UNIT 3

Introduction to Python Programming

Student Learning Outcomes

By the end of this chapter, you will be able to:

- Understand basic programming concepts and set up a Python development environment.
- Write and interpret basic Python syntax and structure, including variables, data types, and input/output operations.
- Use various operators and expressions in Python, including arithmetic, comparison assignment, logical operators and operator precedence.

Introduction

Welcome to the world of Python programming! Python is a popular language known for its simplicity and readability, making it ideal for both beginners and professionals. In this chapter, we will start with the basics, including setting up your development environment and learning fundamental programming concepts. By the end of this chapter, you will have a comprehensive understanding of Python, enabling you to write, test, and debug your own programs.

3.1 Introduction to Python Programming

Python is a widely-used high-level programming language famous for its simplicity and readability. It is a versatile language and applicable to various fields, including web development, data analysis and artificial intelligence etc. Whether you are creating a simple script or developing a complex software, Python's user-friendly nature helps you get started quickly and efficiently.

Basic Programming Concepts

Computer programming is the process of creating a set of instructions that tell a computer how to perform a task. These instructions are written in a programming language that the computer can understand and execute. Think of computer programming like giving directions to a friend on how to reach your house. You need to be clear and precise so your friend doesn't get lost. Similarly, when we write programs, we give clear and precise instructions to the

computer to complete specific tasks.

Setting Up Python Development Environment

The development environment refers to the process of preparing a computer to write, run, and debug Python code effectively. This involves installing and configuring the necessary software, tools, and libraries that make development smoother and more efficient.

TOOL TIP

When installing Python, make sure to check the box that says "Add Python to PATH." This makes it easier to run Python from the command line. We can also use online services to write and run Python program.

Steps of Python installation

To complete the installation process of Python, follow these steps:

1. Download Python:

Visit https://www.python.org/. Navigate to the "Downloads" section and click on the latest version of Python (e.g., Python 3.13.x) for your operating system (Windows, macOS, or Linux).

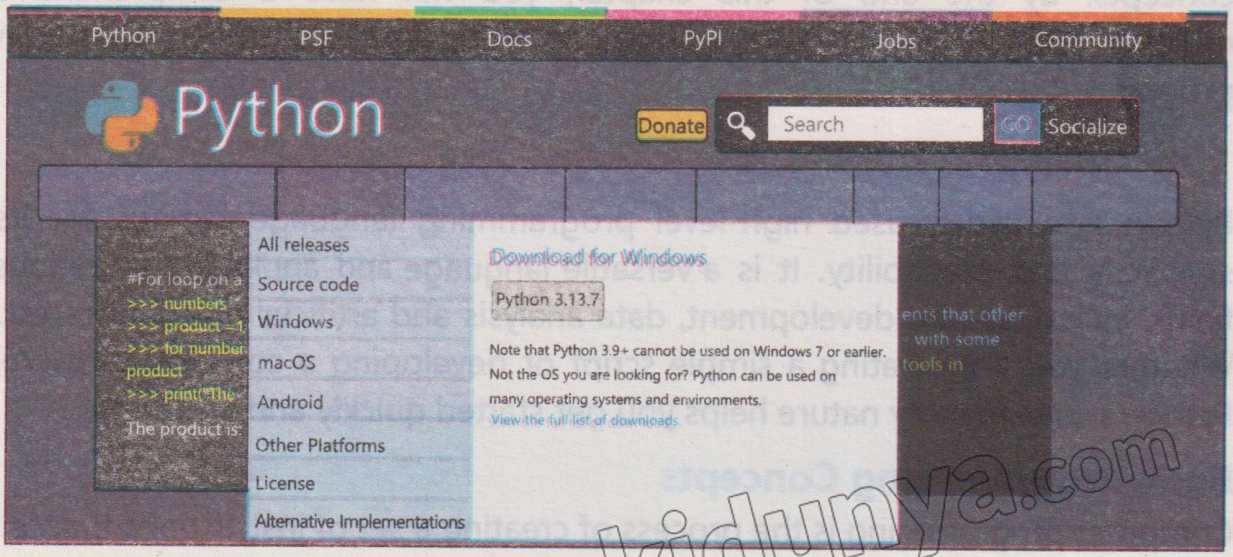


Figure 3.1: Python Home Page

The installer will be downloaded automatically.

