Day 4 - Functions and Data Structures

Defining and Using Functions in Python

In Python, functions are reusable blocks of code that perform a specific task. They help modularize your code and make it more organized. Here's how you can define and use functions in Python.

Defining a Function

To define a function in Python, you use the `def` keyword followed by the function name and parentheses. Any arguments or parameters the function requires are specified within the parentheses. The function body is indented below the function definition.

Example:

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

Calling a Function

To call a function, you simply write its name followed by parentheses. If the function takes arguments, you pass them within the parentheses.

Example:

```
greet()  # Output: Hello, there!
```

Function Arguments

Functions can accept arguments, which are values passed into the function for it to work with. There are two types of function arguments: positional arguments and keyword arguments.

Positional Arguments

Positional arguments are passed based on their position or order. The order and number of arguments passed must match the function definition.

Example:

```
def greet(name):
    print(f"Hello, {name}!")

greet("Alice")  # Output: Hello, Alice!
```

Keyword Arguments

Keyword arguments are passed with a keyword and corresponding value, allowing you to pass arguments in any order. You can also provide default values for arguments.

Example:

```
def greet(name, message="Hello"):
    print(f"{message}, {name}!")

greet(name="Alice")                    # Output: Hello, Alice!
greet(message="Hi", name="Bob")       # Output: Hi, Bob!
```

Returning Values

Functions can also return values using the return statement. This allows the function to compute a result and provide it back to the caller.

Example:

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

result = add_numbers(3, 4)
print(result)  # Output: 7
```

Basic Data Structures: Lists and Dictionaries

Lists

A list is a versatile data structure in Python that stores an ordered collection of elements. Elements within a list can be of different data types, and they are enclosed in square brackets ([]). Lists are mutable, meaning you can modify them after creation.

Example:

```
fruits = ["apple", "banana", "orange"]
print(fruits)              # Output: ['apple', 'banana', 'orange']
print(fruits[0])           # Output: apple
fruits.append("grape")     # Add an element to the end of the list
print(fruits)              # Output: ['apple', 'banana', 'orange', 'grape']
```

Dictionaries

A dictionary is another commonly used data structure in Python. It stores a collection of key-value pairs, where each key is unique. Dictionaries are enclosed in curly braces ({}) and have keys and corresponding values separated by a colon (😃.

Example:

```
student = {
    "name": "Alice",
    "age": 20,
    "university": "ABC"
}
print(student)                     # Output: {'name': 'Alice', 'age': 20,
'university': 'ABC'}
print(student["name"])             # Output: Alice
student["major"] = "Computer Science"  # Add a new key-value pair
print(student)                     # Output: {'name': 'Alice', 'age': 20,
'university': 'ABC', 'major': 'Computer Science'}
```

These are the basics of defining and using functions, as well as working with lists and dictionaries in Python.