"""
Module:
A module is python file.
(A file with a .py extension)

There are 2 types of modules
1. Built in modules
   Ex:
   math, random, os, re, datetime
2. User defined modules
   import <file_name>

How to use Modules in our application:
We use import statement to use the classes,
functions or variables defined in the module.

There are 4 types of imports

1. Module Name import
   Syntax:
   import <module_name>
   Usage:
   module_name.function_name()
2. Specific Import
   from module_name import function1, variable1, ...
   Usage:
   function()
3. Module Name Alias Import
   import module_name as <alias_name>
   Usage:
   alias_name.function_name()
4. Wild Import
   from module_name import *
   Usage:
   function_name()
   This is known as wild import

   We should try to avoid the wild import as it imports all the
   variables, functions and classes present in the module to the
   current namespace.

In Python, a module is a file containing Python definitions and statements. These files typically have a .py extension. Modules serve as a mechanism for organizing and reusing code, promoting modularity and making programs easier to manage and understand.
Key characteristics of Python modules:
Code Organization: Modules allow you to group related functions, classes, variables, and other Python objects into a single file. This improves code organization and readability, especially in larger projects.
Reusability: Once a module is created, its contents can be imported and used in other Python scripts or modules, eliminating the need to rewrite the same code multiple times.
Namespace: Each module has its own namespace, preventing naming conflicts between objects defined in different modules.
Types of Modules:
Built-in Modules: Python comes with a rich standard library containing numerous

built-in modules for various tasks (e.g., math for mathematical functions, os for operating system interaction, datetime for date and time handling).
Custom Modules: You can create your own modules by simply saving a Python file with a .py extension.
C Modules: Modules can also be written in C and dynamically loaded at runtime for performance-critical operations.
Using Modules:
To use the functionalities defined in a module, you need to import it using the import statement.
Python

```
import math
print(math.sqrt(25)) # Accessing a function from the 'math' module
```

You can also import specific components from a module using the from keyword:
Python

```
from math import sqrt
print(sqrt(36)) # Directly using the 'sqrt' function
```

When we import a module, python searches for the module in 3 places.
1. Built-in Modules
2. Directory Where python is installed.
3. Current Directory where the script is located.

Locating Python Modules
Whenever a module is imported in Python the interpreter looks for several locations. First, it will check for the built-in module, if not found then it looks for a list of directories defined in the sys.path. Python interpreter searches for the module in the following manner -

First, it searches for the module in the current directory.
If the module isn’t found in the current directory, Python then searches each directory in the shell variable PYTHONPATH. The PYTHONPATH is an environment variable, consisting of a list of directories.
If that also fails python checks the installation-dependent list of directories configured at the time Python is installed.

To add a different location, we use sys.path
sys - system module
sys.path - list of paths
We add the location to the sys.paths
"""

```
"""
# Import 1
import math

print(math.pi)
print(math.sqrt(100))
"""

"""
# Import 2
from math import pi, sqrt
```

```
print(pi)
print(sqrt(100))
"""

"""
# Import 3

import math as m

print(m.pi)
print(m.sqrt(100))
"""

"""
# Import 4

from math import *

print(pi)
print(sqrt(100))
"""


# User Defined Modules
import my_module1


print(my_module1.a)
my_module1.foo()

# import module2
import sys

print(sys.path)
sys.path.append("./Folder1")
print(sys.path)

import my_module2


print(my_module2.b)


# __all__ Keyword Usage in Modules
# Using Wild Import with __all__
from my_module3 import *

print(a)
# print(b)

foo()
# bar()


# Using Specific Import
from my_module3 import a, b, foo, bar

print(a)
```

```
print(b)

foo()
bar()

# Using Module Level Import
import my_module3

print(my_module3.a)
print(my_module3.b)

my_module3.foo()
my_module3.bar()

# import skjdslkjd


import my_module4
```