MANIPAL
ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

MDS5103/MSBA5104 Segment 04

FUNCTIONS AND METHODS - TUTORIAL

Table of Contents

Introduction

In this topic, we will look at various functions and methods available in Python and the differences between the same.

1. Functions in Python

A function is a reusable code aimed at solving a particular task. For example, max, round, type, sorted, help, or length. Python provides several in-built functions to do such specific operations. They can be used independently and are not invoked using an object of a class. The code below demonstrates the functionality of functions 'type()', 'len()', and 'int()'. The 'type()' function returns the type of the object, the 'len()' function returns the length or the number of elements in an object and the 'int()' function converts the object to an integer.

```
Python = True
Age= 25.5

Income = ['Amit', 75000, 'Tina', 65000, 'Heena', 45000, 'Reena', 25000]

# 'type' is a function to identify the 'type' of variable: Age

print(type(Age))

# 'len' is a function to get the length of variable: Income
print(len(Income))

# Convert var to an integer: out2
IntPy = int(Python)

IntPy
```

Output

```
<class 'float'>
8
1
```

The code below demonstrates the functionality of the 'sorted()' function. In this, the two lists are concatenated first and then, 'sorted()' function is applied. This will sort the elements in ascending order. To sort in descending order, the reverse attribute can be set to 'True' as shown below.

```
# Create lists
Age1 = [25,65,34,44]
Age2 = [10,50]
# Concatenate together
Age_f = Age1 + Age2
print(Age_f)
# Sort in descending order using the function 'sorted'
Age_full = sorted(Age_f)
# Print out full_sorted
print(Age_full)
print(Age_FULL)
```

Output

```
[25, 65, 34, 44, 10, 50]
[10, 25, 34, 44, 50, 65]
[65, 50, 44, 34, 25, 10]
```

The 'help()' function can be used with any function to obtain information on the function. The code below returns the documentation on the 'sorted()' function. The syntax with various options is returned.

```
help(sorted)
```

Output

```
Help on built-in function sorted in module built-ins:
sorted(iterable, /, *, key=None, reverse=False)
    Return a new list containing all items from the iterable in ascending order.
    A custom key function can be supplied to customise the sort order, and the
    reverse flag can be set to request the result in descending order.
```

Figure 1 shows various built-in functions available with Python 3.

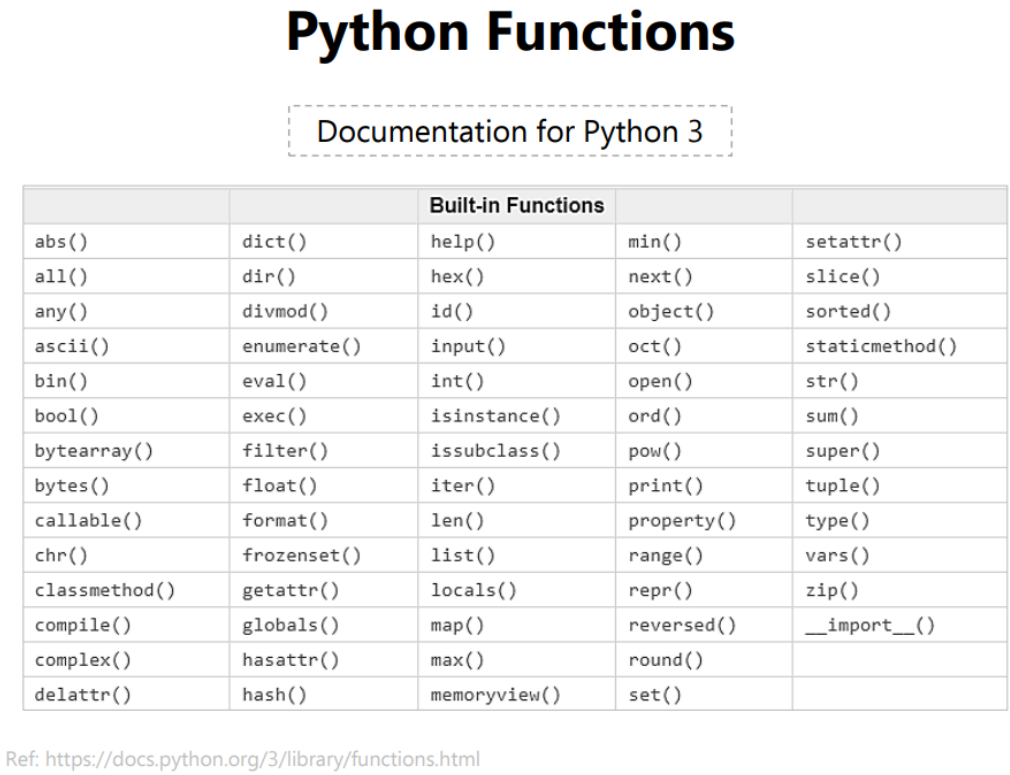

Figure 1: Built-in Functions of Python3

2. Methods in Python

Python objects have associated methods depending on the type of the object. 'Methods' are associated with python objects whereas the 'Functions' are generic and are relatable to any type of object. For example, 'capitalise' and 'replace' are specific methods for 'string' whereas, 'index' and 'count' are specific methods for 'list'. However, 'len' is a function that can work with any python object.

To invoke the methods, a dot is specified after the python object. This is followed by the method name. The code below demonstrates the invocation of the index method on a string object.

```
#Index is a method
Income.index('Heena')
```

Output

4

The 'upper()' and 'replace()' are the two methods that can be used on a string. The code below demonstrates the same.

```
Word = "python"
print(Word.upper())
print(Word.replace("n","n course"))
```

Output

PYTHON

python course

The method 'replace()' is not supported for num/int/float objects. Invoking the 'replace()' method with an integer object will result in an error as shown below.

```
Income.replace(50000,45000)
```

```
AttributeError                            Traceback (most recent call last)

~\AppData\Local\Temp/ipykernel_22320/2658052097.py in <module>
      1 #Replace is not a method for num/int/float objects
----> 2 Income.replace(50000,45000)

AttributeError: 'list' object has no attribute 'replace'
```

The 'index()' method that we used on the string can also be used on an integer or list as shown below.

```
Income.index(45000)
```

Output

```
5
```

```
Word.index('t')
```

Output

```
2
```

The 'append()' method will append an object in the list.

```
Income.append("Raina")
Income
```

Output

```
['Amit', 75000, 'Tina', 65000, 'Heena', 45000, 'Reena', 25000, 'Raina']
```

The 'count()' is a method that can be used with both string and list as shown below.

```
Sample = "Beautiful"
print(Sample.count("u"))
```

Output

```
2
```

The 'count()' method in the list displays how often 25000 appears in Income.

```
print(Income.count(25000))
```

Output

```
1
```

To access the elements in their range, we have to specify them in a list using a 'list()' function on the object.

```
c=range(11)print(c)
print (list(c))
```

Output

```
range(0, 11)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

E-references:

- Dive into Python by Mark Pilgrim

  Link: https://diveintopython3.net/

- Automate the Boring Stuff with Python by Al Sweigart

  Link: https://automatetheboringstuff.com/#toc