Chapter 4 :Variables and Data Types in Python

Variables and Data Types in Python

Introduction to Variables

A variable in Python is essentially a name given to a memory location where a value is stored. Variables are fundamental in programming as they allow you to store, manipulate, and retrieve data. Here are some examples:

```
a = 30          # 'a' is an integer variable
b = "updategadh"     # 'b' is a string variable
c = 71.22       # 'c' is a floating-point variable
```

In these examples:

- `a` is an integer, storing the value `30`.
- `b` is a string, storing the text `"updategadh"`.
- `c` is a float, storing the value `71.22`.

Understanding Data Types

Python automatically determines the type of data assigned to a variable, making it easy to work with different kinds of data. Here are the primary data types in Python:

1. **Integers (`int`)**: Whole numbers, e.g., `71`, `-20`.
2. **Floating-point numbers (`float`)**: Numbers with a decimal point, e.g., `88.44`, `-0.005`.
3. **Strings (`str`)**: A sequence of characters, e.g., `"updategadh"`, `"Python is fun!"`.
4. **Booleans (`bool`)**: Represents two values: `True` or `False`.
5. **None**: Represents the absence of a value or a null value.

```
a = 71           # Python identifies 'a' as an integer <class 'int'>
b = 88.44        # Python identifies 'b' as a float <class 'float'>
name = "updategadh"   # Python identifies 'name' as a string <class 'str'>
```

Rules for Naming Variables (Identifiers)

When naming variables, also known as identifiers, it's important to follow specific rules:

- A variable name can contain alphabets, digits, and underscores (`_`).
- A variable name must start with an alphabet or an underscore.
- A variable name cannot start with a digit.
- No spaces are allowed within a variable name.

Examples of valid variable names:

- `rishabh`
- `one8`
- `seven`
- `_seven`

Examples of invalid variable names:

- `7up` (starts with a digit)

- `my variable` (contains a space)

Operators in Python

Operators in Python are symbols that perform operations on variables and values. Here are some common types of operators:

1. **Arithmetic Operators**: Used to perform basic arithmetic operations.

- `+` : Addition
- `-` : Subtraction
- `*` : Multiplication
- `/` : Division **Example**:

```
a = 10
b = 5
result = a + b  # result is 15
```

1. **Assignment Operators**: Used to assign values to variables.

- `=` : Assigns the right-hand value to the left-hand variable
- `+=` : Adds the right-hand value to the left-hand variable and assigns the result to the left-hand variable
- `-=` : Subtracts the right-hand value from the left-hand variable and assigns the result to the left-hand variable **Example**:

```
a = 10
a += 5  # a is now 15
a -= 3  # a is now 12
```

1. **Comparison Operators**: Used to compare two values.

- `==` : Checks if two values are equal
- `!=` : Checks if two values are not equal
- `>` : Checks if the left value is greater than the right value
- `<` : Checks if the left value is less than the right value
- `>=` : Checks if the left value is greater than or equal to the right value
- `<=` : Checks if the left value is less than or equal to the right value **Example**:

```
a = 10
b = 5
is_equal = (a == b)  # is_equal is False
```

1. **Logical Operators**: Used to combine conditional statements.

- `and` : Returns `True` if both statements are true
- `or` : Returns `True` if at least one statement is true
- `not` : Reverses the result, returns `False` if the result is true **Example**:

```
a = 10
b = 5
c = 20
```

```
    result = (a > b) and (c > a)  # result is True
```

Practical Exercise: Combining Concepts

Let's create a simple Python script that uses variables, different data types, and operators:

```
# variables.py

# Define variables of different types
age = 25                 # Integer
height = 5.9             # Float
name = "Alice"           # String
is_student = True        # Boolean

# Perform some operations
years_left = 65 - age    # Arithmetic operation
welcome_message = "Hello, " + name + "!"  # String concatenation

# Print the results
print(welcome_message)
print(f"You have {years_left} years left until retirement.")
print(f"Is {name} a student? {is_student}")
```

Conclusion

In this chapter, we've explored variables and data types in Python, understanding how to store and manipulate different types of data. We've also learned about naming rules for variables and common operators used in Python. Mastering these concepts is essential as they form the foundation of all Python programming. Keep practicing, and you'll be well on your way to becoming proficient in Python!

