

Modules and Libraries

Day 13 - Python Basics

Shaida Muhammad

Wednesday, March 19, 2025

ShaidaSherpao@gmail.com

Agenda

Python Online Free Ramzan Course 2025 Taught by: Shaida Muhammad

	wriat are modules and libraries:	5	Creating custom packages (with
			initpy)

- Importing modules

 6 Installing third-party libraries
- The Python Standard LibraryHands-on practice
- 4 Creating custom modules

What are modules and libraries?

What are Modules and Libraries?

- Module: A file containing Python code (functions, classes, variables) that can be reused in other programs.
- **Library:** A collection of modules that provide specific functionality.
- Why use modules and libraries?
 - Reusability: Avoid rewriting code.
 - Organization: Break code into manageable parts.
 - Efficiency: Leverage pre-built solutions.



Importing Modules

Syntax:

import module_name

• Example:

```
import math
print(math.pi) # Output: 3.141592653589793
```

Importing Specific Functions:

```
from math import sqrt
print(sqrt(25)) # Output: 5.0
```

Aliasing Modules:

```
import math as m
print(m.sqrt(36))  # Output: 6.0
```

- Key Points:
 - O Use import to bring in entire modules.
 - O Use from ... import to bring in specific functions or classes.
 - O Use as to create aliases for modules.



The Python Standard Library

- Definition: A collection of modules that come preinstalled with Python.
- Common Modules:
 - math: Mathematical functions.
 - random: Random number generation.
 - datetime: Date and time manipulation.
 - os: Operating System interactions.
 - sys: System-specific parameters and functions.
- Example:

```
import random
print(random.randint(1, 10))
# Output: Random number between 1 and 10
```



Creating Custom Modules

- Steps:
 - O Create a .py file with Python code.
 - O Import the file in another script.
- Example:
 - O Create a file mymodule.py:

```
def greet(name):
    return f"Pa Khair Raaghly, {name}!"
```

O Import and use it:

```
import mymodule
print(mymodule.greet("Ali"))
# Output: Pa Khair Raaghly, Ali!
```

- Key Points:
 - O The module name is the file name (without .py).
 - O Use import to access functions, classes, or variables from the module.



Creating Custom Packages

- Definition: A package is a collection of modules organized in a directory.
- Structure:

```
mypackage/
    __init__.py
    module1.py
```

- __init__.py:
 - O Makes a directory a Python package.

module2.py

O Can be empty or contain initialization code.



Creating Custom Packages ...

- Example:
 - O Create a folder mypackage with the following files:
 - mypackage/__init__.py:

```
print("Initializing mypackage...")
```

mypackage/module1.py:

```
def greet(name):
```

```
return f"Starry Mashy, {name}!"
```

mypackage/module2.py:

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

```
return a + b
```



Creating Custom Packages ...

Python Online Free Ramzan Course 2025
Python Online Free Ramzan Muhammad
Taught by: Shaida Muhammad

O Import and use the package:

```
import mypackage.module1
import mypackage.module2
```

```
print(mypackage.module1.greet("Ali")) # Output:
Starry Mashy, Ali!
print(mypackage.module2.add(3, 5)) # Output: 8
```

- Using __all__ in __init__.py:
 - O Controls what gets imported with from mypackage import *.
 - O Example:

```
__all__ = ["module1"]
```

Installing Third-Party Libraries

- Using pip: Python's package installer.
- Syntax: pip install library_name
- Example: pip install requests
- Using a Third-Party Library:

```
import requests
response = requests.get("https://www.example.com")
print(response.status_code) # Output: 200 (if successful)
```

• Key Points:

- Use pip to install libraries from the Python Package Index (PyPI).
- Always check documentation for usage instructions.

Hands-On Practice

Task 1: Import and use the math module.

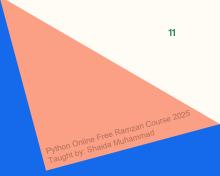
```
import math
print(math.sqrt(64))
```

 Task 2: Use the random module to generate a random number.

```
import random
print(random.randint(1, 100))
```

- Task 3: Create a custom module and import it.
 - O Create mymodule.py: def add(a, b): return a + b
 - O Use it:

```
import mymodule
print(mymodule.add(3, 5))
# Output: 8
```



- Task 4: Create a custom package and import it.
 - O Create a folder mypackage with __init__.py, module1.py, and module2.py.
 - O Import and use the package: import mypackage.module1 import mypackage.module2

```
print(mypackage.module1.greet("Ali"))
# Output: Starry Mashy, Ali!
print(mypackage.module2.add(3, 5)) # Output: 8
```

Task 5: Install and use the requests library.import requests

```
response = requests.get("https://www.example.com")
print(response.status_code)
# Output: 200 (if successful)
```

Recap

- Modules are reusable Python files.
- Libraries are collections of modules.
- Use import to bring in modules or specific functions.
- The Python Standard Library provides many useful modules.
- Create custom modules by writing Python code in .py files.
- Create custom packages by organizing modules in folders with __init__.py.
- Use pip to install third-party libraries.



Homework

1. Explore the Standard Library:

Use the datetime module to print today's date.

2. Create a Custom Module:

 Write a module with functions for basic math operations (add, subtract, multiply, divide).

3. Create a Custom Package:

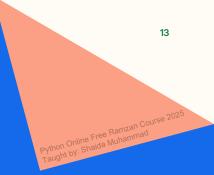
Organize multiple modules into a package and use __init__.py
 to initialize it.

4. Install Third-Party Libraries:

o Install the numpy, pandas, scikit-learn, and matplotlib.

5. Advanced Practice:

 Write a program that uses the os module to list files in a directory.



Modules and Libraries

Python Online Free Ramzan Course 2025
Paght by: Shalda Muhammad

Q&A

- Do you have any questions?
- Share your thoughts.

Modules and Libraries

Python Online Free Ramzan Course 2025 Taught by: Shaida Muhammad

Closing

Next class: Error Handling