2. Install Python:

- Run the downloaded .exe file.

- Check the box "Add Python to PATH" during installation.
- Select "Install Now" or customize the installation if needed (e.g., choose the installation directory).
- Wait for the installation to complete, then click "Close".

Figure 3.2: Setup Process

3. Set Up an IDE:

- Download and install a popular IDE like PyCharm, VS Code, or IDLE (included with Python).
- PyCharm is a powerful IDE for Python development, available in a free Community edition or a paid Professional edition.
- Visit: https://www.jetbrains.com/pycharm/ to download the version suitable for your operating system (Windows, macOS, or Linux).

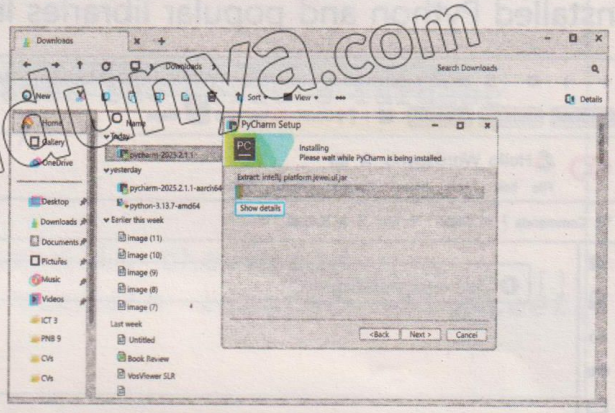


Figure 3.3: Installation Process

4. Configuring the IDE with Python

- Open PyCharm and start a new project.
- In the "Configure" or "Settings" menu, go to "Project Interpreter."
- Click the gear icon, select "Add Interpreter," and choose the path to your Python executable (e.g., C:\Python312\python.exe on Windows or /usr/bin/python3 on

Figure 3.4: Python IDE

Linux/macOS). Browse to your installation folder if it's not auto-detected. Apply the settings and let PyCharm index the interpreter.

5. Testing the Configuration

- Create a new file (e.g., test.py) in your IDE.
- Write a simple script: print("Hello, Python!").
- Run the script (use the "Run" button in PyCharm/IDLE or press F5 in VS Code after setting up a task).
- If the output appears (e.g., "Hello, Python!"), your IDE is correctly configured.

Google Colab

Another excellent option for using Python is Google Colab. It is a free cloud-based platform that provides a Jupyter notebook environment with pre-installed Python and popular libraries like NumPy, Pandas, and Matplotlib. To

