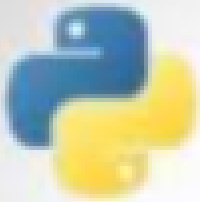
Python

Python,
dir, __dict__ and help method

DAY #71



Python dir( ) Function

```
dir([object])
```

With argument: tries to return the list of valid attributes for that object
Without argument: returns the list of names in the current local scope

The dir() Method in Python

- The dir() method in Python is used to get the list of names of the attributes of the passed object in an alphabetically sorted manner,
- When nothing is passed, the method returns back the list of all the local attributes.

Syntax,

dir([object])

AskPython

Example-1: Use of dir() function without argument

Create a python file with the following script to check the returned value of the dir() function when it is used without any argument. In the script, the dir() function without argument is called before importing any module and after importing two modules.

```
#Print the output of dir() function before importing
any module
print("The output of dir() function before
import:\n", dir())

#Import modules
import sys
import os

#Print the output of dir() function after importing
sys and os modules
print("\nThe output of dir() function after
import:\n", dir())
```

Output:

The following output will appear after executing the above script. After importing the modules, the output shows that os and sys have been added to the dir() output.

```
fahmida@fahmida:~/python$ python3 dir1.py
The output of dir() function before import:
 ['__annotations__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__
name__', '__package__', '__spec__']

The output of dir() function after import:
 ['__annotations__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__
name__', '__package__', '__spec__', 'os', 'sys']
fahmida@fahmida:~/python$
```

Example-2: Use of dir() function for the string as an argument

Create a python file with the following script where the string object has been used as the argument of the dir() function. In this case, the dir() function will return the list of all attributes of the string object.

```
#Define a string value
text = 'LinuxHint'
#Print the output of dir() function for string value
print("\nThe output of dir() function for the string
data:\n", dir(text))
```

Output:

The following output will appear after executing the above script.

```
fahmida@fahmida:~/python$ python3 dir2.py

The output of dir() function for the string data:
 ['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '_
_format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__has
h__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__'
, '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '_
_rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefo
ld', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map',
 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower
', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', '
lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rs
plit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'transl
ate', 'upper', 'zfill']
fahmida@fahmida:~/python$
```

Example-6: Use of dir() function for the object of a class

Create a python file with the following script where the user-defined class object has been used as the argument of the dir() function. In this case, the dir() function will return the list of all attributes of the class object.

```
#Define a class with a constructor
class Client:

    def __init__(self, name, mobile, email):
        self.name = name
        self.mobile = mobile
        self.email = email

#Create an object of the class
objClient = Client('Amir
Hossain','+8801937865645','amir@gmail.com' )

#Print the dir() function output for the object
print("The output of the dir() function for the
object:\n", dir(objClient))
```

```
fahmida@fahmida:~/python$ python3 dir6.py
The output of the dir() function for the object:
 ['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__
ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__',
 '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '_
_setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'email', 'mobile',
 'name']
fahmida@fahmida:~/python$
```

Example-7: Use of dir() function for a particular module

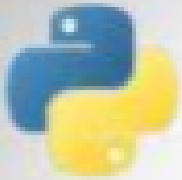
Create a python file with the following script where the module has been used as the argument of the dir() function. The dir() function has been used with **sys** and **randint** modules in the script. The dir() function will return the list of all attributes of these modules as the output.

```
#Import sys module
import sys
#Import randint from random
from random import randint

#Print the output of dir() function for the sys
print("The output of the dir() for sys:\n",dir(sys))
#Print the output of dir() function for the randint
print("\nThe output of the dir() for
randint:\n",dir(randint))
```

```
fahmida@fahmida:~/python$ python3 dir7.py
The output of the dir() for sys:
 ['__breakpointhook__', '__displayhook__', '__doc__', '__excepthook__', '__interactivehook_
_', '__loader__', '__name__', '__package__', '__spec__', '__stderr__', '__stdin__', '__stdo
ut__', '__unraisablehook__', '_base_executable', '_clear_type_cache', '_current_frames', '_
debugmallocstats', '_framework', '_getframe', '_git', '_home', '_xoptions', 'abiflags', 'ad
daudithook', 'api_version', 'argv', 'audit', 'base_exec_prefix', 'base_prefix', 'breakpoint
hook', 'builtin_module_names', 'byteorder', 'call_tracing', 'callstats', 'copyright', 'disp
layhook', 'dont_write_bytecode', 'exc_info', 'excepthook', 'exec_prefix', 'executable', 'ex
it', 'flags', 'float_info', 'float_repr_style', 'get_asyncgen_hooks', 'get_coroutine_origin
_tracking_depth', 'getallocatedblocks', 'getcheckinterval', 'getdefaultencoding', 'getdlope
nflags', 'getfilesystemencodeerrors', 'getfilesystemencoding', 'getprofile', 'getrecursionl
imit', 'getrefcount', 'getsizeof', 'getswitchinterval', 'gettrace', 'hash_info', 'hexversio
n', 'implementation', 'int_info', 'intern', 'is_finalizing', 'maxsize', 'maxunicode', 'meta
_path', 'modules', 'path', 'path_hooks', 'path_importer_cache', 'platform', 'prefix', 'pyca
che_prefix', 'set_asyncgen_hooks', 'set_coroutine_origin_tracking_depth', 'setcheckinterval
', 'setdlopenflags', 'setprofile', 'setrecursionlimit', 'setswitchinterval', 'settrace', 's
tderr', 'stdin', 'stdout', 'thread_info', 'unraisablehook', 'version', 'version_info', 'war
noptions']

The output of the dir() for randint:
 ['__call__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__
func__', '__ge__', '__get__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init
_subclass__', '__le__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__re
pr__', '__self__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__']
fahmida@fahmida:~/python$
```



