



Python sys Module

Last Updated : 18 Nov, 2023

The **sys module** in [Python](#) provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter. Let's consider the below example.

Sys Module in Python

Python sys.version

In this example, **sys.version** is used which returns a string containing the version of Python Interpreter with some additional information. This shows how the sys module interacts with the interpreter. Let us dive into the article to get more information about the sys module.

Python3

```
import sys
print(sys.version)
```

Output:

[Python Basics](#) [Interview Questions](#) [Python Quiz](#) [Popular Packages](#) [Python Projects](#) [Practice Python](#) [AI Wit](#)

```
[GCC 8.4.0]
```

Input and Output using Python Sys

The sys modules provide variables for better control over input or output. We can even redirect the input and output to other devices. This can be done using three variables –

- stdin
- stdout
- stderr

Read from stdin in Python

stdin: It can be used to get input from the command line directly. It is used for standard input. It internally calls the `input()` method. It, also, automatically adds '\n' after each sentence.

Example:

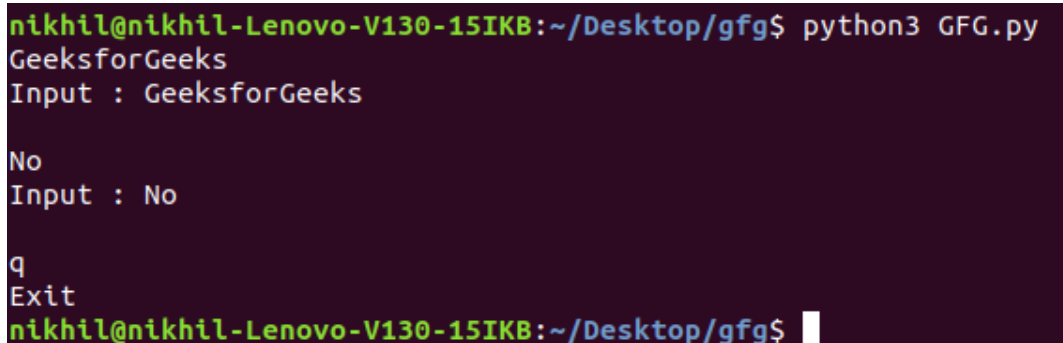
This code reads lines from the standard input until the user enters 'q'. For each line, it prints "Input : " followed by the line. Finally, it prints "Exit".

Python3

```
import sys
for line in sys.stdin:
    if 'q' == line.rstrip():
        break
    print(f'Input : {line}')

print("Exit")
```

Output:



```
nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg$ python3 GFG.py
GeeksforGeeks
Input : GeeksforGeeks

No
Input : No

q
Exit
nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg$
```

Python sys.stdout Method

stdout: A built-in file object that is analogous to the interpreter's standard output stream in Python. stdout is used to display output directly to the screen

console. Output can be of any form, it can be output from a print statement, an expression statement, and even a prompt direct for input. By default, streams are in text mode. In fact, wherever a print function is called within the code, it is first written to `sys.stdout` and then finally on to the screen.

Example:

This code will print the string “**Geeks**” to the standard output. The `sys.stdout` object represents the standard output stream, and the `write()` method writes the specified string to the stream.

Python3

```
import sys
sys.stdout.write('Geeks')
```

Output

Geeks

stderr function in Python

stderr: Whenever an exception occurs in Python it is written to `sys.stderr`.

Example:

This code will print the string “**Hello World**” to the standard error stream. The `sys.stderr` object represents the standard error stream, and the `print()` function writes the specified strings to the stream.

Python3

```
import sys
def print_to_stderr(*a):
    print(*a, file = sys.stderr)

print_to_stderr("Hello World")
```

Output:



Command Line Arguments

Command-line arguments are those which are passed during the calling of the program along with the calling statement. To achieve this using the sys module, the sys module provides a variable called sys.argv. Its main purpose are:

- It is a list of command-line arguments.
- `len(sys.argv)` provides the number of command-line arguments.
- `sys.argv[0]` is the name of the current Python script.

Example: Consider a program for adding numbers and the numbers are passed along with the calling statement.

This code calculates the sum of the command-line arguments passed to the Python script. It imports the sys module to access the command-line arguments and then iterates over the arguments, converting each one to an integer and adding it to a running total. Finally, it prints the total sum of the arguments.

