1. Modules

Creating a Module

To create a module in Python, you simply need to create a new file with a `.py` extension. For example, let's create a file called `my_module.py`.

In this file, you can define functions, classes, or variables that you want to include in your module.

```python
# my_module.py

def greet(name):
    """Prints a greeting with the given name."""
    print(f"Hello, {name}!")

def add_numbers(a, b):
    """Adds two numbers and returns the result."""
    return a + b
```

In this example, we've defined two functions: `greet` and `add_numbers`. These functions are now part of our `my_module` module.

Importing a Module

To use the functions or variables defined in a module, you need to import the module into your Python script or program. There are several ways to import a module:

1. **Import the entire module**:

```python
import my_module

my_module.greet("Alice")  # Output: Hello, Alice!
result = my_module.add_numbers(3, 4)  # result = 7
```

2. **Import specific functions or variables**:

```python
from my_module import greet, add_numbers
```

```
greet("Bob")  # Output: Hello, Bob!
result = add_numbers(5, 6)  # result = 11
```

3. **Import all functions or variables using wildcard**:

```
from my_module import *

greet("Charlie")  # Output: Hello, Charlie!
result = add_numbers(7, 8)  # result = 15
```

Note: While the wildcard import (`from my_module import *`) is convenient, it can lead to naming conflicts and is generally not recommended, especially in larger projects.

Module Search Path

When you import a module, Python searches for the module file in several locations, known as the module search path. The search path includes:

1. The current directory
2. The directories listed in the `PYTHONPATH` environment variable
3. The default Python installation directories

You can view the module search path by running the following code:

```
import sys
print(sys.path)
```

This will print a list of directories that Python searches when you import a module.

Packages

In Python, a package is a way to organize related modules into a directory hierarchy. A package is essentially a directory containing one or more Python modules and a special file called `__init__.py`.

Here's an example structure of a package called `my_package`:

```
my_package/
    __init__.py
    module1.py
    module2.py
    subpackage/
        __init__.py
        module3.py
```

To import a module from a package, you need to specify the package name followed by the module name, separated by a dot (`.`). For example:

```
import my_package.module1
from my_package.subpackage import module3
```

The `__init__.py` file in a package is used to initialize the package and can also contain code that runs when the package is imported.

See Also
2. The main program