Python help( ) Function

```
help(object)
```

It invokes the built-in help system.

Help function in Python

The **Python help function** is used to display the documentation of modules, functions, classes, keywords, etc.

The help function has the following syntax:

```
help([object])
```

Python help() function arguments

```
object: Call help of the given object.
```

If the help function is passed without an argument, then the interactive help utility starts up on the console.

Python help() Example

Let us check the documentation of the print function in the python console.

Python3

```
help(print)
```

Output:

```
Help on built-in function print in module builtins:


print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Help function output can also be defined for user-defined functions and classes. The docstring(documentation string) is used for documentation. It is nested inside triple quotes and is the first statement within a class or function or a module.

Let us define a class with functions:

Python3

```
class Helper:
    def __init__(self):
        '''The helper class is initialized'''

    def print_help(self):
        '''Returns the help description'''
        print('helper description')

help(Helper)
help(Helper.print_help)
```

On running the above program, we get the output of the first help function as shown below:

```
Help on class Helper in module __main__:


class Helper(builtins.object)
 |  Methods defined here:
 |
 |  __init__(self)
 |      The helper class is initialized
 |
 |  print_help(self)
 |      Returns the help description
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)


Help on function print_help in module __main__:


print_help(self)
    Returns the help description
```

Python help() function docstring

The docstrings are declared using ”’triple single quotes”’ or """triple double quotes""" just below the class, method or function declaration. All functions should have a docstring.

Accessing Docstrings: The docstrings can be accessed using the __doc__ method of the object or using the help function.

Python3

```
def my_function():
    '''Demonstrates triple double quotes
    docstrings and does nothing really.'''
    return None

print("Using __doc__:")
print(my_function.__doc__)

print("Using help:")
help(my_function)
```

Output:

```
Using __doc__:
Demonstrates triple double quotes
    docstrings and does nothing really.
Using help:
Help on function my_function in module __main__:


my_function()
    Demonstrates triple double quotes
    docstrings and does nothing really.
```

Built-in Method

Python vars() - Returns __dict__ attribute of the given object

Syntax - vars(object)

Get a dictionary from an Objects Fields

In this article, we will discuss how to get a dictionary from object's field i.e. how to get the class members in the form of a dictionary. There are two approaches to solve the above problem:

1. By using the __dict__ attribute on an object of a class and attaining the dictionary. All objects in Python have an attribute __dict__, which is a dictionary object containing all attributes defined for that object itself. The mapping of attributes with its values is done to generate a dictionary.
2. By calling the in-built vars method, which is used to return __dict__ attribute of a module, class, class instance, or an object.

Using Python vars() method

- The vars() method is used to return the __dict__ attribute for any module, class, class instance or any similar object.
- The object must have a __dict__ attribute. Otherwise, vars() will raise a TypeError exception.
- This will act like the locals() function, if you don't give any arguments.

```
>>> print(vars(my_obj))
{a: 10, b: 20}
```

#**Method 1:** To generate a dictionary from an arbitrary object using __dict__attribute:

Python3

```
# class Animals is declared
class Animals:

    # constructor
    def __init__(self):

        # keys are initialized with
        # their respective values
        self.lion = 'carnivore'
        self.dog = 'omnivore'
        self.giraffe = 'herbivore'

    def printit(self):
        print('Dictionary from the object fields\
        belonging to the class Animals:')


# object animal of class Animals
animal = Animals()

# calling printit method
animal.printit()
# calling attribute __dict__ on animal
# object and printing it
print(animal.__dict__)
```

Output:

Dictionary from the object fields belonging to the class Animals:
{'lion': 'carnivore', 'dog': 'omnivore', 'giraffe': 'herbivore'}

#Method 2: To generate a dictionary from an arbitrary object using an in-built vars method:

Python3

```
# class A is declared
class A:

    # constructor
    def __init__(self):

        # keys are initialized with
        # their respective values
        self.A = 1
        self.B = 2
        self.C = 3
        self.D = 4

# object obj of class A
obj = A()

# calling vars method on obj object
print(vars(obj))
```

Output:

```
{'A': 1, 'B': 2, 'C': 3, 'D': 4}
```