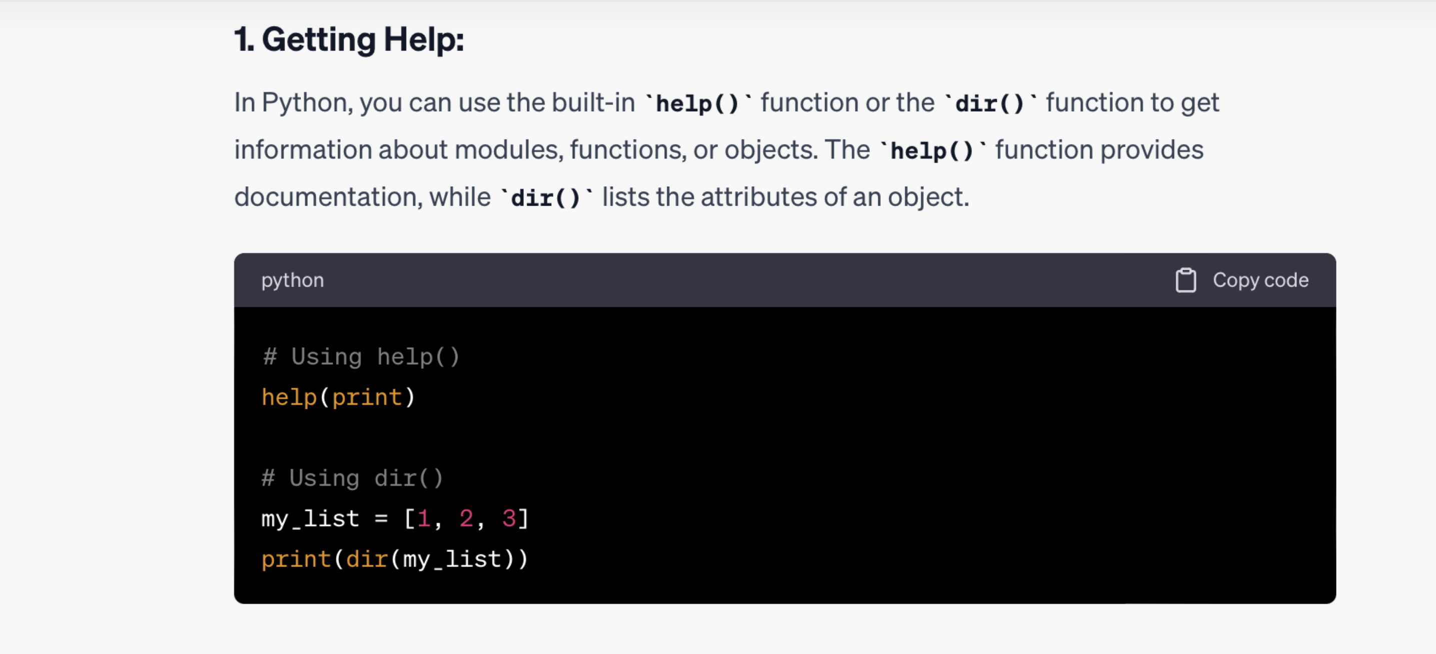