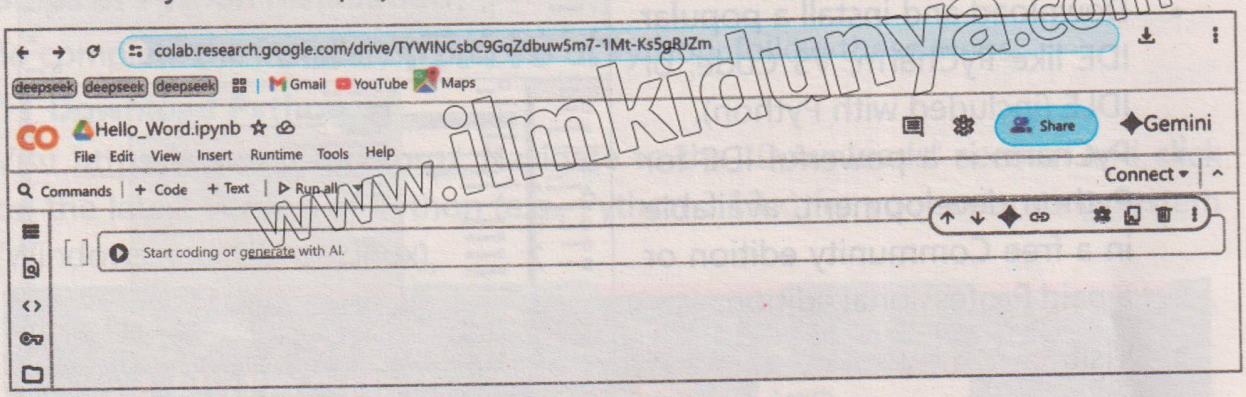


Figure 3.5: Google Colab IDE

get started, visit https://colab.research.google.com/, sign in with a Google account, and create a new notebook. No local installation is required, making it ideal for beginners or those working on machines with limited resources. You can write and execute Python code in cells.

3.2 Basic Python Syntax and Structure

The following Python program demonstrates the simplicity and readability of the language:

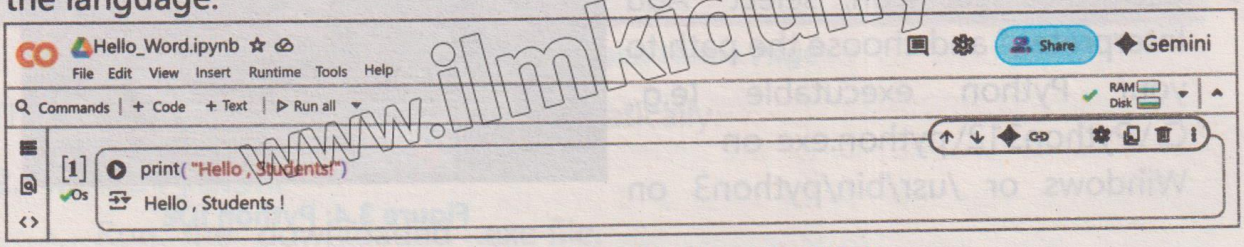


Figure 3.6: Python Program in Colab

In this example, the print function is utilized to output the message enclosed in double quotation marks. This illustrates Python's straightforward syntax, where functions like print are used to perform actions such as displaying text.

Python Comments

Comments are the lines that are not executed by the Python interpreter. They are used to provide explanations or notes for the Programmer. Single-line comments start with the # symbol.

```
# This is a single - line comment
print ( "K2 is the second-highest mountain in the
world " )
'''
This is a multi-line comment.
'''
print ( "Edhi Foundation operates the world's largest
volunteer ambulance network." )
'''
Output:
K2 is the second-highest mountain in the world
Edhi Foundation operates the world's largest volunteer
ambulance network.
'''
```

Interesting Information

We can use ''' ...''' (Triple Quotes) in python for multiline comments but actually they are multiline string literals which are ignored at runtime.

Variables in Python

A variable in programming functions as a storage container within a computer's memory. It allows the storage of data that can be retrieved later in the code. In the example below, the variable age is utilized to hold numerical data. The value of a variable can change throughout the execution of a program, which is why it is referred to as a "variable":

```
age = 71
print("Azam lived for", age, "years")
age = 60
print ( " Iqbal lived for", age, "years")
```

```
# Output
# Azam lived for 71 years
# Iqbal lived for 60 years
```

This example illustrates how the variable "age" is assigned different values to represent the ages of Azam and Iqbal, which are then printed as part of the output.

Variable Naming Rules in Python

Variable names in Python must adhere to the following rules:

- The name must begin with a letter (a-z, A-Z) or an underscore (_). It cannot begin with a digit.
- Subsequent characters can include letters, digits (0-9), or underscores ( ).
- Variable names are case-sensitive, meaning age and Age are considered two different variables.
- Python's reserved keywords, such as for, while, if, etc., cannot be used as variable names.
- **Reserved words** in Python are those words which have predefined meaning in Python. They cannot be used as variable, function or any other identifier.

DO YOU KNOW?

