Modules and Packages — essential for keeping your code **organized, reusable, and scalable**.

What is a Module?

A **module** is simply a (`.py`) file containing Python code — functions, classes, or variables — that you can **import and use** in another file.

Example:

math_tools.py

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
```

main.py

```
import math_tools

print(math_tools.add(5, 3))          # Output: 8
print(math_tools.subtract(10, 4))    # Output: 6
```

Use `from ... import` if you only want certain parts:

```
from math_tools import add

print(add(2, 2))  # Direct use without module name
```

Built-in Modules

Python has many powerful built-in modules:

Module	Use
`math`	Math functions
`random`	Random number generation
`datetime`	Working with dates and time
`os`	File & folder operations
`sys`	System-related operations

Examples:

```
import math
print(math.sqrt(16))      # 4.0

import random
print(random.randint(1, 10))  # Random number between 1-10
```

1. `math` (Mathematical Operations)

- `math.sqrt(x)` – Square root
- `math.pow(x, y)` – Power (x^y)
- `math.sin(x)`, `math.cos(x)`, `math.tan(x)` – Trigonometric functions
- `math.log(x)` – Natural logarithm
- `math.ceil(x)` – Round up
- `math.floor(x)` – Round down
- `math.pi`, `math.e` – Constants

2. `os` (Operating System Interactions)

- `os.getcwd()` – Current working directory
- `os.listdir(path)` – List files in a directory
- `os.mkdir(dir)` – Create a directory
- `os.remove(file)` – Delete a file
- `os.rename(old, new)` – Rename a file/dir
- `os.path.exists(path)` – Check if path exists

3. `sys` (System-Specific Functions)

- `sys.argv` – Command-line arguments
- `sys.exit()` – Exit Python
- `sys.version` – Python version
- `sys.path` – Module search path

4. `datetime` (Date & Time Handling)

- `datetime.datetime.now()` – Current date & time
- `datetime.date.today()` – Current date
- `datetime.timedelta(days=x)` – Time difference
- `datetime.strftime(format)` – Format date as string

5. `random` (Random Number Generation)

- `random.randint(a, b)` – Random integer between `a` and `b`
- `random.choice(seq)` – Random element from a sequence
- `random.shuffle(list)` – Shuffle a list
- `random.random()` – Float between `0.0` and `1.0`

What is a Package?

A **package** is a folder that contains **multiple modules** and a special file `__init__.py(can be empty or used for initialization).`

- A package contains all the files you need for a module.
- Modules are Python code libraries you can include in your project.

Creating a Package Structure

```
my_package/
│
├── __init__.py
├── math_utils.py
└── string_utils.py
```

math_utils.py

```
def square(x):
    return x * x
```

string_utils.py

```
def shout(text):
    return text.upper()
```

Using the Package in Your Code:

```
from my_package import math_utils, string_utils

print(math_utils.square(4))           # Output: 16
print(string_utils.shout("hello"))  # Output: HELLO
```

Installing External Packages

You can install third-party packages using:

- `pip install package_name`

Example:

```
pip install requests
```

Then use it :

```
import requests

response = requests.get("https://api.github.com")
print(response.status_code)
```

Summary

Concept	Meaning
Module	`.py` file with functions/classes
Package	Folder of modules with `__init__.py`
`import`	Used to bring in modules
`pip`	Tool to install external packages

Assignment

Create a folder `myutils` and complete the following:

Q1. Create `greet.py` in `myutils/`

- Function `say_hello(name)` → prints `Hello, <name>!`

Q2. Create `math_ops.py` in `myutils/`

- Function `square(n)`
- Function `cube(n)`

Q3. In `main.py`

- Import and use functions from both files.

Q4. Bonus – Use Built-in Module

- Import `datetime` and print today's date.