Python, Snowflake and other functions

Contents

[Python 4](#_Toc88639763)

[Pypi - The Python Package Index 4](#_Toc88639764)

[Install 4](#_Toc88639765)

[*Python* 4](#_Toc88639766)

[What is PIP in python? 4](#_Toc88639767)

[To check the version of pip 4](#_Toc88639768)

[Python Module Index 4](#_Toc88639769)

[Installing Packages 4](#_Toc88639770)

[Python 3.10.0 documentation 5](#_Toc88639771)

[Snowflake Connector for Python 5](#_Toc88639772)

[Installing the Python Connector 5](#_Toc88639773)

[Using the Python Connector 5](#_Toc88639774)

[Using Pandas DataFrames with the Python Connector 5](#_Toc88639775)

[Using the Snowflake SQLAlchemy Toolkit with the Python Connector 5](#_Toc88639776)

[Python Connector API 5](#_Toc88639777)

[Dependency Management Policy for the Python Connector 5](#_Toc88639778)

[Getting Started with Python 5](#_Toc88639779)

[hello.py 5](#_Toc88639780)

[pip install -U ipykernel 5](#_Toc88639781)

[NumPy 5](#_Toc88639782)

[Matplotlib: Python plotting — Matplotlib 3.4.3 documentation 5](#_Toc88639783)

[strandardplot.py 5](#_Toc88639784)

[Python: Start REPL Terminal 6](#_Toc88639785)

[Initialize the debugger, press F5 6](#_Toc88639786)

[Jupyter notebooks 8](#_Toc88639787)

[Data Serialization 8](#_Toc88639788)

[References 9](#_Toc88639789)

[Reading and Writing CSV Files in Python 10](#_Toc88639790)

[Python’s Built-in CSV Library 10](#_Toc88639791)

[OrderedDict vs dict in Python: The Right Tool for the Job 11](#_Toc88639792)

[Dictionaries 11](#_Toc88639793)

[Pandas 15](#_Toc88639794)

[Python With statement 15](#_Toc88639795)

[Python Open / Close Statement 16](#_Toc88639796)

[Special Methods 18](#_Toc88639797)

[Print Statement 19](#_Toc88639798)

[Import Statement and Python Modules / Libraries 19](#_Toc88639799)

[PIP statement and packages 20](#_Toc88639800)

[References 22](#_Toc88639801)

# Python

<https://www.python.org/psf/>

## Pypi - The Python Package Index

|  |  |
| --- | --- |
| * The Python Package Index (PyPI) is a repository of software for the Python programming language. * PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages](https://packaging.python.org/installing/). * Package authors use PyPI to distribute their software. [Learn how to package your Python code for PyPI](https://packaging.python.org/tutorials/packaging-projects/).   <https://pypi.org/> | **Installing Packages**   * “package” in this context is being used to describe a bundle of software to be installed (i.e. as a synonym for a distribution). * It does not to refer to the kind of package that you import in your Python source code (i.e. a container of modules). * It is common in the Python community to refer to a distribution using the term “package”. * Using the term “distribution” is often not preferred, because it can easily be confused with a Linux distribution, or another larger software distribution like Python itself. * <https://packaging.python.org/tutorials/installing-packages/> |
| <https://pypi.org/help/> |  |

# Install

Install Python - dbt is a Python module distributed on pypi, and can be installed via pip

# Python

## What is PIP in python?

Pip is the standard package manager for Python. It allows us to install and manage additional packages that are not part of the Python standard library.

Pip is an essential tool that comes along with python (after version 3.4 and 2.7.9) to install new packages. These packages are fetched from the PyPi’s repositories and are automatically integrated with your Python.

### To check the version of pip

> python -m pip version

# [Python Module Index](https://docs.python.org/3/py-modindex.html)

## [Installing Packages](https://packaging.python.org/tutorials/installing-packages/)

# [Python 3.10.0 documentation](https://docs.python.org/3/index.html)

# [Snowflake Connector for Python](https://docs.snowflake.com/en/user-guide/python-connector.html)

## [Installing the Python Connector](https://docs.snowflake.com/en/user-guide/python-connector-install.html)

## [Using the Python Connector](https://docs.snowflake.com/en/user-guide/python-connector-example.html)

## [Using Pandas DataFrames with the Python Connector](https://docs.snowflake.com/en/user-guide/python-connector-pandas.html)

## [Using the Snowflake SQLAlchemy Toolkit with the Python Connector](https://docs.snowflake.com/en/user-guide/sqlalchemy.html)