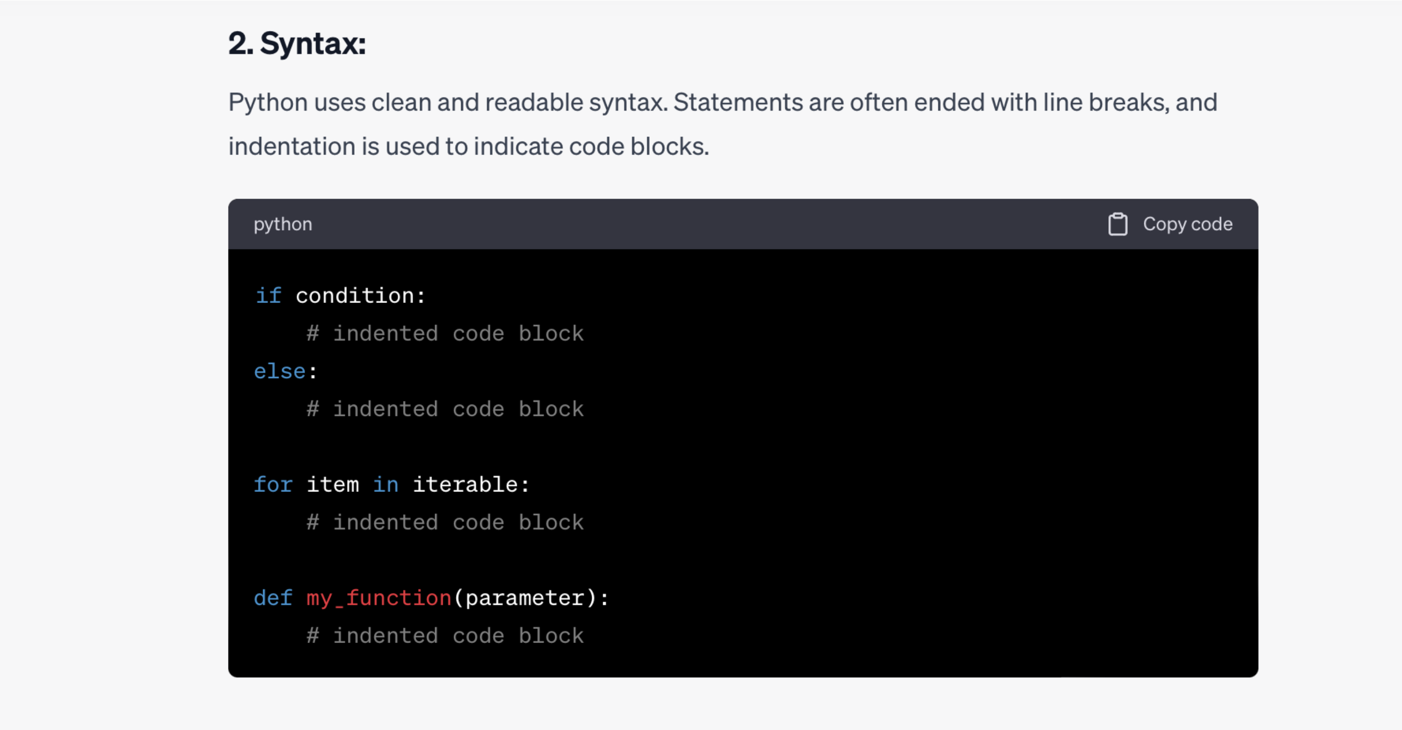