Chapter 3: Practice Set Solution - Modules, Comments, and Pip

Practice Problems and Solutions

1. **Write a Python program to display the following text using the `print` function: Problem:**

```
    Python is fun!
    Let's learn Python together!
```

Solution:

```
    # fun_with_python.py

    # This program prints a couple of motivational lines about Python
    print("Python is fun!")
    print("Let's learn Python together!")
```

Instructions: Save the above code in a file named `fun_with_python.py` and run it using `python fun_with_python.py` in your terminal.

1. **Use Python REPL to calculate the sum of the first 10 positive integers. Solution:**

```
    # Enter the following code in the Python REPL
```

```
# Using a loop
total = 0
for i in range(1, 11):
    total += i
print("Sum of the first 10 positive integers:", total)

# Alternatively, using the sum function
total = sum(range(1, 11))
print("Sum of the first 10 positive integers:", total)
```

Instructions: Open your terminal, type `python` to enter the Python REPL, and then type the code above to see the output.

1. **Install the `requests` module using Pip and use it to fetch data from a public API. Problem:**
   Fetch and print the current weather for a specific city using the OpenWeatherMap API. **Solution:**

```
# First, install the requests module
pip install requests
```

```
# weather_fetcher.py

import requests

# Replace with your own API key
API_KEY = 'your_api_key'
CITY = 'London'
URL = f'http://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}&units=metric'

# Sending a GET request to the API
response = requests.get(URL)

# If the request was successful
if response.status_code == 200:
    data = response.json()
    temperature = data['main']['temp']
    description = data['weather'][0]['description']
    print(f"The current temperature in {CITY} is {temperature}°C with {description}.")
else:
    print("Failed to retrieve data.")
```

Instructions: Save the code in a file named `weather_fetcher.py` and run it using `python weather_fetcher.py`. Don't forget to replace `'your_api_key'` with an actual API key from OpenWeatherMap.

1. **Write a Python program to list all files and directories in the current working directory using the `os` module. Problem:**
   List all files and directories in the current directory. **Solution:**

```
# list_directory.py

import os

# Get the current working directory
current_directory = os.getcwd()
```

```
# List all files and directories in the current directory
files_and_dirs = os.listdir(current_directory)

print("Files and directories in '", current_directory, "':")
for item in files_and_dirs:
    print(item)
```

Instructions: Save the code in a file named `list_directory.py` and run it using `python list_directory.py`. This will display the contents of the directory from which you run the script.

1. **Enhance the program you wrote for problem 4 by adding detailed comments. Solution:**

```
# list_directory_with_comments.py

import os  # Import the os module to interact with the operating system

# Get the current working directory using os.getcwd()
current_directory = os.getcwd()

# Use os.listdir() to list all files and directories in the current directory
files_and_dirs = os.listdir(current_directory)

# Print the current directory path
print("Files and directories in '", current_directory, "':")

# Loop through the list of files and directories and print each one
for item in files_and_dirs:
    print(item)
```

Instructions: Save this code in a file named `list_directory_with_comments.py` and run it using `python list_directory_with_comments.py`. The comments explain each step of the code, making it easier to understand.

These problems and solutions should help you get hands-on experience with Python's fundamental features, making you more comfortable with the basics of programming, using modules, and commenting code effectively.

Chapter 4 : Practice Set (Variables and Data Types)

Practice Problems

1. **Write a Python program to create variables of different types and print their values.**
    - Define an integer, float, string, and boolean variable, and display their values using the `print` function.
2. **Create a Python script that performs arithmetic operations on two numbers and displays the results.**
    - Use the `+`, `-`, `*`, and `/` operators to perform addition, subtraction, multiplication, and division, respectively.
3. **Write a Python program that uses logical operators to check multiple conditions.**
    - Define three variables and use the `and`, `or`, and `not` operators to evaluate complex conditions.
4. **Write a Python script to demonstrate the use of assignment operators.**
    - Start with an integer variable and use `+=`, `-=`, `*=`, and `/=` operators to modify its value.
5. **Comment on a program that calculates the area of a rectangle using variables for length and width.**
    - Include comments explaining the purpose of each variable, the formula used, and the result.