Additional resources

Here is a list of resources that may be helpful as you continue your learning journey.

Learn more about Python data structures (Python documentation) on the Python website: [Python.org - Data structures](https://docs.python.org/3/tutorial/datastructures.html)

Explore common Python data structures at the Real Python website: [Real Python - Data structures](https://realpython.com/python-data-structures)

# Additional resources

The following resources will be helpful as additional references in dealing with different concepts related to the topics you have covered in this module.

Learn more about exceptions and errors in Python on the Python website: [Exceptions and Errors in Python - Python docs](https://docs.python.org/3/library/exceptions.html)

Check out the PyNative website to learn more about file handling in Python: [File handling in Python](https://pynative.com/python/file-handling/)

# Additional resources

The following resources will be helpful as additional references in dealing with different concepts related to the topics you have covered in this module.

* [Programming styles in Python](https://newrelic.com/blog/nerd-life/python-programming-styles)
* [Different types of algorithms used in Python](https://www.thetechplatform.com/post/different-types-of-algorithms-in-data-structure)
* [Introduction to Big-O notation](https://dev.to/sarah_chima/the-big-o-notation-an-introduction-34f7)

# Additional resources on functional programming

The following resources will be helpful as additional references in dealing with different concepts related to the topics you have covered in this module.

* [Python Map, reduce and list comprehension](https://www.knowledgehut.com/blog/programming/python-map-list-comprehension)
* [Recursion in Python](https://realpython.com/python-recursion/)
* [Functional Programming in Python](https://stackabuse.com/functional-programming-in-python/)

# Additional resources on OOP

The following resources will be helpful as additional references in dealing with different concepts related to the topics you have covered in this lesson.

* [OOP Principles](https://www.geeksforgeeks.org/python-oops-concepts/)
* [In-depth understanding of MRO](https://www.python.org/download/releases/2.3/mro/)
* [OOP Principles/ Classes and objects](https://realpython.com/python3-object-oriented-programming/)

# Additional resources on modules

The following resources will be helpful as additional references in dealing with different concepts related to the topics you have covered in this lesson.

* [OOP Principles](https://www.geeksforgeeks.org/python-oops-concepts/)
* [In-depth understanding of MRO](https://www.python.org/download/releases/2.3/mro/)
* [OOP Principles/ Classes and objects](https://realpython.com/python3-object-oriented-programming/)

# Additional Resources on Packages

The following resources will be helpful as additional references in dealing with different concepts related to the topics you have covered in this module:

* [Popular packages in Python](https://www.netsolutions.com/insights/top-10-python-frameworks-for-web-development-in-2019/)
* [Popular Python packages for web development](https://www.netsolutions.com/insights/top-10-python-frameworks-for-web-development-in-2019/)
* [ML and AI libraries in Python](https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c)
* [Data Science libraries in Python](https://www.dataquest.io/blog/15-python-libraries-for-data-science/)

## Flags used

For example, -v is the flag:

**python3 -m pytest abc.py -v**

Some other flag options are:

1

2

3

4

5

6

7

8

-v for verbose

-q quiet mode

-s allows the print statement inside the functions to be executed

-x is to flag the tests to stop execution after first failure

-m is used to mark a specific function

-k is a flag for searching and running tests with a specific keyword

--tb is to disable the traceback code of errors

--maxfail n specifies maximum number of test fails allowed





## Tips

* The rule of thumb is that the assert statement looks for a Boolean result. You can use in, not in, is, <, >, other than == to check Boolean values.
* You can add multiple assert statements inside a single test function.

## Additional reading

### Fixtures

Fixtures are a type of function that is applied to functions to be tested. These functions must run before that test is executed. The purpose of fixtures is to supply data from multiple sources including URLs and databases to the test before running the test. Fixtures are used in cases where code repeats initialization.

Format:

**@pytest.fixture**

### Markers

Markers are used to 'mark' specific functions to be executed by letting users create special names. There are many built-in markers such as xfail, xpass, skip and so on.

They follow a format such as:

**@pytest.mark.<markername>**

For example:

**@pytest.mark.alpha**

Running the specific marked test in the command line can be done with the following command:

**pytest -m <markername> -v**

Which will be as follows for a marker called alpha.

**pytest -m alpha -v**

# Additional resources

The following resources will be helpful as additional references in dealing with different concepts related to the topics we have covered in this module.

* [Test-Driven Development](https://testdriven.io/blog/modern-tdd/)
* [Test-driven Development with PyTest](https://stackabuse.com/test-driven-development-with-pytest/)
* [PyTest Official website](https://docs.pytest.org/en/7.1.x/)
* [Test automation packages in Python](https://www.geeksforgeeks.org/best-python-modules-for-automation/)