1. Getting Help:

In Python, you can use the built-in `help()` function or the `dir()` function to get information about modules, functions, or objects. The `help()` function provides documentation, while `dir()` lists the attributes of an object.

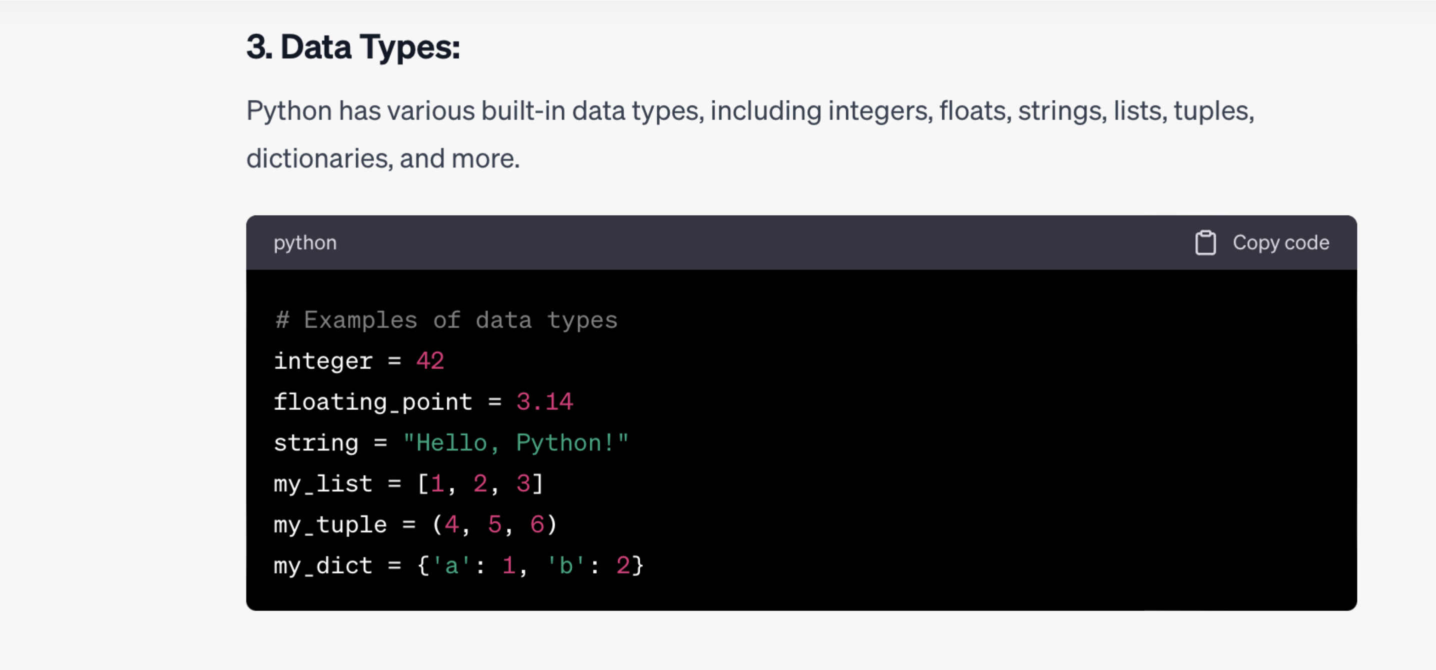
```python
# Using help()
help(print)

# Using dir()
my_list = [1, 2, 3]
print(dir(my_list))
```



2. Syntax:

Python uses clean and readable syntax. Statements are often ended with line breaks, and indentation is used to indicate code blocks.

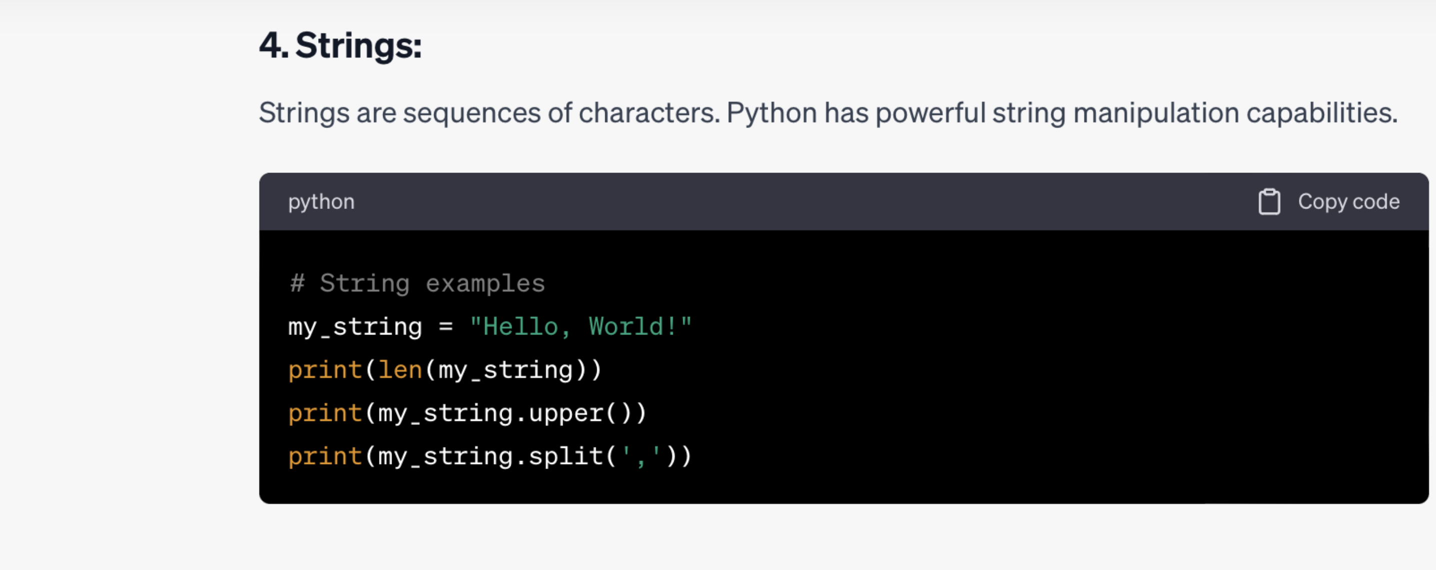
```python
if condition:
    # indented code block
else:
    # indented code block

for item in iterable:
    # indented code block

def my_function(parameter):
    # indented code block
```



3. Data Types:

Python has various built-in data types, including integers, floats, strings, lists, tuples, dictionaries, and more.