## [Python Connector API](https://docs.snowflake.com/en/user-guide/python-connector-api.html)

## [Dependency Management Policy for the Python Connector](https://docs.snowflake.com/en/user-guide/python-connector-dependencies.html)

# Getting Started with Python

PS> py -3 –version

## hello.py

msg = "Hello World"

print(msg)

PS C:\python> python.exe hello.py

PS C:\Python> python hello.py

## pip install -U ipykernel

>> C:/Users/frankt/AppData/Local/Programs/Python/Python39/python.exe -m

pip install -U ipykernel

## [NumPy](https://numpy.org/doc/stable/user/whatisnumpy.html)

## [Matplotlib: Python plotting — Matplotlib 3.4.3 documentation](https://matplotlib.org/)

## strandardplot.py

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0, 20, 100)  # Create a list of evenly-spaced numbers over the range

plt.plot(x, np.sin(x))       # Plot the sine of each x point

plt.show()                   # Display the plot

PS> python -m pip install matplotlib

PS> python -m pip install numpy

PS> C:\Users\frankt\AppData\Local\Programs\Python\Python39\python.exe -m pip install --upgrade pip

PS C:\python> python.exe strandardplot.py

## **Python: Start REPL Terminal**

|  |  |
| --- | --- |
| From the Command Palette (Ctrl+Shift+P), select the **Python: Start REPL** command to open a REPL terminal for the currently selected Python interpreter. In the REPL, you can then enter and run lines of code one at a time. |  |
|  |  |

## Initialize the debugger, press F5

VS Code uses JSON files for all of its various configurations; launch.json is the standard name for a file containing debugging configurations.

These different configurations are fully explained in [Debugging configurations](https://code.visualstudio.com/docs/python/debugging); for now, just select **Python File**, which is the configuration that runs the current file shown in the editor using the currently selected Python interpreter.

Debugging toolbar

A debug toolbar appears along the top with the following commands from left to right: continue (F5),

* step over (F10),
* step into (F11),
* step out (Shift+F11),
* restart (Ctrl+Shift+F5), and
* stop (Shift+F5).

# Jupyter notebooks

|  |  |
| --- | --- |
| The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.  Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. | Jupyter Notebook provides you with an easy-to-use, interactive data science environment across many programming languages that doesn't only work as an IDE, but also as a presentation or education tool. It's perfect for those who are just starting out with data science! |
| Inspired by scientific programs like Mathematica or Sage, the Notebook offers a modern and powerful web interface to Python. Compared to the original IPython terminal, the Notebook offers a more convenient text editor, the possibility to write rich text, and improved graphical capabilities. | IPython continued to exist as a Python shell and kernel for **Jupyter**, but the notebook interface and other language-agnostic parts of IPython were moved under the Jupyter name. Jupyter is language agnostic and its name is a reference to core programming languages supported by Jupyter, which are Julia, Python, and R. |

Ipykernel

|  |  |
| --- | --- |
| **Definition**  A powerful interactive Python shell and a Jupyter kernel to work with Python code in Jupyter notebooks and other interactive frontends. | **What is Ipykernel package?**  This package provides the IPython kernel for Jupyter. |
|  |  |

# Data Serialization

|  |  |
| --- | --- |
| Definition  Serialization is the process of converting a data object—a combination of code and data represented within a region of data storage—into a series of bytes that saves the state of the object in an easily transmittable form. ... Serialization encompasses both the storage of the object and exchange of data.  Data serialization is the process of converting data objects present in complex data structures into a byte stream for storage, transfer and distribution purposes on physical devices. Computer systems may vary in their hardware architecture, OS , addressing mechanisms. | Serialization formats:  There are two groups of serialization format: text-based and binary-based. As the name suggests, a text-based serialization is the process of serializing an object in a human-readable format, while binary-based is in a not human-readable format. |
| Data Deserialization:  JSON is a format that encodes objects in a string. Serialization means to convert an object into that string, and deserialization is its inverse operation (convert string -> object). | Java Example:  To serialize an object means to convert its state to a byte stream so that the byte stream can be reverted back into a copy of the object. A Java object is serializable if its class or any of its superclasses implements either the java. Io  Python Example:  Serialization is the process of encoding the from naive datat type to JSON format. The Python module json converts a Python dictionary object into JSON object, and list and tuple are converted into JSON array, and int and float converted as JSON number, None converted as JSON null. |

