Overview

In this lesson, you will learn how to organize programs that use **multiple functions across multiple files**, while preventing code from running when a file is imported as a module. You will learn how and why to use a `main` function, and how Python decides what code should run.

Important Information

As programs grow, they often use many different functions. There may be multiple modules that are being imported. Usually you want your program to start at one specifc point, but this can become more difficult as more and more code is added to your projects. This can be solved by structuring your project around a `main()` function.

Without structure, importing files can cause code to run when you do **not** want it to.

The Problem: Code Running on Import

By default, **Python runs every line of code in a file from top to bottom**. That includes `print()` statements, function calls, variable declaration and intialization, and any code not inside a function.

Example:

`helpers.py`

```
def greet():
    print("Hello!")

print("Helpers file running")
```

`main.py`

```
import helpers
```

When `main.py` runs, this prints:

```
Helpers file running
```

This happens even though `greet()` was never called. This is often **not what the programmer intends**.

The `main` Function

We want helper files to define functions. We only want code to run **when we tell it to**. We need to create one clear starting point for the program. This is done using a `main` function.

A common pattern in python is to put the main logic of a program inside a function called `main`.

For example:

```
def main():
    print("Program starting")
```

On its own, this function does nothing until it is called.

The `__name__` Check

Python keeps track of which file is being run directly. If a file is run directly, Python sets a special variable called `__name__` to `"__main__"`. If a file is imported, `__name__` is set to the file's name. You do **not** change `__name__`. Python does this automatically.

Because the file that is being run directly has the `__name__` variable set to `"__main__"`, it means that we can check to see if our file is being run or imported with the following block of code. When the file is run directly, we can call the `main()` function:

```
if __name__ == "__main__":
    main()
```

With this example, python knows to only run `main()` if the file is being run directly, and to **not** run `main()` if the file is an imported module,

Complete Example

`helpers.py`

```
def greet():
    print("Hello from helpers")

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

No code runs automatically when this file is imported.

`main.py`

```
from helpers import greet, add

def main():
    greet()
    result = add(3, 4)
    print(result)

if __name__ == "__main__":
    main()
```

In this example, code does not run accidentally on import. This is because: * `helpers.py` defines functions only * `main.py` controls when the program runs

Why Use `main()` Functions or `__name__`?

Using this pattern in your projects allows you as a programmer to be more intentional with your code. You prevent unexpected output by making your files safer to import. It also makes debugging easier and code easier to read by breaking programs into smaller pieces.

Common Errors and Fixes

When using this pattern, you may find yourself running into issues sometimes. The following section is intended to help you identify problems and possible solutions.

Error: Code Runs When Imported

Cause: Code is written at the top level

Fix: Move the code into `main()` and use the `__name__` check

Error: Nothing Runs

Cause: `main()` exists but is never called

Fix:

```
if __name__ == "__main__":
    main()
```

Error: NameError for a Function

Cause: Function is called before it is defined or imported

Fix:

- Define functions above `main()`
- Import functions before using them

Error: Multiple Files Running Code

Cause: More than one file has top-level code

Fix:

- Only one file should act as the main program
- Other files should define functions only

Set Up

Create two Python files:

- `helpers.py`
- `main.py`

Copy, Change, Challenge

Copy

`helpers.py`

```
def say_hi():
    print("Hi!")
```

`main.py`

```
from helpers import say_hi

def main():
    say_hi()

if __name__ == "__main__":
    main()
```

Run `main.py`.

Change

Add another function to `helpers.py` and call it from inside `main()`.

Challenge

Create a program with:

- At least three helper functions
- A `main()` function that calls all of them
- No code that runs automatically when files are imported