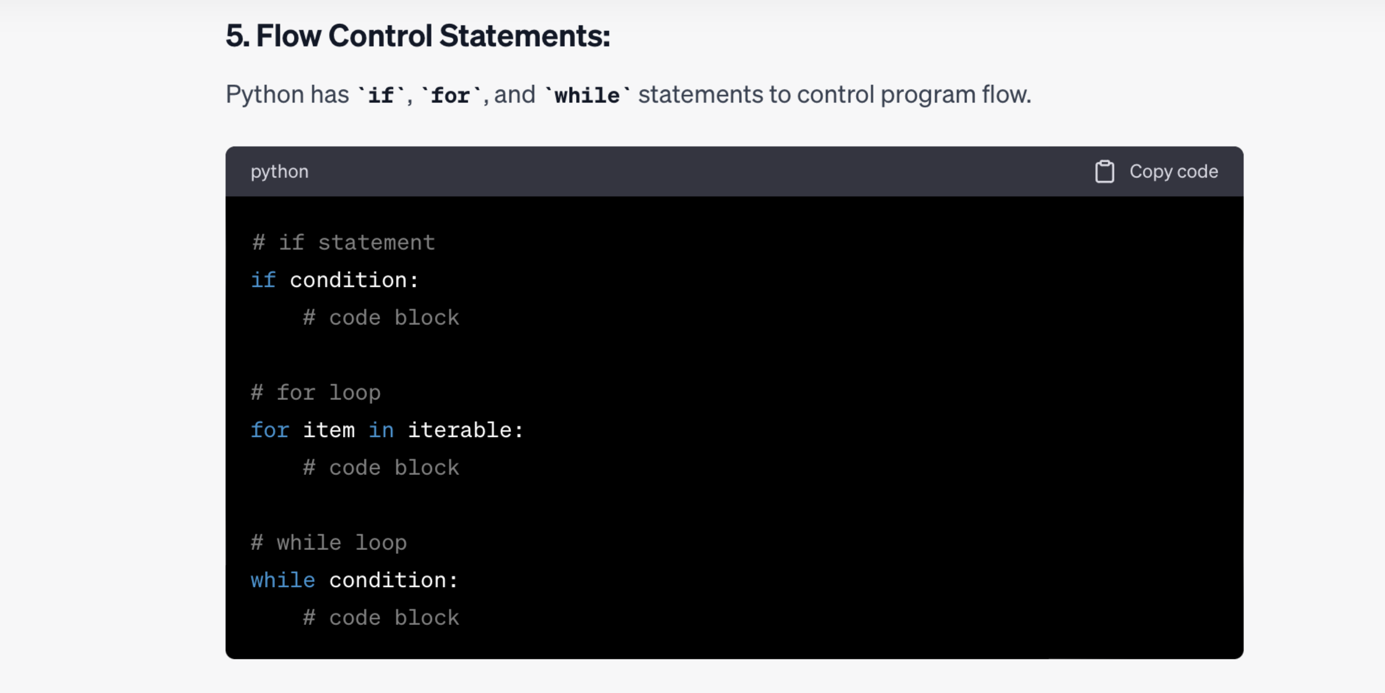
```python
# Examples of data types
integer = 42
floating_point = 3.14
string = "Hello, Python!"
my_list = [1, 2, 3]
my_tuple = (4, 5, 6)
my_dict = {'a': 1, 'b': 2}
```



4. Strings:

Strings are sequences of characters. Python has powerful string manipulation capabilities.

