char in s
s.count(x,be-g=0,end=len(s)))	count the number of occurence of x in s
s.endswith(suff-ix,beg=0,end=le-n(s))	check if s ends with suffix (within the given area)
s.startswith(prefi-x,beg=0,end=-len(s))	check if s starts with x
s.expandtabs(ta-bsize=8)	expand the "tab" in s to space
s.find(x,beg=0,-end=len(s))	return start index of x in s if x is in s, else -1

str-library (cont)

s.inde-x(x,be-g=0,en-d=len(s))	similar to find, but raises an exception if x is not in s
s.rindex()	
s.rfind()	
s.isal-num()	True if every char(>=1) in s is number or letter
s.isalpha()	True if every char(>=1) in s is letter
s.isdigit()	True if every char(>=1) in s is number
s.isnu-meric()	True if all characters in the string are numeric(>=1)
s.isde-cimal()	Return True if the string is a decimal string(>=1), False otherwise.
s.issp-ace()	True if s only contains space
s.join()	Concatenate any number of strings using s as delimiter
s.upper()	all to uppercase
s.isupper()	True if all cased chars are supercase(>=1)
s.lower()	all to lowercase
s.islower()	True if all cased chars are lowercase(>=1)
s.lstrip()	return a new string leading whitespace removed
s.strip()	Return a copy of the string with leading and trailing whitespace removed

By **yunshu** (xys)
cheatography.com/xys/

Published 20th December, 2020.
Last updated 20th December, 2020.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

str-library (cont)

s.rstrip()	Return a copy of the string with trailing whitespace removed.
s.split(del,-maxsplit = s.coun-t(del))	Return a list of the words in the string, using del as the delimiter string
s.splitlines-(keepends)	Return a list of the lines in the string, breaking at line boundaries. Line breaks are not included in the resulting list unless keepends is given and true.
s.swap-case()	lower <-> upper
s.titile()	titilization: all words are capitalized
s.replace-(ol-d,new,*max*)	Return a copy with all occurr-ences of substring old replaced by new

list

[1,2,3]+[-4,5,6]	[1,2,3,4,5,6]
arr = [0]*10	Array arr = new Array[10]
l.appe-nd(obj)	append obj at end of l
l.count(obj)	count occurence number of obj in l
l.extend(*iter*)	Extend list by appending elements from the iterable
l.index(obj,-beg=0,end-=len(l))	Return first index of value. Raises ValueError if the value is not present

list (cont)

l.remo ve(obj)	Remove first occurrence of value. Raises ValueError if the value is not present

l.sort(cmp=None,key=None,reverse=False)

tuple

(1,2)+(3,4)	(1,2,3,4)
(0)*10	(0,0,0,0,0,0,0,0,0,0)

dict (hashtable)

d = {'age':20}	create a dict
d['age'] = 30	add/update value
d.pop(key)	deleting key and value
d.clear()	create a dict
d.get(key,de-fault=None)	get value by key, or default if key not exists
d.has_key-(key)	True if d has key
d.items()	a list of (key,value) of d
d.update(d2)	updating (k,v) of d2 to d1
d.pop(key)	delete and return the value pointed by the key
d.popitem()	delete and return a pair of (k,v) randomly

dict features:
1. fast for searching and inserting, which won't be affected by the number of keys
2. occupy a lot of memory

set

s = set([1,2,3])	creat a set
s.add(4)	adding element
s.remove(4)	deleting element
s1 & s2	intersection of sets
s1 | s2	union of sets
s.clear()	clear the set
s.pop()	remove one element randomly

s1.symmetric_difference(s2)

copy

a = li	a: new pointer to li
a = li[:]	first level copy
a = list(li)	first level copy
a = copy.copy(li)	first level copy
a = copy.deepcopy(li)	recursive copy

```
import copy
li = [1,2,3,[4,5]]
```

list generation expression

```
[a+b for a in list1 for b in list2]
```

@property

```
class Student(object):
    @pr operty
    defscore(self): return 100
    @score.setter
    defscore(self, value): pass
```

the three names (score) should be consistent

regular expression

```
import re
```

re.match(-pattern,s-tring,*flags*)	Try to apply the pattern at the start of the string, returning a Match object, or None if no match was found.
re.search-(pattern,-string,*flag*)	Scan through string looking for a match to the pattern, returning a Match object, or None if no match was found.
matchO-bject.s-pan()	return (a,b) where a is the start inex and b is the end index of the matching
re.compil-e(p-attern,*flag*)	Compile a regular expression pattern, returning a Pattern object, which can be used in re.match/re.search

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

parameters

func(*-args)	accepting any parameters
func(**kw)	accepting only key word parameters

closure

```
def
create_myFunc_at_runtime(*runtime_para):
   def myFunc(x):
        (return x + runtim e_para)
       pass
    return myFunc
```

Build A Class: Test

__slots__ = ('name','age')	this class have only 2 attributes now: name & age
__eq__(self,obj)	override "==" operator
__ne__(self,obj)	!=
__le__(self,o)	<=
__ge__(self,o)	>=
__lt__(self,o)	<
__gt__(self,o)	>
__str__(self)	override str()
__repr__(self)	repr()
__len__(self)	len()
__getitem__(-self,n)	subscritable and slice-able
__setitem__(-self,key,-value)	supporting item assignment
__call__(self)	-> callable

inheritance

```
overriding __init__:
super( child class, sel f)._ -
_i nit __( *para)
```

datetime

from datetime import datetime

dt = datetime(201-5,4,19,12,20)	2015-04-19 12:20:00
datetime.now()	current date and time
datetime.strptime('2015-6-1 18:19:59','%Y-%m-%d %H:%M:%S')	str -> datetime
dt.strftime('%a,%b %d %H %M')	datetime -> str
from datetime import timedelta	datetime addition and subtraction

now + timedelta(hours = 10)

now + timedelta(days=1)

JSON

import json

js=json.dump-s(py)	convet from python obj to json
py = json.load-s(js)	convert from json to python obj

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com