Python : In-built Functions & Methods

A **function** is a block of code to carry out a specific task, will contain its scope and is called by name.

Example:

```
len([2,3,4,5,6]) --> 5: returns count of elements.
sum([2,3,4,5,6]) --> 20: returns sum of all the elements.
```

A method in Python is similar to a function, except it is associated with objects/classes. Methods in Python are very similar to functions except for two significant differences.

The method is implicitly used for an object for which it is called.

The method is accessible to data that is contained within the class.

Example:

```
'hello world'.count('o') --> 2: count the number of 'o's.
ex = {'name':'abc','age':20}
ex.keys() --> dict_keys(['name', 'age']): .keys() is the method
                                           which gives keys of dictionary as output
```

String Methods

Imp: All string methods return new values. They do not change the original string. [As String is immutable]

Method	Scenarios	Compatibility	Examples
count()	Returns the number of times a specified value occurs in a string	String, List, Tuple	ex = 'Hello World' ex.count('o') —> 2
index()	Searches the string for a specified value and returns the position of where it was found.	String, List, Tuple	ex = 'Hello World' ex.index('o') —> 4 ; returns index of the first occurrence
replace()	Returns a string where a specified value is replaced with a specified value.	String	ex = 'Hello world' ex.replace('world', 'World') —> "Hello World"
lower()	Converts a string into lowercase	String	ex = 'Hello world' ex.lower() —> "hello world"
upper()	Converts a string into upper case.	String	ex = 'Hello world' ex.upper() —> "HELLO WORLD"
title()	Converts the first character of each word to upper case.	String	ex = 'hello world' ex.title() —> 'Hello World'
strip()	Returns a trimmed version of the string.	String	ex = ' hello world ' ex.strip() —> 'hello world'
split()	Splits the string at the specified separator and returns a list.	String	ex = 'Hello world' ex.split(' ') —> ['Hello', 'world']
join()	method takes all items in an iterable and joins them into one string.	String	ex = ['h' , 'e' , 'l' , 'l' , 'o'] "-".join(ex) —> h-e-l-l-o

List Methods

Method	Description	Compatibility	Examples
count()	Returns the number of elements with the specified value	String, List, Tuple	ex = [1,2,3,3,3,5] ex.count(3) —> 3
index()	Returns the index of the first element with the specified value	String, List, Tuple	ex = [1,2,3,3,5] ex.count(3) —> 2
append()	Adds an element at the end of the list	List	ex = [1,2,3,3,5] ex.append(6) —> [1,2,3,3,5,6]
extend()	Add the elements of a list (or any iterable) to the end of the current list	List	ex = [1,2,3,3,5] ex.extend([6,7]) —> [1,2,3,3,5,6,7]
insert()	Adds an element at the specified position	insert(position, element)	ex = [1,2,3,3,5] ex.insert(0,6) —> [6,1,2,3,3,5]
pop()	Removes the element at the specified position (.pop(position))	List, dictionary, set	ex = [1,2,3,3,5] ex.pop() —> [1,2,3,3] ex.pop(0) —> [2,3,3,5]
remove()	Removes the first item with the specified value	List, set	ex = [1,2,3,3,5] ex.remove(1) —> [2,3,3,5] ex.remove(3) —> [1,2,3,5]
sort()	Sorts the list	List	ex = [2,6,3,5] ex.sort()—> [2,3,5,6] ex.sort(reverse = True)—> [6,5,3,2]

clear()	Removes all the elements from the list	List, Dictionary, Set	ex = [2,6,3,5]
			ex.clear()
			print(ex) —> []

copy()	Returns a copy of the list	List, Dictionary, Set	ex = [2,6,3,5]
			ex1 = ex.copy() { Shallow copy}
			print(ex1 == ex) —> True