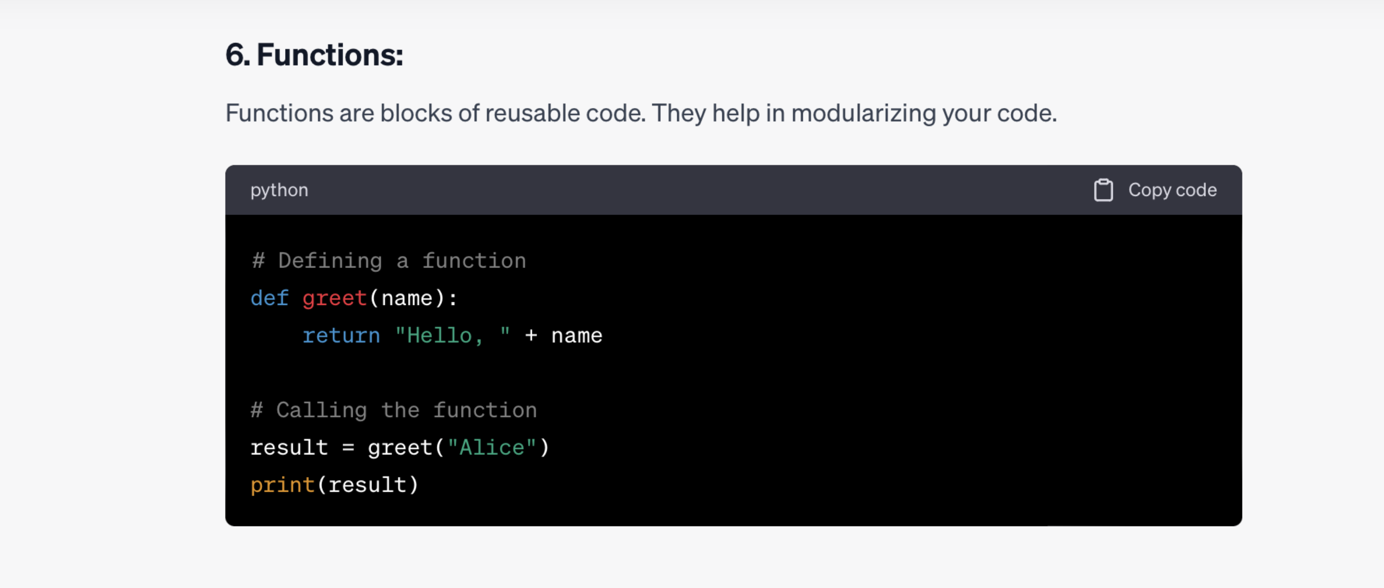
```python
# String examples
my_string = "Hello, World!"
print(len(my_string))
print(my_string.upper())
print(my_string.split(','))
```



5. Flow Control Statements:

Python has `if`, `for`, and `while` statements to control program flow.

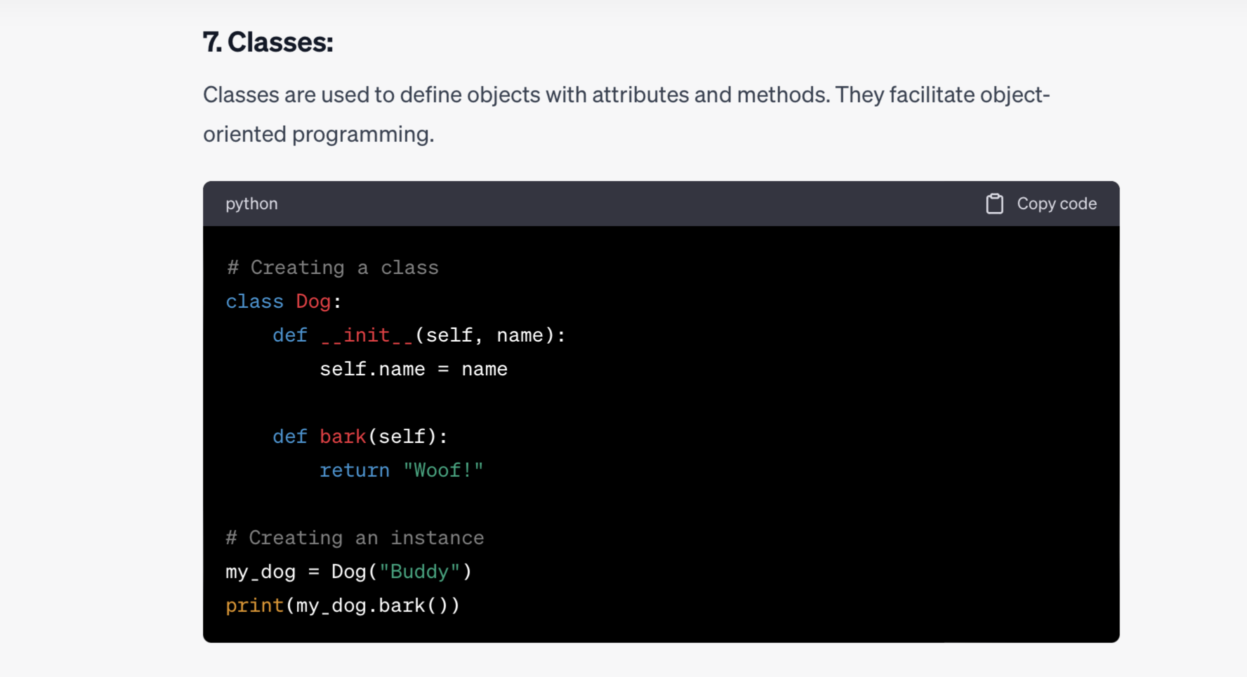
```python
# if statement
if condition:
    # code block

# for loop
for item in iterable:
    # code block

# while loop
while condition:
    # code block
```