# References

* [Getting Started with Python in VS Code](https://code.visualstudio.com/docs/python/python-tutorial)
* [Using Python environments in VS Code](https://code.visualstudio.com/docs/python/environments#_global-virtual-and-conda-environments)
* [Check the Version of the Python Interpreter](https://www.geeksforgeeks.org/check-the-version-of-the-python-interpreter/)
* [Virtualenv and venv: Python virtual environments explained](https://www.infoworld.com/article/3239675/virtualenv-and-venv-python-virtual-environments-explained.html)

To learn to build web apps with the Django and Flask frameworks, see the following tutorials:

* [Use Django in Visual Studio Code](https://code.visualstudio.com/docs/python/tutorial-django) – Web applications
* [Use Flask in Visual Studio Code](https://code.visualstudio.com/docs/python/tutorial-flask) – Web UI
* **[PYTHON](https://code.visualstudio.com/docs/python/python-tutorial" \l "python-articles)**
  + [**Tutorial**](https://code.visualstudio.com/docs/python/python-tutorial)
  + [Editing Code](https://code.visualstudio.com/docs/python/editing)
  + [Linting](https://code.visualstudio.com/docs/python/linting)
  + [Debugging](https://code.visualstudio.com/docs/python/debugging)
  + [Environments](https://code.visualstudio.com/docs/python/environments)
  + [Testing](https://code.visualstudio.com/docs/python/testing)
  + [Data Science](https://code.visualstudio.com/docs/python/data-science)
  + [Python Interactive](https://code.visualstudio.com/docs/python/jupyter-support-py)
  + [Django Tutorial](https://code.visualstudio.com/docs/python/tutorial-django)
  + [Flask Tutorial](https://code.visualstudio.com/docs/python/tutorial-flask)
  + [Create containers](https://code.visualstudio.com/docs/python/tutorial-create-containers)
  + [Python on Azure](https://code.visualstudio.com/docs/python/python-on-azure)
  + [Settings Reference](https://code.visualstudio.com/docs/python/settings-reference)

Source code encoding in UTF-8 vs. ASCII and EBCDIC

|  |  |
| --- | --- |
| **By default, Python source files are treated as encoded in UTF-8.**  UTF-8 encodes Unicode characters into a sequence of 8-bit bytes. ... By comparison, ASCII (American Standard Code for Information Interchange) includes 128 character codes. Eight-bit extensions of ASCII, (such as the commonly used Windows-ANSI codepage 1252 or ISO 8859-1 “Latin -1”) contain a maximum of 256 characters. | **Which is better ASCII or UTF-8?**  All characters in ASCII can be encoded using UTF-8 without an increase in storage (both requires a byte of storage). UTF-8 has the added benefit of character support beyond "ASCII-characters".  UTF-8 is an encoding system for Unicode. It can translate any Unicode character to a matching unique binary string, and can also translate the binary string back to a Unicode character. This is the meaning of “UTF”, or “Unicode Transformation Format.” |
| **What is EBCDIC used for?**  EBCDIC is an 8-bit character encoding widely used in IBM midrange and mainframe computers. This encoding was developed in 1963 and 1964. EBCDIC was developed to enhance the existing capabilities of binary-coded decimal code. | [The history of UTF-8 as told by Rob Pike](http://doc.cat-v.org/bell_labs/utf-8_history) |

# Reading and Writing CSV Files in Python

<https://realpython.com/python-csv/#reading-csv-files-with-csv>

## Python’s Built-in CSV Library

<https://docs.python.org/3/library/csv.html>

## OrderedDict vs dict in Python: The Right Tool for the Job

# Dictionaries

|  |  |
| --- | --- |
| Dictionaries are a cornerstone of Python. The language itself is built around dictionaries. Modules, classes, objects, globals(), locals(): all of these are dictionaries. | Unlike sequences, which are iterables that support element access using integer indices, dictionaries are indexed by keys. |
| An associative array, where arbitrary keys are mapped to values. The keys can be any object with \_\_hash\_\_() and \_\_eq\_\_() methods.  In Python 3.6 and beyond, the keys and values of a dictionary are iterated over in the same order in which they were created. However, this behavior may vary across different Python versions, and it depends on the dictionary’s history of insertions and deletions. | The keys in a dictionary are much like a set, which is a collection of hashable and unique objects. Because the objects need to be hashable, mutable objects can’t be used as dictionary keys. |
| * [How to Iterate Through a Dictionary in Python: The Basics](https://realpython.com/iterate-through-dictionary-python/#a-few-words-on-dictionaries) |  |
|  |  |