Following are some of the reserved words in Python:

- if, else, elif, for, while, break, continue, and, or, not, in, is, True, False, None and print

We Can not use them as a variable name

Types of Variables

In Python, you can create variables of different types to store various kinds of data. Python itself decides the data type of variable unless specified. Here are some common types of variables:

- **Integer (int):** Stores whole numbers. Example: age = 17
- **Floating-point (float):** Stores decimal numbers. Example: price = 19.99
- **String (str):** Stores text. Example: name = "Ali"
- **Boolean (bool):** Stores True or False. Example: is_student = True

Input and Output Operations in Python

Input and output operations allow you to interact with the user. You can ask the user to enter data (input) and display information to the user (output).

Input: We use the **input ()** function to get user input. The input () function displays a message on the screen and waits for the user to type something and press Enter. The text entered by the user is then stored in a variable. For example:

```
name = input("Enter your name: ")
print (" Hello, " + name + "!")
# Output
# Enter your name: Maryam Ashraf
# Hello, Maryam Ashraf!
```

First line of code asks the user to enter their name and stores it in the variable name.

Output: Use the print () function to display information on the screen. The print () function takes one or more arguments and displays them. For example:
Second line of code displays a greeting message that includes the user's name.

DO YOU KNOW?

1. In print () function we use ',' between multiple values/ variables to be printed separately.
2. In python indentation defines structure or scope your code. Incorrect indentation may cause error in your code.

Handling Integer and Float Inputs

To handle numeric inputs, you use the int() or float() functions to convert input strings to integers or floating-point numbers.

Integer Inputs

```
# Example : Handling integer input
  user_age = int(input("Enter your age: "))
  print("Your age is:",user_age)
# Output
# Enter your age: 16
# Your age is : 16
```

Float Inputs

```
# Example: Handling float input
user_height = float(input("Enter your height in meters:
") )
print("Your height is", user_height,"meters")
# Output
# Enter your height in meters: 1.5
# Your height is 1.5 meters
```

3.3 Operators and Expressions

Operators are symbols that perform operations on operands. An expression is a combination of operators, and values that produces a result. Let's explore

different types of operators in Python.

Arithmetic Operators

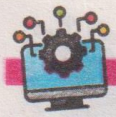
Arithmetic operators are used to perform basic mathematical operations such as addition, subtraction, multiplication, division, modulus, exponentiation, and floor division as shown in the following code.

```
# Define variables
a= 10
b= 3
# Perform all arithmetic operations on these numeric
variables and print results
print(a, "+", b, "=", a + b) # Output: 10+3=13
print(a, "*", b, " = " , a * b) # Output: 10 * 3 = 30
print(a, "/", b, " = ", a / b)
# Output: 10 / 3 = 3.333333333333333
print(a, "//", b, " = ",a // b) #floor division
# Output: 10 // 3=3
print(a, "%", b, " = ", a % b)
# Output: 10%3 = 1
print(a,"**", b, " = ", a ** b)
# ** represent power operator:
# Output: 10**3= 1000
```

Comparison Operators

Comparison operators are used to compare two values or expressions. They determine the relational logic between them, such as equality, inequality, greater than, less than, and so on. These operators return a Boolean value (True or False) based on the comparison result. Here's a Python program that demonstrates the usage of all comparison operators.

```
Define variables
X = 10
Y = 5
# Greater than
print (x, ">", y, "=", x > y) # Output: 10 > 5 = True
# Less than
print (x, "<", y, "=", x < y) # Output: 10 < 5 = False
```



```
#Equal to
print (x, "==", y, "=", x == y) # Output: 10 == 5 = False
# Not equal to
print (x, "!=", y, "=", x != y) # Output: 10 != 5 = True
# Greater than or equal to
print (x, ">=", y, "=", x >= y) # Output: 10 >= 5 = True
# Less than or equal to
print (x, "<=", y, "=", x <= y) # Output: 10 <= 5 = False
```

The above code demonstrates Python's comparison operators by defining two variables, **x** and **y**, and comparing them using various operators like >, <, ==, !=, >=, and <=.

