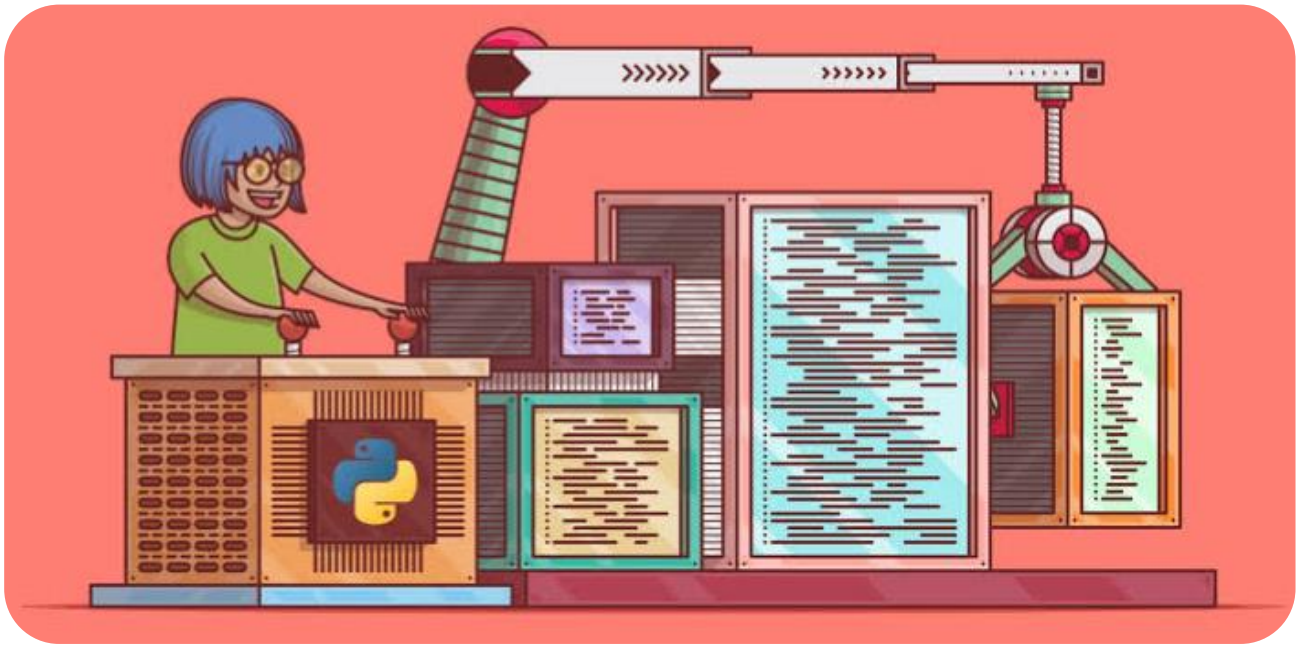


Python Modules

- A module in Python programming allows us to logically organize the Python code. A module is a single source code file. The module in Python has the .py file extension. The name of the module will be the name of the file.
- A Python module can be defined as a Python program file which contains a Python code including Python functions, class or variables. In other words, we can say that our Python code file saved with the extension (.py) is treated as the module.

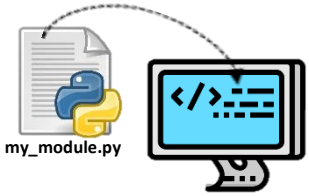
Types of Modules

1. Built-in Modules

These modules are part of Python's core installation.

- **Functionalities:** They offer essential operations like mathematical calculations, input/output handling, and system interactions.
- **Example:** `math`, `os`, `datetime`,

User-defined Modules: Created by programmers for specific project needs, promoting code organization and reusability. Can be imported using an absolute import path (e.g. `import my_module`).

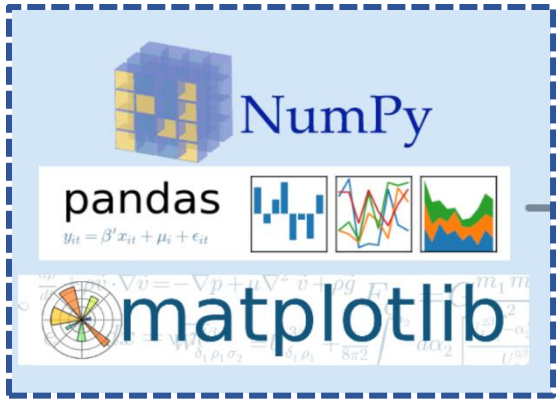


2. External Modules

- **Functionalities:** These modules extend Python's capabilities beyond the standard library.
- **Categories:**

Third Party modules: Developed by the community, providing specialized tools and libraries, they must be installed separately using tools like `pip` or `conda`.