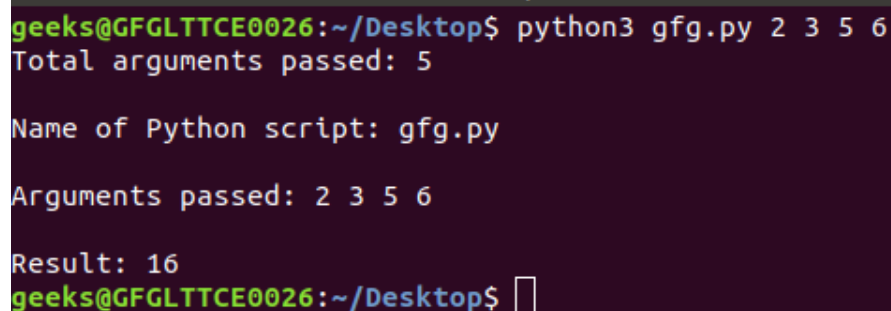
Python3

```
import sys
n = len(sys.argv)
print("Total arguments passed:", n)
print("\nName of Python script:", sys.argv[0])
print("\nArguments passed:", end = " ")
for i in range(1, n):
    print(sys.argv[i], end = " ")
Sum = 0
```

```
for i in range(1, n):
    Sum += int(sys.argv[i])

print("\n\nResult:", Sum)
```

Output:



```
geeks@GFGLTTCE0026:~/Desktop$ python3 gfg.py 2 3 5 6
Total arguments passed: 5

Name of Python script: gfg.py

Arguments passed: 2 3 5 6

Result: 16
geeks@GFGLTTCE0026:~/Desktop$
```

Exiting the Program

`sys.exit([arg])` can be used to exit the program. The optional argument `arg` can be an integer giving the exit or another type of object. If it is an integer, **zero is considered “successful termination”**.

Note: A string can also be passed to the `sys.exit()` method.

Example:

This code checks if the age is less than 18. If it is, it exits the program with a message **“Age less than 18”**. Otherwise, it prints the message **“Age is not less than 18”**. The `sys.exit()` function takes an optional message as an argument, which is displayed when the program exits.

Python3

```
import sys
age = 17
if age < 18:
    sys.exit("Age less than 18")
else:
    print("Age is not less than 18")
```

Output:

An exception has occurred, use %tb to see the full traceback.

SystemExit: Age less than 18

Working with Modules

`sys.path` is a built-in variable within the `sys` module that returns the list of directories that the interpreter will search for the required module.

When a module is imported within a Python file, the interpreter first searches for the specified module among its built-in modules. If not found it looks through the list of directories defined by `sys.path`.

Note: `sys.path` is an ordinary list and can be manipulated.

Example 1: Listing out all the paths

This code will print the system paths that Python uses to search for modules. The `sys.path` list contains the directories that Python will search for modules when it imports them.

Python3

```
import sys
print(sys.path)
```

Output:

```
['/usr/lib/python3.6.zip', '/usr/lib/python3.6', '/usr/lib/python3.6/lib-dynload', '', '/home/nikhil/.local/lib/python3.6/site-packages', '/usr/local/lib/python3.6/dist-packages', '/usr/local/lib/python3.6/dist-packages/language_tool-0.3-py3.6.egg', '/usr/lib/python3/dist-packages', '/home/nikhil/.local/lib/python3.6/site-packages/IPython/extensions', '/home/nikhil/.ipython']
```

Example 2: Truncating the value of `sys.path`

This code will print an error message because the `pandas` module is not in the `sys.path` list. The `sys.path` list is a list of directories that Python will search for modules when it imports them. By setting the `sys.path` list to an empty list, the code effectively disables Python's ability to find any modules.

Python3

```
import sys
sys.path = []
import pandas
```

Output:

```
ModuleNotFoundError: No module named 'pandas'
```

sys.modules return the name of the Python modules that the current shell has imported.

Example:

This code will print a dictionary of all the modules that have been imported by the current Python interpreter. The dictionary keys are the module names, and the dictionary values are the module objects.

Python3

```
import sys
print(sys.modules)
```

Output:

```
{'builtins': <module 'builtins' (built-in)>, 'sys': <module 'sys' (built-in)>, 'frozen_importlib': <module 'importlib_bootstrap' (frozen)>, '_imp_': <module '_imp' (built-in)>, 'warnings': <module 'warnings' (built-in)>, 'thread': <module 'thread' (built-in)>, 'weakref': <module 'weakref' (built-in)>, 'frozen_importlib_external': <module 'importlib_bootstrap_external' (frozen)>, 'io': <module 'io' (built-in)>, 'marshal': <module 'marshal' (built-in)>, 'posix': <module 'posix' (built-in)>, 'zipimport': <module 'zipimport' (built-in)>, 'encodings': <module 'encodings' from '/usr/lib/python3.6/encodings/init.py>', 'codecs': <module 'codecs' from '/usr/lib/python3.6/codecs.py>', 'codecs': <module 'codecs' (built-in)>, 'encodings.aliases': <module 'encodings.aliases' from '/usr/lib/python3.6/encodings/aliases.py>', 'encodings.utf_8': <module 'encodings.utf_8' from '/usr/lib/python3.6/encodings/utf_8.py>', 'signal': <module 'signal' (built-in)>, 'main': <module 'main'>, 'encodings.latin_1': <module 'encodings.latin_1' from '/usr/lib/python3.6/encodings/latin_1.py>, 'io': <module 'io' from '/usr/lib/python3.6/io.py>, 'abc': <module 'abc' from '/usr/lib/python3.6/abc.py>, 'weakrefset': <module 'weakrefset' from '/usr/lib/python3.6/weakrefset.py>, 'bootlocale': <module 'bootlocale' from '/usr/lib/python3.6/bootlocale.py>, 'locale': <module 'locale' (built-in)>, 'site': <module 'site' from '/usr/lib/python3.6/site.py>, 'os': <module 'os' from '/usr/lib/python3.6/os.py>, 'errno': <module 'errno' (built-in)>, 'stat': <module 'stat' from '/
```

