Python - string methods like

find(ch)---returns first index

<https://www.programiz.com/python-programming/methods/string/find>

rfind(ch) --- returns highest index of ch

<https://www.programiz.com/python-programming/methods/string/rfind>

replace

split()

collections -

OrderedDict,

Counter,

itertools.takewhile,

itertools.chain,

itertools next function,

map,

filter,

functools reduce,

ord and chr functions,

bitwise operators, xor, zip function, \* unpacks the sequence/collection into positional arguments, pandas

Python dictionary

setdefault() Method

Description

The method setdefault() is similar to get(), but will set dict[key]=default if key is not already in dict.

Syntax

Following is the syntax for setdefault() method −

dict.setdefault(key, default=None)

Parameters

key − This is the key to be searched.

default − This is the Value to be returned in case key is not found.

Return Value

This method returns the key value available in the dictionary and if given key is not available then it will return provided default value.

Example

The following example shows the usage of setdefault() method.

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7}

print "Value : %s" % dict.setdefault('Age', None)

print "Value : %s" % dict.setdefault('Sex', None)

When we run above program, it produces following result −

Value : 7

Value : None

get() method

Get() method for dictionaries in Python

In python dictionaries, following is a conventional method to access a value for a key.

dic = {"A":1, "B":2}

print(dic["A"])

print(dic["C"])

The problem that arises here is that the 3rd line of the code returns a key error :

Traceback (most recent call last):

File ".\dic.py", line 3, in

print (dic["C"])

KeyError: 'C'

The get() method is used to avoid such situations. This method returns the value for the given key, if present in the dictionary. If not, then it will return None (if get() is used with only one argument).

Syntax :

Dict.get(key, default=None)

Example:

dic = {"A":1, "B":2}

print(dic.get("A"))

print(dic.get("C"))

print(dic.get("C","Not Found ! "))

Output:

1

None

Not Found !

list methods

join:

divmod

Python divmod()

The divmod() method takes two numbers and returns a pair of numbers (a tuple) consisting of their quotient and remainder.

The syntax of divmod() is:

divmod() Parameters

The divmod() takes two parameters:

x - a non-complex number (numerator)

y - a non-complex number (denominator)

Return Value from divmod()

The divmod() returns

(q, r) - a pair of numbers (a tuple) consisting of quotient q and remainder r

If x and y are integers, the return value from divmod() is same as (a // b, x % y).

If either x or y is a float, the result is (q, x%y). Here, q is the whole part of the quotient

import string

lower = string.ascii\_lowercase

upper = string.ascii\_uppercase

lower has value 'abcdefghijklmnopqrstuvwxyz'

upper has value 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

Python any()

The any() method returns True if any element of an iterable is True. If not, any() returns False.

The syntax of any() is:

any(iterable)

any() Parameters

The any() method takes an iterable (list, string, dictionary etc.) in Python.

Return Value from any()

any() returns:

True if at least one element of an iterable is true

False if all elements are false or if an iterable is empty

When Return Value

All values are true True

All values are false False

One value is true (others are false) True

One value is false (others are true) True

Empty Iterable False

Python all()

The all() method returns True when all elements in the given iterable are true. If not, it returns False.

The syntax of all() method is:

all(iterable)

all() Parameters

The all() method takes a single parameter:

iterable - any iterable (list, tuple, dictionary, etc.) which contains the elements

Return Value from all()

The all() method returns:

True - If all elements in an iterable are true

False - If any element in an iterable is false

Truth table for all() When Return Value

All values are true True

All values are false False

One value is true (others are false) False

One value is false (others are true) False

Empty Iterable True

In Python, a string of text can be aligned left, right and center.

.ljust(width)

This method returns a left aligned string of length width.

>>> width = 20

>>> print 'HackerRank'.ljust(width,'-')

HackerRank----------

.center(width)

This method returns a centered string of length width.

>>> width = 20

>>> print 'HackerRank'.center(width,'-')

-----HackerRank-----

.rjust(width)

This method returns a right aligned string of length width.

>>> width = 20

>>> print 'HackerRank'.rjust(width,'-')

----------HackerRank

Textwrap

The textwrap module provides two convenient functions: wrap() and fill().

textwrap.wrap()

The wrap() function wraps a single paragraph in text (a string) so that every line is width characters long at most.

It returns a list of output lines.

>>> import textwrap

>>> string = "This is a very very very very very long string."

>>> print textwrap.wrap(string,8)

['This is', 'a very', 'very', 'very', 'very', 'very', 'long', 'string.']

textwrap.fill()

The fill() function wraps a single paragraph in text and returns a single string containing the wrapped paragraph.

>>> import textwrap

>>> string = "This is a very very very very very long string."

>>> print textwrap.fill(string,8)

This is

a very

very

very

very

very

long

string.

>>> from collections import Counter

>>>

>>> myList = [1,1,2,3,4,5,3,2,3,4,2,1,2,3]

>>> print Counter(myList)

Counter({2: 4, 3: 4, 1: 3, 4: 2, 5: 1})

>>>

>>> print Counter(myList).items()

[(1, 3), (2, 4), (3, 4), (4, 2), (5, 1)]

>>>

>>> print Counter(myList).keys()

[1, 2, 3, 4, 5]

>>>

>>> print Counter(myList).values()

[3, 4, 4, 2, 1]