Creating and using modules in Python is a fundamental part of writing clean, maintainable code. A module is simply a Python file with a `.py` extension that contains definitions and statements, such as functions, classes, or variables. You can import these modules into other Python files to reuse the code.

Step-by-Step Guide to Creating and Using Python Modules

Step 1: Create a Python Module

First, create a Python file that will serve as your module. Let's call this file `mymodule.py`.

```
# mymodule.py

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

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

class MathOperations:
    def multiply(self, a, b):
        return a * b

    def divide(self, a, b):
        if b == 0:
            raise ValueError("Cannot divide by zero")
        return a / b
```

Step 2: Create a Python Script to Import the Module

Next, create another Python file where you will import and use the module you just created. Let's call this file `main.py`.

```
# main.py

# Importing the entire module
import mymodule

result_add = mymodule.add(5, 3)
result_subtract = mymodule.subtract(5, 3)
print(f"Addition: {result_add}")
print(f"Subtraction: {result_subtract}")

# Importing specific functions and classes from the module
from mymodule import MathOperations
```

```
math_ops = MathOperations()
result_multiply = math_ops.multiply(5, 3)
result_divide = math_ops.divide(10, 2)
print(f"Multiplication: {result_multiply}")
print(f"Division: {result_divide}")
```

Explanation

1. **Creating a Module (`mymodule.py`)**:
   - This file defines two functions, `add` and `subtract`, and a class `MathOperations` with methods for multiplication and division.
2. **Importing the Module (`main.py`)**:
   - You can import the entire module using `import mymodule` and then access its functions and classes using the module name (`mymodule.add`, `mymodule.subtract`).
   - Alternatively, you can import specific items from the module using `from mymodule import MathOperations` and use them directly.

Module Search Path

When you import a module, Python searches for the module in the following locations:

1. The directory containing the input script (or the current directory when no file is specified).
2. The list of directories contained in the `PYTHONPATH` environment variable.
3. The installation-dependent default path (e.g., `/usr/lib/python3.9`).

Creating a Package

A package is a way of organizing related modules into a directory hierarchy. To create a package, you need to create a directory and include an `__init__.py` file (which can be empty) in it.

Step 1: Create a Package Directory

Let's create a package named `mypackage`.

```
mypackage/
    __init__.py
    mymodule.py
```

Step 2: Use the Package in Your Script

Now, you can use the package in your `main.py` script.

```
# main.py

# Importing from the package
from mypackage import mymodule

result_add = mymodule.add(5, 3)
result_subtract = mymodule.subtract(5, 3)
print(f"Addition: {result_add}")
print(f"Subtraction: {result_subtract}")

from mypackage.mymodule import MathOperations

math_ops = MathOperations()
result_multiply = math_ops.multiply(5, 3)
result_divide = math_ops.divide(10, 2)
print(f"Multiplication: {result_multiply}")
print(f"Division: {result_divide}")
```

Explanation

1. **Package Structure**:
   - The directory `mypackage` contains an `__init__.py` file and the `mymodule.py` file. This structure tells Python that `mypackage` is a package.
2. **Importing from the Package**:
   - You can import the `mymodule` module from the `mypackage` package using `from mypackage import mymodule`.
   - You can also import specific classes or functions directly from the module within the package using `from mypackage.mymodule import MathOperations`.