Reference Count

sys.getrefcount() method is used to get the reference count for any given object. This value is used by Python as when this value becomes 0, the memory for that particular value is deleted.

Example:

This code prints the reference count of the object a. The reference count of an object is the number of times it is referenced by other objects. An object is garbage collected when its reference count reaches 0, meaning that it is no longer referenced by any other objects

Python3

```
import sys
a = 'Geeks'
print(sys.getrefcount(a))
```

Output

4

More Functions in Python sys

Function	Description
sys.setrecursionlimit()	sys.setrecursionlimit() method is used to set the maximum depth of the Python interpreter stack to the required limit.
sys.getrecursionlimit() method	sys.getrecursionlimit() method is used to find the current recursion limit of the interpreter or to find the maximum depth of the Python interpreter stack.
sys.settrace()	It is used for implementing debuggers, profilers and coverage tools. This is thread-specific and must register the trace using threading.settrace(). On a

Function	Description
	higher level, <code>sys.settrace()</code> registers the traceback to the Python interpreter
<code>sys.setswitchinterval()</code> method	<code>sys.setswitchinterval()</code> method is used to set the interpreter's thread switch interval (in seconds).
<code>sys.maxsize()</code>	It fetches the largest value a variable of data type <code>Py_ssize_t</code> can store.
<code>sys.maxint</code>	<code>maxint/INT_MAX</code> denotes the highest value that can be represented by an integer.
<code>sys.getdefaultencoding()</code> method	<code>sys.getdefaultencoding()</code> method is used to get the current default string encoding used by the Unicode implementation.

Looking to dive into the world of programming or sharpen your Python skills? Our [Master Python: Complete Beginner to Advanced Course](#) is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to advanced techniques. With **hands-on projects**, real-world examples, and expert guidance, you'll gain the confidence to tackle complex **coding challenges**. Whether you're starting from scratch or aiming to enhance your skills, this course is the perfect fit. Enroll now and master Python, the language of the future!

[Previous Article](#)[Python SQLite](#)[Next Article](#)[OS Module in Python with Examples](#)

Similar Reads

Python - sys.settrace()

Python sys module provides some of the powerful functions but they are complex to understand. One of which is the sys.settrace() which is used for implementing...

3 min read

Python | sys.getallocatedblocks() method

This sys module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It provide...

1 min read

Python | sys.getrecursionlimit() method

This sys module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It provide...

1 min read

Python | sys.getdefaultencoding() method

This sys module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It provide...

1 min read

Python | sys.setrecursionlimit() method

This sys module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It provide...

2 min read

Python | sys.setswitchinterval() method

This sys module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It provide...

2 min read

Python | sys.getswitchinterval() method

This sys module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It provide...

1 min read

Python - sys.stdout.flush()

A data buffer is a region of physical memory storage used to temporarily store data while it is being moved from one place to another. The data is stored in a...

3 min read

sys.stdout.write in Python

This is a built-in Python module that contains parameters specific to the system i.e. it contains variables and methods that interact with the interpreter and are...

2 min read

sys.maxsize in Python

maxsize attribute of the sys module fetches the largest value a variable of data type Py_ssize_t can store. It is the Python platform's pointer that dictates the...

2 min read

Article Tags : [Python](#) [Python-sys](#)

Practice Tags : [python](#)



Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design

Low Level Design

UML Diagrams

Interview Guide

Design Patterns

OOAD

System Design Bootcamp

Interview Questions

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Commerce

World GK

Interview Preparation

Competitive Programming

Top DS or Algo for CP

Company-Wise Recruitment Process

Company-Wise Preparation

Aptitude Preparation

Puzzles

GeeksforGeeks Videos

DSA

Python

Java

C++

Web Development

Data Science

CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved