## Module 14 - Python Modules

Contents

[Module 14 - Python Modules 1](#_Toc216013777)

[Modules in Python 1](#_Toc216013778)

[Popular Python Modules 1](#_Toc216013779)

[Types of Modules 3](#_Toc216013780)

[Importing Modules 3](#_Toc216013781)

[Using the dir() Function 4](#_Toc216013782)

[Renaming Modules 5](#_Toc216013783)

[Creating a Module 5](#_Toc216013784)

[Python Module Search Path 5](#_Toc216013785)

[Reloading a Module 6](#_Toc216013786)

[Packages 6](#_Toc216013787)

[Importing a Package 6](#_Toc216013788)

[Examples 7](#_Toc216013789)

[Datetime Module 7](#_Toc216013790)

[User-Defined Module ‘Converter’ 7](#_Toc216013791)

[Creating Modules with Variables 8](#_Toc216013792)

[Creating Modules with Variables Example 2 8](#_Toc216013793)

[Datetime Module Example 2 8](#_Toc216013794)

[Random Module Example 9](#_Toc216013795)

[Reference and Additional Materials: 10](#_Toc216013796)

## Modules in Python

One of the biggest advantages of Python is its library support, which includes many built-in modules. These [modules](https://www.w3schools.com/python/python_modules.asp) are files that contain Python code, often made of functions and tools designed to perform specific tasks. You can think of a module as a code library. It is a reusable file that you can include in your program to make your work easier and more efficient. Instead of writing the same code repeatedly, you can use modules to save time and effort.

Using modules helps male your code more organized and easier to read. It also improves how your code is structured, making it simpler to maintain in the future. By grouping related functions and logic into a module, you create a cleaner and more understandable structure. Python [modules](https://docs.python.org/3/tutorial/modules.html) support good coding practices through reusability, efficiency, and clarity.

## Popular Python Modules

Several Python modules remain popular and widely used across various fields due to their reliability, efficiency, and versatility. Pandas and NumPy are widely used for data manipulation and numerical computations. Matplotlib, Seaborn, and Plotly are popular for data visualization. They offer a range of static, interactive, and clean charting options. Scikit-learn is a reliable library in machine learning. It is often used for classical algorithms and model evaluation. These are just popular examples and there are [many more](https://www.datacamp.com/blog/top-python-libraries-for-data-science) out there to explore for yourself.

## 

## Types of Modules

There are two main types of modules, [built-in modules](https://www.w3schools.com/python/gloss_python_module_built-in.asp) and user-defined modules.

Built-in modules are ones that come pre-installed with Python. These [modules](https://docs.python.org/3/library/functions.html) provide ready-to-use tools for common tasks. For example, the math module includes functions for mathematical operations, random helps generate random numbers, and datetime allows you to work with dates and times

[User-defined](https://www.digitalocean.com/community/tutorials/how-to-write-modules-in-python-3) modules are modules that you create yourself. These are just regular Python files that contain functions, classes, or variables you want to reuse in other programs. For instance, if you write a set of functions to clean and analyze data in one project, you can save them in a file and import them into another project when needed.

## Importing Modules

To use a module, you use the [import](https://www.geeksforgeeks.org/python-modules/#:~:text=Python%20Import%20From%20Module) keyword.

import math

When the interpreter finds an import statement, it imports the module presented in a search path.

Another way of importing modules is by importing specific functions. This comes most in handy when you know exactly what you need from the module and what you need it for.

from math import sqrt, pi

print(sqrt(25))

print(pi)

If you want to import all functions and attributes from a specific module, instead of writing all the functions names, you can import them using an asterisk.

from math import \*

print(pow(4,2))

print(factorial(5))

print(pi\*3)

print(sqrt(100))

You can also import multiple modules at once:

import math, random

print(math.factorial(5))

print(random.randint(10, 20))

## Using the dir() Function

If you are unsure what functions or variables a module has, use the [dir() function](https://www.w3schools.com/python/ref_func_dir.asp) to list all the attributes. When you use dir(), it returns a list of names that are currently defined in that context, including functions, variables, and special methods (those with double underscores, like \_\_init\_\_ or \_\_doc\_\_). This function is especially useful when you're working with a module or object that you’re not fully familiar with. It allows you to see what functionality is available without needing to consult external documentation. It's also helpful for debugging or understanding what has been defined in your current program's namespace.

import math

print(dir(math))

This will show you a list of all functions and constants available in the math module.

## Renaming Modules

Sometimes, module names can be long or conflict with existing variables in your code. To avoid this and improve readability, you can [rename a module](https://www.datacamp.com/tutorial/modules-in-python#:~:text=Renaming%20Modules%20in%20Python) using the [as](https://www.geeksforgeeks.org/python-as-keyword/) keyword. For example, import numpy as np is a common way to import the NumPy library. This lets you refer to the module using the shorter alias np, making your code cleaner and easier to type. Here’s another example using the math module:

import math as m

print(m.pi)

## Creating a Module

[Creating](https://youtu.be/5SGqHlQTxLA?si=Zfyv1TBGWCAjmRJl) your own module is simple. All you need to do is save your Python code in a file with a .py extension. For example, you can write a file named math\_tools.py with some custom functions. Once saved, this file becomes a module and can be imported into other scripts using import math\_tools. This is especially useful for projects that require the same code in multiple files or for sharing reusable code with others.