6. Functions:

Functions are blocks of reusable code. They help in modularizing your code.

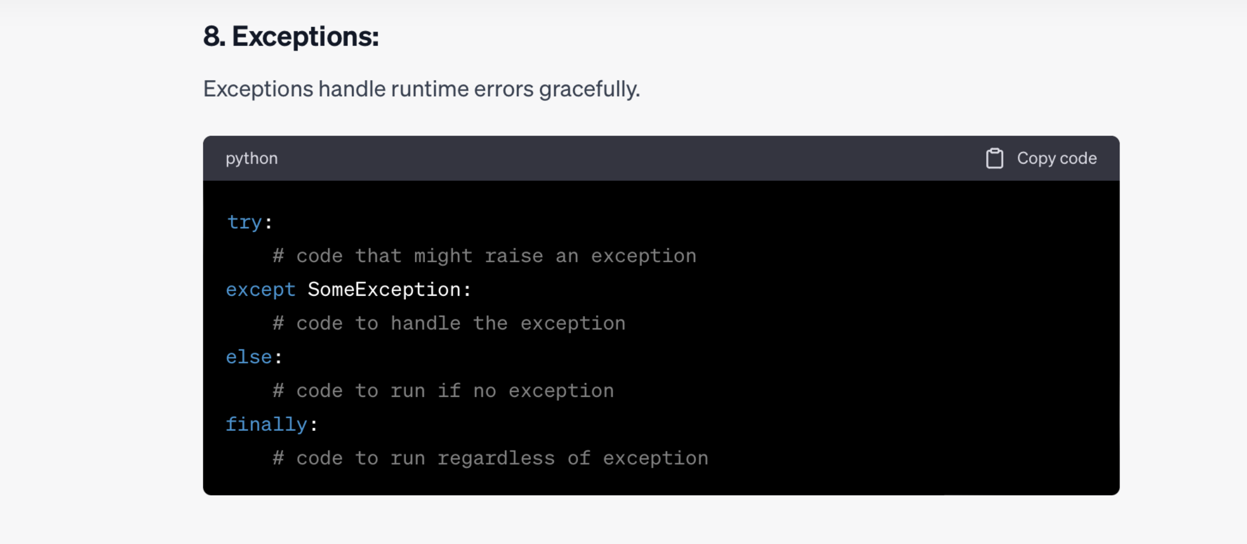
```python
# Defining a function
def greet(name):
    return "Hello, " + name

# Calling the function
result = greet("Alice")
print(result)
```



7. Classes:

Classes are used to define objects with attributes and methods. They facilitate object-oriented programming.