|  |  |
| --- | --- |
| Code | Output |
| **# Define dictionary**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  print(thisdict) | {'brand': 'Ford', 'model': 'Mustang', 'year': 1964} |
| **# Print the number of items in the dictionary**  print(len(thisdict)) | 3 |
| **# Print the data type**  print(type(thisdict)) | <class 'dict'> |
| **# String, int, boolean, and list data types:**  thisdict = {  "brand": "Ford",  "electric": False,  "year": 1964,  "colors": ["red", "white", "blue"]  }  print(thisdict) | {'brand': 'Ford', 'electric': False, 'year': 1964, 'colors': ['red', 'white', 'blue']} |
| **# Change values**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  thisdict["year"] = 2018 | {'brand': 'Ford', 'model': 'Mustang', 'year': 2018} |
| **# update values**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  thisdict.update({"year": 2020}) | {'brand': 'Ford', 'model': 'Mustang', 'year': 2020} |
| **# add items by adding new index key and assigning a value to it**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  thisdict["color"] = "red"  print(thisdict) | {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'} |
| **# add items by updating new index key and assigning a value to it**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  thisdict.update({"color": "red"}) | {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'} |
| **# remove dictionary item:**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  thisdict.pop("model")  print(thisdict) | {'brand': 'Ford', 'year': 1964} |
| **popitem() method removes the last inserted item:**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  thisdict.popitem()  print(thisdict) | {'brand': 'Ford', 'model': 'Mustang'} |
| **# del keyword removes the item with the specified key name:**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  del thisdict["model"]  print(thisdict) | {'brand': 'Ford', 'year': 1964} |
| **# del keyword can also delete the dictionary completely:**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  del thisdict  print(thisdict) #this will cause an error because "thisdict" no longer exists. |  |
| **# The clear() method empties the dictionary values:**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  thisdict.clear()  print(thisdict) | {} |
| **# loop through a dictionary by using a for loop. Print all key names:**  for x in thisdict:   print(x) | brand  model  year |
| **Print all values in the dictionary, one by one:**  for x in thisdict:  print(thisdict[x]) | Ford  Mustang  1964 |
| **# values() method to return values of a dictionary:**  for x in thisdict.values():  print(x) | Ford  Mustang  1964 |
| **# keys() method to return the keys of a dictionary:**  for x in thisdict.keys():  print(x) | brand  model  year |
| **Loop through both keys and values, by using the items():**  for x, y in thisdict.items():  print(x, y) | brand Ford  model Mustang  year 1964 |
| **#copy of a dictionary with the copy() method:**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  mydict = thisdict.copy()  print(mydict) |  |
| **copy of a dictionary with the dict() function:**  thisdict = {  "brand": "Ford",  "model": "Mustang",  "year": 1964  }  mydict = dict(thisdict)  print(mydict) |  |
| **Nested Dictionaries:**  myfamily = {   "child1" : {     "name" : "Emil",     "year" : 2004   },   "child2" : {     "name" : "Tobias",     "year" : 2007   },   "child3" : {     "name" : "Linus",     "year" : 2011   } } | {'child1': {'name': 'Emil', 'year': 2004},  'child2': {'name': 'Tobias', 'year': 2007},  'child3': {'name': 'Linus', 'year': 2011}} |
| **# Create three dictionaries, then create one dictionary that will contain the other three dictionaries:**  child1 = {   "name" : "Emil",   "year" : 2004 } child2 = {   "name" : "Tobias",   "year" : 2007 } child3 = {   "name" : "Linus",   "year" : 2011 }  myfamily = {   "child1" : child1,   "child2" : child2,   "child3" : child3 } | {'child1': {'name': 'Emil', 'year': 2004},  'child2': {'name': 'Tobias', 'year': 2007},  'child3': {'name': 'Linus', 'year': 2011}} |

* clear() Removes all the elements from the dictionary
* copy() Returns a copy of the dictionary
* fromkeys() Returns a dictionary with the specified keys and value
* get() Returns the value of the specified key
* items() Returns a list containing a tuple for each key value pair
* keys() Returns a list containing the dictionary's keys
* pop() Removes the element with the specified key
* popitem() Removes the last inserted key-value pair
* setdefault() Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
* update() Updates the dictionary with the specified key-value pairs
* values() Returns a list of all the values in the dictionary