Dictionary Methods

Method	Description	Compatibility	Examples

keys()	Returns a list containing the dictionary's keys	dictionary	ex = {'a' :1 , 'b':2}
			ex.keys() —> dict_keys(['a', 'b'])

values()	Returns a list of all the values in the dictionary	dictionary	ex = {'a' :1 , 'b':2}
			ex.values() —> dict_values([1, 2])

items()	Returns a list containing a tuple for each key-value pair	dictionary	ex = {'a' :1 , 'b':2}
			ex.items() —> dict_items([('a', 1), ('b', 2)])

setdefault()	Returns the value of the specified key. If the key does not exist, insert the key with the specified value	dictionary	ex = {'a' :1 , 'b':2}
			ex.setdefault('a',"None") —> 1
			ex = {'a' :1 , 'b':2}
			ex.setdefault('c',"None") —> None (No Error)

update()	Updates the dictionary with the specified key-value pairs	dictionary	ex = {'a' :1 , 'b':2}
			ex.update({'c':2}) —> {'a': 1, 'b': 2, 'c': 2}

pop()	Removes the element with the specified key	List, dictionary, set	ex = {'a' :1 , 'b':2}
			ex.pop('b') —> {'a': 1}

clear()	Removes all the elements from the dictionary	List, Dictionary, Set	ex = {'a' :1 , 'b':2}
			ex.clear()
			print(ex) —> {}

copy()	Returns a copy of the dictionary	List, Dictionary, Set	ex = {'a' :1 , 'b':2}
			ex2 = ex.copy()
			print(ex2) —>

Set Methods

Method	Description	Compatibility	Examples

add()	Adds an element to the set	set, dict.keys()	ex = {'a','b'}
			ex.add('c') —> {'c', 'b', 'a'}

update()	Update the set with another set or any other iterable	set, dict.keys()	ex = {'a','b'}
			ex.update(['c','d']) —> {'c', 'b', 'a', 'd'}

union()	Return a set containing the union of sets	set, dict.keys()	ex1 = {'a','b','c'}
			ex2 = {'c','d','e'}
			ex1.union(ex2) —> {'a', 'b', 'c', 'd', 'e'}

intersection()	Returns a set that is the intersection of two or more sets	set, dict.keys()	ex1 = {'a','b','c'}
			ex2 = {'c','d','e'}
			ex1.intersection(ex2) —> {'c'}

difference()	Returns a set containing the difference between two or more sets	set, dict.keys()	ex1 = {'a','b','c'}
			ex2 = {'c','d','e'}
			ex1.difference(ex2) —> {'a', 'b'}

symmetric_difference()	Returns a set with the symmetric differences of two sets	set, dict.keys()	ex1 = {'a','b','c'}
			ex2 = {'c','d','e'}
			ex1.symmetric_difference()(ex2) —> {'a', 'b', 'd', 'e'}

remove()	Removes the specified element	List, set	ex1 = {'a','b','c'}
			ex1.remove('a') —> {'b', 'c'}

clear()	Removes all the elements from the set	List,Dictionary, Set	ex1 = {'a','b','c'}
			ex1.clear() —> {}

copy()	Returns a copy of the set	List, Dictionary, Set	ex1 = {'a','b','c'}
			ex2 = ex1.copy()
			print(ex1 == ex2) —> True

Commonly used Inbuilt Functions & Methods

Python has several functions that are readily available for use. These functions are called built-in functions.

General Purpose Functions

Function	Description	Compatibility	Examples
print()	function prints the specified message to the screen or other standard output device.	Compatible with all data types.	print("Hello World") —> "Hello World" print([1,2,4,5]) —> [1,2,4,5] —————————- To include variable in the print use f string x = 'hello world' print(f"I want to print {x}") —> I want to print hello world.
input()	function allows user input and stores value as a string.	Accepts all types of input	input("Enter your name:") —> Asks user to input name. ——————————- Input will stores values as string data type. Typecasting is needed x = int(input('Enter your