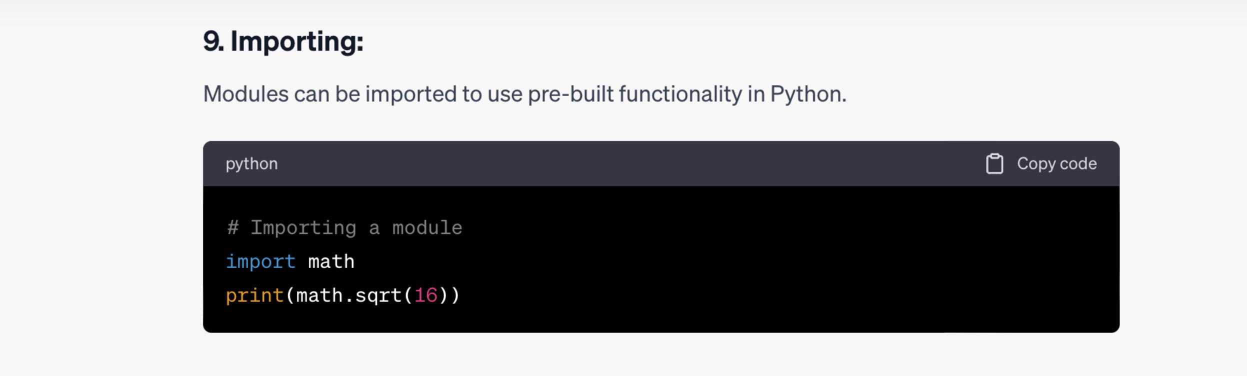
```python
# Creating a class
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self):
        return "Woof!"

# Creating an instance
my_dog = Dog("Buddy")
print(my_dog.bark())
```



8. Exceptions:

Exceptions handle runtime errors gracefully.

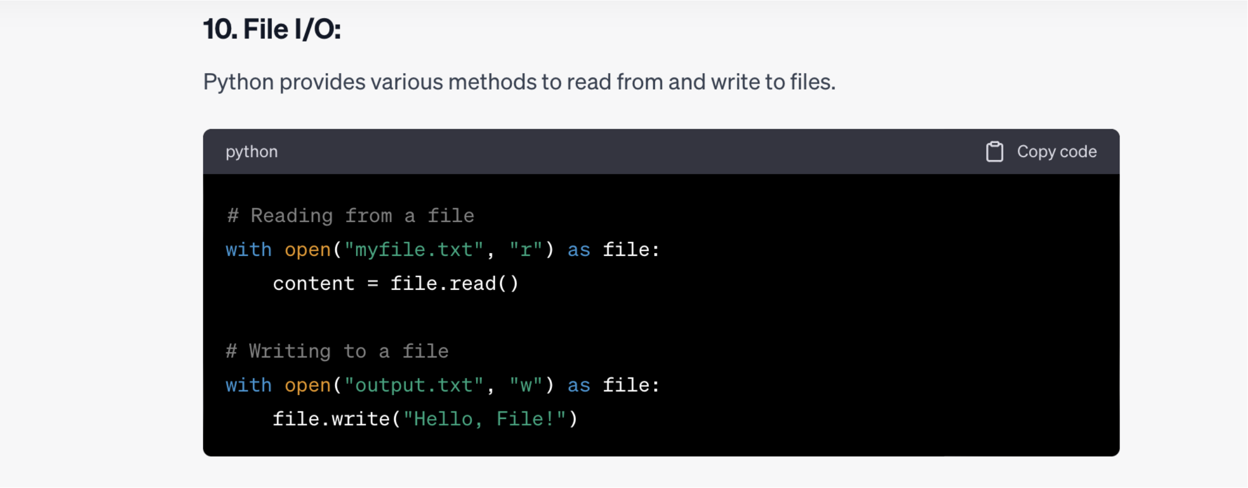
```python
try:
    # code that might raise an exception
except SomeException:
    # code to handle the exception
else:
    # code to run if no exception
finally:
    # code to run regardless of exception
```



9. Importing:

Modules can be imported to use pre-built functionality in Python.

