Cheatography

HiBe Cheat Sheet

by HiBe (HiBe) via cheatography.com/196652/cs/41358/

Strings [immutable, iterable]

`string = " hello world"`	"hello world"
`string = " hello world".c ap ita lize()`	"Hello world"
`string = " Hello World".l ower()`	"hello world"
`string = " Hello World".u pper()`	"HELLO WORLD"
`string = " hello world".t itle()`	"Hello Word"
`string = " Hello wOrld".s wa pcase()`	"hELLO WoRLD"
`string = " ß".c ase fold()`	"ss"

`.casef old()` is a more aggressive `.lower()` used for string comparisons.

`int = " abaaba ababa".c ou nt( " aba ")`	3

"ababa" is counted to have 1 "aba"

`int = " aca dac ad".f in d("c ad")`	1
`int = " aca dac ad".r fi nd( " cad ")`	5

`string.find(`***`value, start, end`***`)` finds the ***first*** occurrence of the ***value*** after ***start*** before ***end***.

`str``.find(`***`str2`***`) == `***`str`***`.index(`***`str2`***`)`, except for when ***str2*** isn't found `.find()` returns 0 while `.index()` throws an error

`string = "ab ab aa ab ab".r ep lac e("a b", " x ", 3)`	"x x aa x ab"
`str``.replace(`***`oldvalue, newvalue, count`***`)`	replaces ***count*** of ***oldvalues*** in the ***string*** with ***newvalues***. Default ***count*** is alll.

`str``.format() == f'`***`str`***`'`

`bool = " _,_._b an_ ana __..,".s tr ip( " _,." -)`	"ban_ana"
`bool = " _,_._b an_ ana __..,".l st rip ("_,.")`	"ban_ana__.."
`bool = " _,_._b an_ ana __..,".r st rip ("_,.")`	"ban_ana__.."
`string = " hel lo".c en ter(10, '.')`	"..hello..."

`str``.center(`***`int, char`***`)` returns a string of ***int*** characters, with ***string*** in the middle and ***chars*** on the front and the end.

`bool = " ab".i sa lpha()`	True
`bool = " a1".i sa lnum()`	True
`bool = " -12 ".is num eric()`	False
`bool = " 1.2 ".is digit()`	False
`bool = " aba ".en dsw ith ("ba ")`	True
`bool = " aba ".st art swi th( " ab")`	True
`bool = " a12 ".is lower()`	True
`bool = " A12 ".is upper()`	True
`list = "Master Lordern".split("er")`	['Mast', ' Lord', 'n']
`string = " -10.2".z fi ll(7)`	"-0010.2"

`.zfill()` fills the beginning of the string with zeroes until the specified amount of characters are taken up, -, + and . included. Unlike .rjust(), .zfill() puts 0s after - and + signs.

`string = " hello world".t itle()`	"Hello Word"

By **HiBe** (HiBe)
cheatography.com/hibe/

Published 19th November, 2023.
Last updated 19th November, 2023.

Strings [immutable, iterable] (cont)

`string = " Hello wOrld".s wa pcase()`	"hELLO WoRLD"
`string = " hello world"[2:7]` *string slicing*	"llo w"

All string methods ***return*** a new string ***without mutating*** the original string

Lists [mutable, iterable]

`list = []`	[]
list = list()	[]
`list = [1, 2, 3]`	[1, 2, 3]
`list.a ppe nd(4)`	[1, 2, 3, 4]
`list.i nse rt(1, 8)`	[1, 8, 2, 3, 4]
`list.i nsert(`***`ind, elem`***`)` inserts ***elem*** at position ***ind***	
`list.e xte nd([8, 9, 1, 4, 3])`	[1, 8, 2, 3, 4, 8, 9, 1, 4, 3]
`list.e xtend(`***`<it era ble>`***`)` extends the list by the elements of the ***iterable***	
`list.pop()`	[1, 8, 2, 3, 4, 8, 9, 1, 3]
`list.p op(0)`	[8, 2, 3, 4, 8, 9, 1, 3]
`list.pop(`***`ind`***`)` pops the element at position ***ind***	
`list.r emo ve(8)`	[2, 3, 4, 8, 9, 1, 3]
`list.r emove(`***`elem`***`)` removes the ***first*** instance of ***elem***	
`list.i ndex(3)`	1
`list.i ndex(`***`elem`***`)` ***returns*** the index of the ***first elem***	
`list.i ndex()` ***does not mutate*** the original list	
`list.c ount(3)`	2
`list.c ount(`***`elem`***`)` ***returns*** the count of elements with value ***elem***	
`list.c ount()` ***does not mutate*** the original list	
`list.s ort()`	[1, 2, 3, 3, 4 ,8, 9]
`list.s ort()` cannot sort ***int*** against ***str***	
`list.r eve rse()`	[9, 8, 4, 3, 3, 2, 1]
list slicing: `list2 = list[1:5]`	[8, 4, 3, 3]
`list2 = list.c opy()`	list2 is [9, 8, 4, 3, 3, 2, 1]
list slicing or `list.c opy()` ***do not mutate*** the original list, and create ***shallow copies***.	
`list.c lear()`	[]

Tuples [immutable, iterable]

`tuple = ()`	
`int = ("a", " a", " ab").co unt ('a')`	2
`int = ("a", " a", " ab").in dex ('a')`	0

`.index()` returns the position of the first instance

Dictionaries [mutable, iterable]

`dictionary = {}`	{}
`dictionary = dict()`	{}
`dictionary = {key1: value1, key2: value2}`	{key1: value1, key2: value2}
`dictio nar y["s tr1 "] = " str 2"`	{key1: value1, key2: value2, 'str1': 'str2'}
`dictio nar y.p op( 'str1')`	{key1: value1, key2: value2}
`dictio nar y.p opi tem()`	{key1: value1}

`.popitem()` removes the ***last*** added element

dictionary2 = {key2: value2}

dictionary.update(dictionary2)	{key1: value1, key2: value2}

.update(***dict***) updates the dictionary with the key:value pairs of ***dict***. In case of a conflict, ***dict*** is prioritized.

`list = dictio nar y.k eys()`	[key1, key2]
`list = dictio nar y.v alues()`	[value1, value2]
`list = dictio nar y.i tems()`	[(key1, value1), (key2, value2)]

`.items()` returns a list of tuples

`something = dictio nar y.get(`***keyname, value***`)`

if ***keyname*** in a dictionary does not exist as a key, ***value*** is returned.

`dictio nary2 = dictio nar y.c opy()`	{key1: value1, key2: value2}
`dictio nar y.c lear()`	{}

Functions

```
def function(parameter1, parameter2 = defaul tvalue):
    <so met hin g>
    return value
function(argument1, argument2)
```