## Python Module Search Path

When you import a module in Python, the interpreter follows a specific order to locate it. First, it checks if the module is a built-in module that comes with Python. If the module name is not found among the built-ins, Python then looks through a list of directories defined by a special variable called [sys.path](http://cs.python.org/3/library/sys_path_init.html). This list is initialized based on an environment variable named PYTHONPATH. The PYTHONPATH variable works similarly to the PATH variable in Unix systems separated list of directory paths where Python should look for modules. If PYTHONPATH is not set or the desired module is not found in those directories, Python continues searching in the default installation paths. One common default directory on Unix-based systems is /usr/local/lib/python. This entire search process ensures that Python can locate and load the appropriate module when you use an import statement.

import sys

print(sys.path)

## Reloading a Module

Sometimes, you may update a module but Python won’t recognize the changes right away. In that case, you can use the importlib module to [reload](https://www.geeksforgeeks.org/reloading-modules-python/) it. For example, import importlib followed by importlib.reload(module\_name) forces Python to re-import the module and recognize any updates made to its code. This is helpful during debugging and development.

import time

from importlib import reload

# load 1st time

import cities

#time.sleep(20)

print(cities.cities\_list)

# reload

reload(cities)

print(cities.cities\_list)

## Packages

A [package](https://docs.python.org/3/tutorial/modules.html#:~:text=vars%27%2C%0A%20%27zip%27%5D-,6.4.%20Packages,-%C2%B6) in Python is a way of organizing related modules into a directory structure. It is simply a folder that contains multiple module files and a special \_\_init\_\_.py file, which tells Python that the folder is a package. [Packages](https://www.geeksforgeeks.org/python-packages/) help structure your codebase for large projects by grouping similar functionality together. For example, a package named analytics might contain modules like statistics.py, charts.py, and data\_cleaning.py.

## Importing a Package

To [import a package](https://youtu.be/ItJhkjS6QiU?si=NIswyEyZBQN7N6lz), you use the import statement just like with modules. For example, if you have a package named analytics with a module called charts, you can import it using import analytics.charts. You can also import specific functions or classes from a package’s module with the from keyword, like from analytics.charts import plot\_bar.

## Examples

### Datetime Module

The [datetime](https://www.w3schools.com/python/python_datetime.asp) module in Python provides classes and functions for working with dates, times, and timestamps. It allows you to easily create, manipulate, and format date and time values in your programs.

import datetime

d= datetime.date(2023,3,7) # don't need to pass leading zeros

tday=datetime.date.today()

print(d)

print(tday)

print(tday.day)

print(tday.year)

print(tday.month)

print(tday.weekday()) # Monday 0 and Sunday 6

print(tday.isoweekday()) # Monday 1 and sunday 7

### User-Defined Module ‘Converter’

A python file has been created with the name converter.py which has the following code:

def lbs\_to\_kg(weight):

return weight \*0.45

def kg\_to\_lbs(weight):

return weight / 0.45

It has been called in the below session.

import converter

from converter import lbs\_to\_kg

weight\_kg = int(input("enter weight in kg: "))

print(f'weight in lb is: {converter.kg\_to\_lbs(weight\_kg)}')

weight\_lb = int(input("enter weight in lb: "))

print(f'weight in kg is: {converter.lbs\_to\_kg(weight\_lb)}')

### Creating Modules with Variables

In Python, the module contains Python code like classes, functions, methods, but it also has variables. A variable can list, tuple, dict, etc.

import cities

city = cities.cities\_list[2]

print("Accessing 1st city:", city)

# Get all cities

cities = cities.cities\_list

print("Accessing All cities :", cities)

### Creating Modules with Variables Example 2

import cities1

city = cities1.cities\_list

print("Accessing all", city)

city=cities1.cities\_list[1]

print("Accessing the 2nd city", city)

### Datetime Module Example 2

from datetime import datetime

from datetime import date

import time

today = date.today()

def user\_birthday():

year = int(input('When is your birthday? [YY] '))

month = int(input('When is your birthday? [MM] '))

day = int(input('When is your birthday? [DD] '))

birthday = datetime(year,month,day)

return birthday

def age(birthdate):

today = date.today()

age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.day))

return age

bday = user\_birthday()

age=age(bday)

print(age)

### Random Module Example

import random

for i in range(5):

print(random.random())

for i in range(5):

print(random.randint(1000, 2000))

names =['Meena','Vimal','Vevin', 'Meki']

print(random.choice(names))

## Reference and Additional Materials:

<https://www.datacamp.com/blog/top-python-libraries-for-data-science>

<https://youtu.be/ItJhkjS6QiU?si=NIswyEyZBQN7N6lz>

<https://www.digitalocean.com/community/tutorials/how-to-write-modules-in-python-3>

<https://youtu.be/5SGqHlQTxLA?si=Zfyv1TBGWCAjmRJl>

<https://www.datacamp.com/tutorial/modules-in-python#:~:text=Renaming%20Modules%20in%20Python>