The `__name__` global variable in Python is a special built-in variable that holds the name of the current module. Its primary purpose is to determine whether a script is being run as the main program or if it is being imported as a module into another script.

Here are some key points about `__name__`:

1. **Main Module Identifier**: When a Python script is executed, the `__name__` variable is automatically set to `"__main__"` if the script is being run as the main program. This allows you to execute certain code only if the script is run directly and not when it's imported into another script.
2. **Module Import Indicator**: When a Python script is imported as a module into another script, the `__name__` variable is set to the name of the module. This allows you to differentiate between whether the code is being executed as the main program or if it's being imported as a module.

Here's an example to illustrate how `__name__` works:

```
# File: example.py

def say_hello():
    print("Hello!")

# Check if the script is being run as the main program
if __name__ == "__main__":
    print("This script is being run directly.")
    say_hello()
else:
    print("This script is being imported as a module.")
```

Now, if you run `example.py` directly from the command line:

```
$ python example.py
```

Output:

```
This script is being run directly.
Hello!
```

However, if you import `example.py` into another script:

```
# File: main.py

import example
```

And then run `main.py`:

```
$ python main.py
```

Output:

```
This script is being imported as a module.
```

This demonstrates how the value of `__name__` changes depending on whether the script is being run directly or imported as a module.