			age')) → Str to Int y = eval(input('Enter list/tuple/set/dict') —> To get input as list, tuple,set, dict
help()	Executes the built-in help system		help() —> prompts for keywords for which help is needed
int()	Returns an integer number		initialise the int format.
float()	Returns a floating point number		initialise the float format.
bool()	Returns the boolean value of the specified object		initialise the boolean format.
type()	Returns the type of an object		ex = [2,3,4] type(ex) —> list ; returns the type of object.
round()	Rounds a numbers		round(1.45,1) —> 1.4 ; rounds the number after decimal ———————————- round(1.4567,2) —> 1.46 round(1.4537,2) —→ 1.45 round(1.4567)—> 1 : by default return the integer before decimal if no decimal given as an input.
str()	Returns a string object		typecast or initialise
list()	Returns a list		typecast or initialise
tuple()	Returns a tuple		typecast or initialise
dict()	Returns a dictionary		typecast or initialise
set()	Returns a new set object		typecast or initialise
len()	Returns the length of an object	string, list, tuple, set, dictionary	ex = [3,4,5] len(ex) —> 3
sum()	Sums the items of an iterator	Only takes numbers in input	ex = [3,4,5] sum(ex) —> 12
max()	Returns the largest item in an iterable		ex = [3,4,5] max(ex) —> 5 ————— In case of all string in a list, it will output base on unicode of str. ex = ['A','B', 'c','d'] max(ex) —> d ;as unicode of d is max.
min()	Returns the smallest item in an iterable		ex = [3,4,5] min(ex) —> 3 ——- In case of all string in a list, it will output base on unicode of str. ex = ['A','B', 'c','d'] min(ex) —> A ;as unicode of A is min.
sorted()	function returns a sorted list of the specified iterable object.	String, List, Tuple, set, Dictionary	ex = [3,6,4] sorted(ex) —> [3,4,6] sorted(ex, reverse = True) —> [6,4,3] —————- In case of all string in a list, it will output base on unicode of str. If Reverse = True —>

			descending order ; if blank or reverse = False —> ascending order
abs()	Returns the absolute value of a number	Accepts int and float as input	abs(-1) —> 1 abs(1.0) —> 1.0 abs('a') —> error
enumerate()	Takes a collection (e.g. a tuple) and returns it as an enumerate object	all data types	ex = [3,7,5] list(enumerate(ex)) —> [(0, 3), (1, 7), (2, 5)] ——- Breaks it down to tuple of index and value in case of dictionary: dict(enumerate(ex)) —> {0:3,1: 7,2:5} Converts to index as keys and elements as values.

zip()	Returns an iterator from two or more iterators		list1 = [2,3,3,5]
			list2 = ['a', 'b', 'c', 'd']
			list(zip(list1, list2)) —> [(2, 'a'), (3, 'b'), (3, 'c'), (5, 'd')]
			dict(zip(list1, list2)) —> {2: 'a', 3: 'c', 5: 'd'}
			——-
			Zips it till the list with smallest number of elements.
			list1 = [2,5]
			list2 = ['a', 'b', 'c', 'd']
			list(zip(list1, list2)) —> [(2, 'a'), (5, 'd')]

eval()	Evaluates and executes an expression		eval('[2,3,4,6]') —> [2,3,4,6] eval("print('hello')") —> hello
range()	Returns a sequence of numbers, starting from 0 and increments by 1 (by default)		list(range(1,6,1)) —> [1,2,3,4,5] list(range(1,6,2)) —> [1,3,5] ———— list(range(2,8,3)) —> [2,5,8] list(range(10,0,-2)) —> [10,8,6,4,2] Follows same concept of I,j,k in slicing

del()	Deletes the object		ex = 'hello'
			del ex —> deletes the variable.
			——-
			can be used to delete the key in dictionary
			ex = {1:'hello',2:'world'}
			del ex[1] —> {2:'world'}

Other miscellaneous functions

Method	Description	Compatibility	Examples
dir()	Returns a list of the specified object's properties and methods		dir() —> Prints all the objects
chr()	Returns a character from the specified Unicode code.		chr(49) —> '1'
ord()	Convert an integer representing the Unicode of the specified character		ord('1') —> 49
bin()	Returns the binary version of a number		bin(17) —> '0b10001'