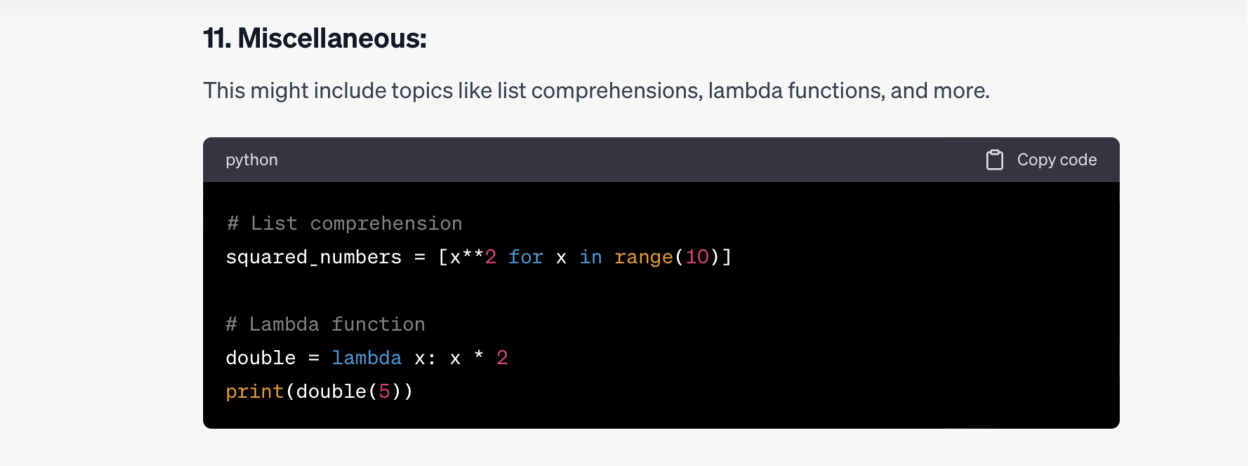
```python
# Importing a module
import math
print(math.sqrt(16))
```



10. File I/O:

Python provides various methods to read from and write to files.

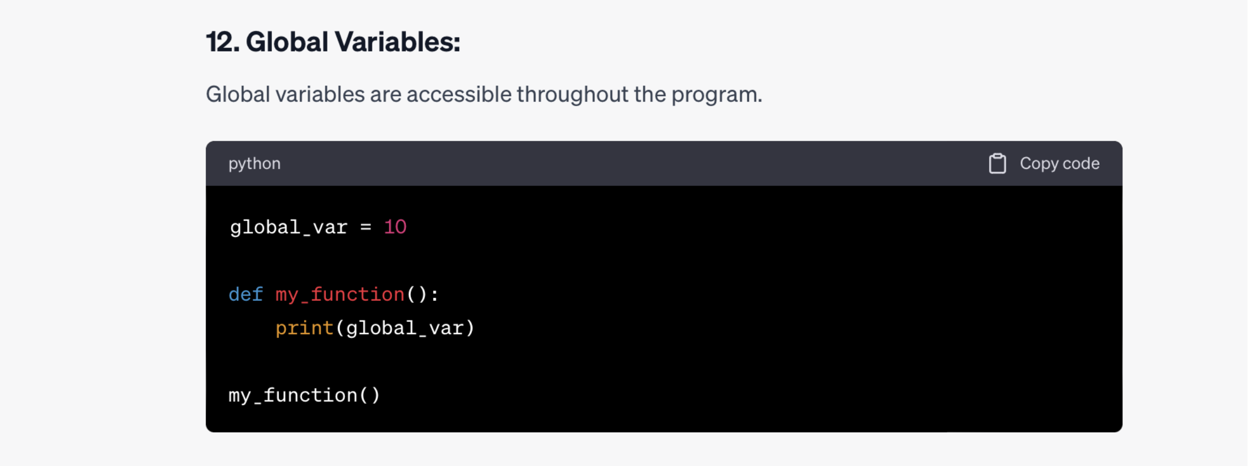
```python
# Reading from a file
with open("myfile.txt", "r") as file:
    content = file.read()

# Writing to a file
with open("output.txt", "w") as file:
    file.write("Hello, File!")
```



11. Miscellaneous:

This might include topics like list comprehensions, lambda functions, and more.

```python
# List comprehension
squared_numbers = [x**2 for x in range(10)]

# Lambda function
double = lambda x: x * 2
print(double(5))
```



12. Global Variables:

Global variables are accessible throughout the program.

```python
global_var = 10

def my_function():
    print(global_var)

my_function()
```