Assignment Operators

Assignment operators are used to assign values to variables. The most common assignment operator is the equal sign (=), which assigns the value on the right to the variable on the left. There are also compound assignment operators like +=, -=, *=, and /=, which combine arithmetic operations with assignment.

```
# Define initial values
a = 10
b = 5

# Assignment
x = a; print("a =" , x)# Output: a = 10
# Addition assignment
a += b; print("a after addition =" , a)# Output: a after addition = 15
# Subtraction assignment
a -= b; print("a after subtraction =" , a)# Output: a after subtraction= 10
# Multiplication assignment
a *= b; print("a after multiplication =" , a)# Output: a after multiplication = 50
# Division assignment
a /= b; print("a after division =" , a)# Output: a after division = 10.0
#Floor division assignment
a //= b; print("a after floor division =" , a)# Output: a after floor division = 2
# Modulus assignment
a %= b; print("a after modulus =" , a)# Output: a after modulus = 2.0
# Exponentiation assignment
a **= b; print("a after exponentiation =", a)# Output: a after exponentiation = 32.0
```

Above code illustrates the use of compound assignment operators in Python. It starts by assigning a value to a variable "a". It then demonstrates various assignment operations including addition, subtraction, multiplication, division, floor division, modulus, and exponentiation. Each operation updates the value of "a" and the results are printed with the updated values. The code comments show the output of each operation.

Logical Operators

Logical operators are used to combine multiple conditions or expressions. The most common logical operators are "and, or, and not", they are used to perform logical operations and return Boolean values based on the evaluation of the expressions involved.

```
# Define variables
x = True
y = False
# Logical AND
logical_and = x and y
 print(x, "and", y ,"=", logical_and) # Output: True and
False =False
# Logical OR
logical_or = x or y
print(x , "or " , y , " =", logical_or) # Output: True or
False = True
# Logical NOT
logical_not_x = not x
print("not", x, " = " , logical_not_x) # Output: not True=
False
logical_not_y = not y
print("not", y, "=" , logical_not_y) # Output: not False=
True
```

The above code demonstrates the use of logical operators in Python. It defines two Boolean variables, x and y, and performs logical operations such as "and, or, not" on them. The results of these logical operations are printed, showing whether the expressions evaluate to True or False. The code also includes comments with the output of each logical operation.

Operator Precedence in Python

Operator precedence determines the order in which operations are performed in an expression. In Python as well as in Mathematics, certain operators have higher precedence and are evaluated before others. Understanding this helps ensure that your calculations are done correctly.

- **Parentheses '()':** Highest precedence. Operations inside parentheses are performed first. (3 + 2) * 4 evaluates to 20.
- **Exponentiation:** Performs power operations next.
  2 + 2 * 2 ** 3 = 18
- **Multiplication '*', Division '/', and Modulus '%':** These operations come next.
- **Addition '+' and Subtraction '-':** These have lower precedence compared to multiplication and division.
  For example: 4 * 3 + 10 / 2 – 11 % 3
  4 * 3 evaluates to 12, 10 / 2 evaluates to 5.0 and 11 % 3 evaluates to 2
  Perform add and subtract operators 12 + 5.0 – 2 = 15.

Example: Consider the following code to show operator precedence:

```
print("=== PYTHON OPERATOR PRECEDENCE DEMONSTRATION ===\n")
# 1. Arithmetic Operators
print("1. ARITHMETIC OPERATORS:")
print(f"3 + 2 * 5    = {3 + 2 * 5}")   # Multiplication before addition
print(f"(3 + 2) * 5 = {(3 + 2) * 5}")# Parentheses change order
print(f"2 ** 3 * 4   = {2 ** 3 * 4}") # Exponentiation before multiplication
print(f"15 / 3 * 2   = {15 / 3 * 2}") # Division and multiplication (left to right)
print(f"15 // 4 + 2 = {15 // 4 + 2}")# Floor division before addition
print(f"11 // 4 + 2 = {11 // 4 + 2}")# Floor division before addition
print(f"17 % 5 * 2   = {17 % 5 * 2}") # Modulo before multiplication
# Output
# === PYTHON OPERATOR PRECEDENCE DEMONSTRATION ===
```

```
# 1. ARITHMETIC OPERATORS:
# 3 + 2 * 5       = 13
# (3 + 2) * 5     = 25
# 2 ** 3 * 4      = 32
# 15 / 3 * 2      = 10.0
# 15 // 4 + 2     = 5
# 11 // 4 + 2     = 4
# 17 % 5 * 2      = 4
```