Example: `numpy`, `pandas`, `matplotlib`.



Python Modules, Packages, and Libraries: A Breakdown

1. Modules

- **Single Python file:** Contains functions, classes, and variables.
- **Purpose:** Organize code into reusable units.
- **Example:** math module provides mathematical functions like sqrt, sin, etc.

2. Packages

- **Directory of modules:** Contains multiple modules related to a specific functionality.
- **Hierarchical structure:** Can contain sub-packages for further organization.
- **Example:** numpy package provides numerical operations and array manipulation.

3. Libraries:

- **Collection of modules and packages:** Offers a broader set of functionalities.
- **Reusable code:** Provides pre-written code for common tasks.
- **Distribution:** Often distributed as standalone packages.
- **Examples:** pandas, matplotlib, scikit-learn.

```
Library

|--Package1
    |-- Module1
    |-- Module2

|--Package2
    |-- Module3
    |-- Module4
```

Feature	Module	Package	Library
Structure	Single file	Directory of modules	Collection of modules and packages
Purpose	Organize code	Organize related modules	Provide reusable functionalities
Example	math	numpy	pandas

Built-in Module (Math Module)

```
import math

1. floor() : This function returns the smallest integral value smaller than the
number. If number is already integer, same number is returned.ascii

print(math.floor(2.3))     # print's 2


2. ceil() : This function returns the smallest integral value greater than the
number. If number is already integer, same number is returned.ascii

print(math.ceil(2.3))     # print's 3


3. cos() : This function returns the cosine of value passed as argument. The
value passed in this function should be in radians.

print(math.cos(math.radians(180)))    # print's -1.0
print(math.cos(math.radians(0)))      # print's 1.0



4. fabs() : This function will return an absolute or positive value

print(math.fabs(10))    # print's 10.0
print(math.fabs(-20))   # print's 20.0


5. factorial() : This function will return an absolute or positive value.

print(math.factorial(5)) # print's 120


6. sqrt() : The method sqrt() returns the square root of x for x>0.

print(math.sqrt(100))   # print's 10.0
print(math.sqrt(25))    # print's 5.0
print(math.sqrt(5))     # print's 2.23606797749979
```

Built-in Module (Random Module)

Python defines a set of functions that are used to generate or manipulate random numbers through the random module. List of all the functions defined in random module are:

1. choice() : Returns a random item from a list, tuple, or string.
Synatx: `random.choice(sequence)`
A sequence like a list, a tuple, a range of numbers etc.

Example:

```
import random

l1 = [10, 20, 30, 40, 50]
print(random.choice(l1))        # print's 30

string = "Python"
print(random.choice(string))    # print's t
```

2. randrange(): Returns a random number between the given range.
Syntax: randrange(start, end, step)

Where,
Start: Optiobnal. An integer specifying at which position to start. Default 0.
End: Required. An integer specifying at which position to end.
Step: Optional. An integer specifying incrementation. Default 1.

Example:

```
# use randrange() funcction to generate in range from 20 to 50.

print(random.randrange(1,100,2))        # print's 11
```

3. shuffle(): It is used to shuffle a sequence(list). Shuffling means changing the position of the elements of the sequence.
Syntax: `random.shuffle(sequence)`

Example:

```
list1 = [10, 20, 30, 40, 50]
print("Shuffling Number List: ")
random.shuffle(list1)

print(list1)        # print's [40, 50, 30, 10, 20]

list2 = ['A', 'B', 'C', 'D', 'E']
print("Shuffling character List: ")
random.shuffle(list2)

print(list2)        # print's ['D', 'A', 'B', 'E', 'C']
```

4. random(): Returns random numbers between 0.0 and 1.0
Syntax: `random.random()`

Example:

```
print(random.random())              # print's 0.2888164291491717
```

5. sample(): Return a list that contains any 3 of the items from a list.
Syntax: `random.sample(sequence, k)`

```
myList = ["Apple", "Banana", "Cherry", "Mango", "Pineapple"]

print(random.sample(myList, k=3))  # print's ['Cherry', 'Banana', 'Pineapple']
```

6. uniform(): The uniform() method returns a random floating number between the two specified numbers (both included).
Syntax: `random.uniform(20, 50)`

Example:

```
print(random.uniform(20,50))        # print's 23.284251087576386
```

Built-in Module (Datetime Module)

Python datetime module deals with date, times and times and time intervals. Date and datetime in Python are the objects, so when you manipulate them, you are actually manipulating objects and not string or timestamps. Whenever you manipulate dates or time, you need to import datetime function.