# Pandas

|  |  |
| --- | --- |
| pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. | *pandas* aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. |
| When working with tabular data, such as data stored in spreadsheets or databases, pandas is the right tool for you. pandas will help you to explore, clean, and process your data. In pandas, a data table is called a DataFrame. | **Use Cases**  [Update column value of CSV in Python using Pandas](https://www.geeksforgeeks.org/update-column-value-of-csv-in-python/) |

<https://pandas.pydata.org/docs/getting_started/index.html>

# Python With statement

|  |  |
| --- | --- |
| with statement in Python is used in exception handling to make the code cleaner and much more readable. It simplifies the management of common resources like file streams. | Notice that unlike the first two implementations, there is no need to call file.close() when using with statement. The with statement itself ensures proper acquisition and release of resources. An exception during the file.write() call in the first implementation can prevent the file from closing properly which may introduce several bugs in the code, i.e. many changes in files do not go into effect until the file is properly closed. |
| # file handling  # 1) without using with statement  file = open('file\_path', 'w')  file.write('hello world !')  file.close()  # 2) without using with statement  file = open('file\_path', 'w')  try:  file.write('hello world')  finally:  file.close() | # using with statement  with open('file\_path', 'w') as file:      file.write('hello world !')  <https://www.geeksforgeeks.org/with-statement-in-python/> |

# Python Open / Close Statement

|  |  |
| --- | --- |
| inbuilt functions for creating, writing and reading files.  Opening a file refers to getting the file ready either for reading or for writing. This can be done using the open() function. This function returns a file object and takes two arguments, one that accepts the file name and another that accepts the mode(Access Mode). | File\_object = open(“File\_Name”, “Access\_Mode”)  **File\_Name:** It is the name of the file that needs to be opened.  **Access\_Mode:** Access modes govern the type of operations possible in the opened file. |
|  |  |

| ***Operation*** | ***Syntax*** | ***Description*** |
| --- | --- | --- |
| *Read Only* | *r* | *Open text file for reading only.* |
| *Read and Write* | *r+* | *Open the file for reading and writing.* |
| *Write Only* | *w* | *Open the file for writing.* |
| *Write and Read* | *w+* | *Open the file for reading and writing. Unlike “r+” is doesn’t raise an I/O error if file doesn’t exist.* |
| *Append Only* | *a* | *Open the file for writing and creates new file if it doesn’t exist. All additions are made at the end of the file and no existing data can be modified.* |
| *Append and Read* | *a+* | *Open the file for reading and writing and creates new file if it doesn’t exist. All additions are made at the end of the file and no existing data can be modified.* |

|  |  |  |
| --- | --- | --- |
| Input | Code | Output |
| Hello Geek!  This is a sample text file for the example. | # open the file using open() function  file = open("sample.txt")    # Reading from file  print(file.read()) | Hello Geek!  This is a sample text file for the example. |
| Hello Geek!  This is a sample text file for the example. | # open the file using open() function  file = open("sample.txt", 'a')    # Add content in the file  file.write(" This text has been newly appended on the sample file") | Hello Geek!  This is a sample text file for the example.  This text has been newly appended on the sample file |
| Hello Geek!  This is a sample text file for the example. | # open the file using open() function  file = open("sample.txt", 'w')    # Overwrite the file  file.write(" All content has been overwritten !") | All content has been overwritten ! |
|  | # open the file using open() function  file = open("sample.txt")    # Reading from file  print(file.read())    # closing the file  file.close() |  |
|  | # open the file using open() function  file = open("sample.txt")    # Reading from file  print(file.read())    # closing the file  file.close()    # Attempt to write in the file  file.write(" Attempt to write on a closed file !") | ValueError: I/O operation on closed file. |

## Special Methods

|  |  |
| --- | --- |
| Python’s dictionaries are mapping objects. Mapping Objects are container objects that supports arbitrary key lookups and implements the methods specified in the Mapping or MutableMapping abstract base classes. Examples include dict, collections.defaultdict, collections.OrderedDict and collections.Counter.  These methods are named using the naming convention of adding a double underscore at the beginning of and at the end of the method’s name. | For mappings (like dictionaries), .\_\_iter\_\_() should iterate over the keys. This means that if you put a dictionary directly into a for loop, Python will automatically call .\_\_iter\_\_() on that dictionary, and you’ll get an iterator over its keys: |
| >>> dir({})  ['\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', ... , '\_\_iter\_\_', ...] | >>> a\_dict = {'color': 'blue', 'fruit': 'apple', 'pet': 'dog'}  >>> for key in a\_dict:  ... print(key)  ...  color  fruit  pet |
|  | for key in a\_dict:  ... print(key, '->', a\_dict[key])  ...  color -> blue  fruit -> apple  pet -> dog |