DO YOU KNOW?

In python "f" before string is used for string interpolation (embedding variables to expression) inside a string.

TIDBIT

Exponentiation is right associative, meaning that when there are multiple ** operators, python evaluates them from right to left.

Summary

- Python is a high-level, versatile programming language. It is known for simplicity and readability.
- Python is used in web development, data analysis and Artificial Intelligence etc.
- Programming involves giving precise instructions to computers. Programs tell computers how to perform specific tasks.
- Download Python from python.org (latest version)
- Installation steps: Download → Run installer → Check "Add Python to PATH" → Install
- IDE Options: PyCharm, VS Code, IDLE.
- Online Option: Google Colab (cloud-based, no installation)
- Arithmetic: +, -, *, /, //, %, **
- Comparison: >, <, ==, !=, >=, <=
- Assignment: =, +=, -=, *=, /=, //=, %=
- Logical: and, or, not
- Comments are used to improve code readability

EXERCISE

Multiple Choice Questions

1. **Which of the following steps is NOT part of the basic programming process?**
   (a) Write Code (b) Compile/Interpret
   (c) Execute (d) Ignore Errors

2. **What should you do when installing Python to run it from the command line more easily?**
   (a) Uncheck "Add Python to PATH"
   (b) Choose a different IDE
   (c) Check "Add Python to PATH"
   (d) Install only the IDE

3. **What will be the output of: print(10 // 3)?**
   (a) 3.333 (b) 3
   (c) 1 (d) 4

4. **What is data type of variable x in x="123"**
   (a) Integer (b) Float
   (c) String (d) Boolean

5. **Which operator has the highest precedence in Python?**
   (a) Addition (+) (b) Multiplication (*)
   (c) Exponentiation (**) (d) Parentheses ()

6. **What is the result of 5 > 3 and 2 < 1?**
   (a) True (b) False
   (c) Error (d) None

7. **What does the += operator do?**
   (a) Adds two numbers (b) Compares values
   (c) Adds and assigns (d) Multiplies values

8. **What is the output of the below code?**
   age = 25; print (" Age : " , age)
   (a) Age: 25 (b) 25
   (c) Age (d) age

9. **Which symbol is used for comments in Python?**

(a) // (b) /*

(c) # (d) --

Short Answer Questions

1. Why are comments important in programming?
2. Discuss three basic data types in python.
3. Describe the difference between integer and float data types in Python.
4. What are logical operators and name all three?
5. How do you get user input in Python?
6. What is the purpose of print() statement in Python?
7. What is operator precedence? Give an example.
8. Write any three main rules for naming variables in Python?

Long Answer Questions

1. What are the basic data types in Python? Explain integers, floats, strings, and Booleans with examples.
2. What are arithmetic operators in Python. Provide examples of each operator and explain when you would use them.
3. Explain the input() and print() functions with examples.
4. Explain how comparison and logical operators work together in conditional statements. Provide a practical example.
5. Write a program in Python that will take a name from user and display greeting message to the given name.
6. Write a program in python that takes two numbers from user and display their sum.
7. Write a program in python that five numbers from user and display their average.
8. Solve the following expressions:

a) `8 + 3 * 4 - 2 ** 2`

b) `20 % 7 * 3 + 10 // 3`

c) `2 * 3 ** 2 % 4 + 10 - 6 / 2`