Example 1: To get current date

```
import datetime

ob = datetime.date.today()
print(ob)
```

```
OUTPUT:
2024-06-04
```

Example 2: To get current current date and time

```
import datetime

ob = datetime.datetime.date.now()
print(ob)
```

```
OUTPUT:
2024-06-04  15:26:37.237265
```

1. Date Class: When an object of this class is instantiated, it represents a date in the format YYYY-MM-DD
Synatx: class datetime.date(year, month, day)

Example:

```
from datetime import date

Today = date.today()
print("Current date =",Today)
print("Current Year =",Today.year)
print("Current Day =",Today.day)
```

```
OUTPUT:
Current date = 2024-02-03
Current Year = 2024
Current Day = 31
```

2. Time Class: Time object represents local time, independent of any day.

Example:

```
from datetime import time

time(hour = 0, minute = 0, second = 0)
a = time()
print("a=",a)

time(hour, minute and second)
b = time(15, 30, 56)
print("b=",b)
print("Hour =",b.hour)
print("Minute =",b.minute)
print("Second =",b.second)
print("Microsecond=",b.microsecond)

time(hour, minute and second)
c = time(hour = 15, minute = 30, second = 56)
print("c=",c)

time(hour, minute, second, microsecond)
d = time(15, 30, 56, 234566)
print("d=",d)
```

```
OUTPUT:
a = 00:00:00
b = 15:30:56
Hour = 15
Minute = 30
Second = 56
Microsecond= 0
c = 15:30:56
d = 15:30:56.234566
```

3. Datetime Class: Information on both date and time is contained in this class.

```
from datetime import datetime

today = datetime.now()
print("Current Date and Time is",today)
OUTPUT:
Current Date and Time is 2024-08-04 20:40:34.929845

datetime(year, month, day)
a = datetime(2021, 2, 6)
print(a)     # print's 2021-02-06 00:00:00


datetime(year, month, day, hour, minute, second, microsecond)
b = datetime(2021, 2, 6, 15, 30, 56, 342380)
print(b)     # print's 2021-02-06 15:30:56.342380
```

```
print("Year=",b.year)
print("Month=",b.month)
print("Hour=",b.hour)
print("Minute=",b.minute)
```

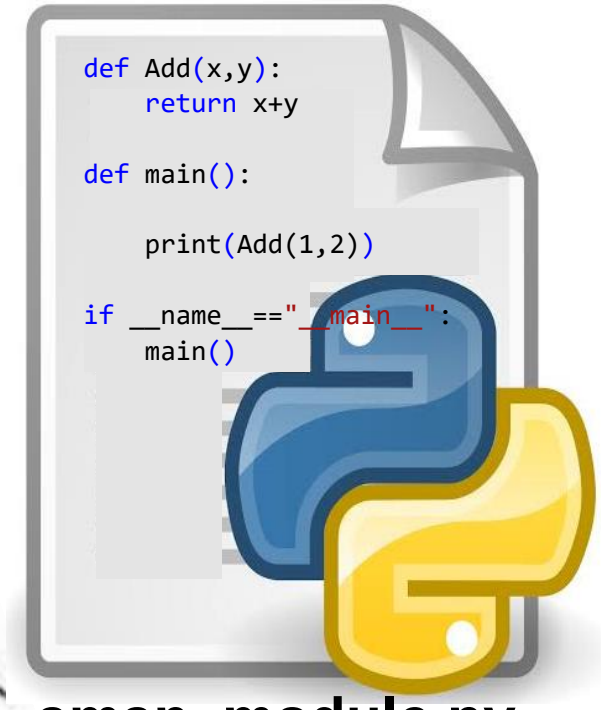
```
OUTPUT:
Year= 2021
Month= 2
Hour= 15
Minute= 30
```

4. Timedate Class: A timedate object represents the difference between two dates or times.

```
Difference between two Dates and Times

D1 = date.today()
D2 = date(year = 2004, month = 3, day = 30)

D3 = D1 - D2

print("D3 = ",D3)       # print's D3 =  7432 days, 0:00:00



T1 = datetime(year=2021, month = 2, day = 6, hour = 7, minute = 9, second = 33)
T2 = datetime(year=2020, month=12, day = 10, hour = 5, minute = 55, second = 13)

T3 = T1 - T2

print("T3 =",T3)        # print's T3 = 58 days, 1:14:20
```

User-defined Module

Any text file with the .py extension containing Python code is basically a module. Different Python objects such as functions, classes, variables, constants, etc., defined in one module can be made available to an interpreter session or another Python script by using the import statement.



aman_module.py

```
import aman_module

result = aman_module.Add(45,
12)
```

aman_package