# Print Statement

# Import Statement and Python Modules / Libraries

|  |  |
| --- | --- |
| The Python import statement imports code from one module into a program. You can import all the code from a module by specifying the import keyword followed by the module you want to import. import statements appear at the top of a Python file, beneath any comments that may exist.  User can import both packages, modules and libraries. | **What is the difference between modules and packages?**  A module is a file containing Python code. A package, however, is like a directory that holds sub-packages and modules.  Module and Library are often used interchangeably with “Python package” because packages can also contain modules and other packages (subpackages). However, it is often assumed that while a package is a collection of modules, a library is a collection of packages. |
| **Python Modules / Libraries**   * [Scikit-learn](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Scikit-learn) * [NuPIC](https://www.mygreatlearning.com/blog/open-source-python-libraries/#NuPIC) * [Ramp](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Ramp) * [NumPy](https://www.mygreatlearning.com/blog/open-source-python-libraries/#NumPy) * [Pipenv](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Pipenv) * [TensorFlow](https://www.mygreatlearning.com/blog/open-source-python-libraries/#TensorFlow) * [Bob](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Bob) * [PyTorch](https://www.mygreatlearning.com/blog/open-source-python-libraries/#PyTorch) * [PyBrain](https://www.mygreatlearning.com/blog/open-source-python-libraries/#PyTorch) * [MILK](https://www.mygreatlearning.com/blog/open-source-python-libraries/#MILK) * [Keras](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Keras) * [Dash](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Dash) * [Pandas](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Pandas) * [Scipy](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Scipy) * [Matplotlib](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Matplotlib) * [Theano](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Theano) * [SymPy](https://www.mygreatlearning.com/blog/open-source-python-libraries/#SymPy) * [Caffe2](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Caffe2) * [Seaborn](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Seaborn) * [Hebel](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Hebel) * [Chainer](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Chainer) * [OpenCV Python](https://www.mygreatlearning.com/blog/open-source-python-libraries/#OpenCV) * [Theano](https://www.mygreatlearning.com/blog/open-source-python-libraries/#th2) * [NLTK](https://www.mygreatlearning.com/blog/open-source-python-libraries/#NLTK) * [SQLAlchemy](https://www.mygreatlearning.com/blog/open-source-python-libraries/#SQLAlchemy) * [Bokeh](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Bokeh) * [Important Python Libraries for Data Science](https://www.mygreatlearning.com/blog/open-source-python-libraries/#Imp) * [Python Standard Library](https://docs.python.org/3/library/) | **PyPi** - <https://pypi.org/>  **Python** - <https://docs.python.org/3/contents.html>  [**json**](https://docs.python.org/3/library/json.html?highlight=json#module-json) - encoder and decoder  [**os**](https://docs.python.org/3/library/os.html) - module provides a portable way of using operating system dependent functionality.  [**io**](https://docs.python.org/3/library/io.html) - Python’s main facilities for dealing with various types of I/O  **pandas** as pd  **numpy** as np  fdatetime import datetime  **boto3** - **Boto3** is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python  **openpyxl** - Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files  [**gc**](https://docs.python.org/3/library/gc.html) – garbage collector  **re** - regular expression matching operations |

# PIP statement and packages

|  |  |
| --- | --- |
| It’s important to note that the term “package” in this context is being used to describe a bundle of software to be installed (i.e. as a synonym for a distribution). It does not to refer to the kind of package that you import in your Python source code (i.e. a container of modules). It is common in the Python community to refer to a distribution using the term “package”. Using the term “distribution” is often not preferred, because it can easily be confused with a Linux distribution, or another larger software distribution like Python itself. | Creating Virtual Environments  Python “Virtual Environments” allow Python packages to be installed in an isolated location for a particular application, rather than being installed globally. If you are looking to safely install global command line tools, see Installing stand alone command line tools. |
|  |  |

# References

* <https://docs.python.org/3.9/tutorial/>
* with statement in Python - <https://www.geeksforgeeks.org/with-statement-in-python/>
* [34 Open-Source Python Libraries You Should Know About](https://www.mygreatlearning.com/blog/open-source-python-libraries/)
* [Difference Between Python Modules, Packages, Libraries, and Frameworks](https://learnpython.com/blog/python-modules-packages-libraries-frameworks/)