# Interpreter: -

Open the interpreter in Windows with command prompt WIN + R and type python or python3.

Second way to open the interpreter in Windows is python -c command [arg]. . .

Closing the interpreter Control + Z or quit ().

We can use python modules as script by using below command

> python -m module [arg] . . .

# Command Line and Environment: -

Generic Syntax:

python [ -bBdEhiIOqsSuvVWx?] [ -c command | -m module – name | script | -] [ arg]

## Interface Options: -

* When called with standard input connected to a tty device, it prompts for commands and executes them until an EOF is read. For Windows, (EOF can be provided by Ctrl + Z and Enter)
* When called with a file name argument or with a file as standard input, it reads and executes script from that file.
* When called with a directory name argument, it reads and executes a script from that directory.
* When called with a **command -c**,it executes python commands from no. of statements separated by NEWLINE character.

e.g. > python -c “print(‘Hello World’)”

* When called with a **command -m module**, the given module located on python default path will be executed as a script.

e.g. > python -m “test”

# Non- Interactive mode:

In Non – Interactive mode, entire input file parsed before the execution.

All the arguments in command end up in the **sys.argv**. Note that the first argument **sys.argv[0]** is program file source location.

* - C<command>:

Execute python code in commands. Command can be one or more python statements separated by the newlines with significant white spaces in each statements.

In this case, sys.argv[0] is having ‘-C’ value and sys.path should have the current directory location.

* - M<module-name>:

Search module name in current working directory and execute it’s \_\_main\_\_ module.

In this case, we don’t need to provide the .py extension. The module name should be valid python module name.

Package names are also permitted. When a package name is supplied instead of a normal module, python will try to execute the <Pack>.\_\_main\_\_ module.

Note: this option will be valid for precompiled modules but not for built in modules or extension modules written in C.

In this case, sys.argv[0] is having ‘-M’ value before finding the module location and sys.path should have the current directory location.

* - I<Isolation Mode>:

- I can be used in Isolation mode where sys.path contains neither the current directory nor the user’s site-packages directory. All PYTHON\* environment variables will be ignored, too.

In this case, sys.argv[0] is having ‘-’ value.

If no options are given in command then empty string (“”) and current directory will be added to sys.path.

# Script:

Execute the python code contained the script, which must be the filesystem path referring to python file , a directory contains the \_\_main\_\_.py file or a zipfile containing the \_\_main\_\_.py file.

If this options are given, the sys.argv will be script name.

If no interface option is given then -i implied. The sys.argv[0] will be having empty string (“”). sys.path should have the current directory location.

Tab-completion and history editing the enabled by default if available for the plateform.

# Generic options:

* - ?:
* - h:
* -- help:

show the description of all the possible command line options.

* -V
* – version:

print the python version and exit.

* --vv:

It will try to start the interpreter mode with possible import for the default modules present in python lib.

# Miscellaneous options:

* - b:

Issue a warning when comparing bytes or bytearrays to str or bytes to int. Issue the error when the option is given twice(- bb).

* - B:

If given python won’t try to write .pyc files on the import of source modules.

* --check-hash-based-pycs(default/always/never):

Control the validation behavior of hash-based .pyc files. Python validate for source file and cached file difference by comparing the hashed source file with hashed cache file, if it is different then python generate the new hashed cache file for source file. Python uses only timestamp based compare for validation.

There are two types for hashed files, checked and unchecked. Checked hash-based file is compared with cached file while importing python module code in run time. Unchecked hash based files will not be compared with cached files for validation. It will be ignored like it is already updated.

We can set –**check-hash-based-pycs** for unchecked hash-base validation.

* - d:

Turn on the parser debugging output.

* - E:

Ignore all python environment variables that might be set.

* - I:

When a script is passed as first argument or the -c option is used, enter the interactive mode after executing script or the command.

This can be used to inspect global variables or stack trace when a script raises an exception.

* - I:

Run Python in Isolation mode. Is will be helpful in avoiding the malicious injection.

* - o:

Remove assert statements and any code conditional on the value of \_\_debug\_\_. Augment the filename for compiled files by adding **.opt-1** before the .pyc extension. Alternative PYTHONOPTIMIZE.

* - OO:

Do -o and also discard docstrings. Augment the filename for compile files by adding **.opt-2** before the .pyc extension.

* - q:

Do not display python version and copyright messages in interactive mode.

* - R:

Turn on the hash randomization. This option only effective when **PYTHONHASHSEED** environment variable is set to **0.**

This option turn on hash randomization, so that the \_\_hash\_\_() values of str and bytes object are added with an unpredictable random value. This values will be constant in individual python executing process, they are not predictable value between repeated invocations of python.

Hash randomization is intented to provide protection against a denial-of-service caused by carefully-chosen inputs that exploit the worst case performance of a dict construction. O(n^2) Complexity.

**PYTHONHASHSEED** value could be set to integer value. This value will be used to hashing the str and bytes object. If value set to 0 then hash randomization will be disabled.

**PYTHONHASHSEED** allows you to set a fixed value for the hash seed secret.

* - s:

Don’t add the user site-packages directory to **sys.path**.

* - S:

Disable the import of the module **site** and site-dependent manipulations of sys.path. Also disable these manipulation if **site** is explicitly imported later.

* - u:

Force the stdout and stderr streams to be unbuffered. This option has no effect on the stdin stream. Alternative to this is PYTHONUNBUFFERED.

* - V:

Print a message each time a module is initialized and show the path or file from which it is loaded.

* - [W:](file:///W:/)

Warning control. Python print warning message to **sys.stderr.**

Warning message format as follows,

[file : line : category](../../../../../../line:category) : message.

By default, each warning is print for each source line where it occurs. This option controls how often the warning messages are printed.

Multiple – W options may be given; when a warning matches more than one option. The action for the last matching option is performed.

Warnings can be controlled by using the PYTHONWARNINGS environment variable and from within the python program using the **warnings** module.

The simplest settings apply a particular action unconditionally to all warnings emitted by a process (even those that are otherwise ignored by default):

-Wdefault *# Warn once per call location*

-Werror *# Convert to exceptions*

-Walways *# Warn every time*

-Wmodule *# Warn once per calling module*

-Wonce *# Warn once per Python process*

-Wignore *# Never warn*

* - x:

Skip the first step from the source file for non-Unix specific forms of #!cmd. This is intended for a DOS specific hack only.

* - X:

Defined for various of implementation-specific options. Cpython currently define the some possible values.

* - J:

Reserved for use of Jpython.

# Environment Variables:

Following Environment variable influence python’s behavior. They are processed before command-line switches other than -E or -I. It is customary that command-line switches override environmental variables where there is a conflict.

# Simulating Python In o Interpreter: