UNIQUE NOTES

COMPUTER — CLASS 10

UNIT 03

INTRODUCTION TO PYTHON PROGRAMMING

Topics Covered

1. Python Programming
2. Python Syntax and Structure
3. Variables in Python
4. I/O Operations
5. Operators & Expressions
6. Operators Precedence

Table of Contents

Introduction
1. Python Programming
2. Python Syntax and Structure
3. Variables in Python
4. I/O Operations
5. Operators & Expressions
6. Operators Precedence
Topic Wise Short Questions (Additional)
Predict the Output
Find the Errors
Topic Wise Multiple Choice Questions (Additional)
Solved Exercise
Short Questions
Long Questions

Unit 03 — Introduction to Python Programming

Introduction

Python is a popular language known for its simplicity and readability, making it ideal for both beginners and professionals. In this unit, we will start with the basics, including setting up your development environment and learning fundamental programming concepts.

Q.1 What is Python programming language, and in which fields is it commonly used? *[10503001]*

★ **Key Points:** *Python Programming Language*

Python is a popular, easy-to-learn high level programming language designed to be readable and user-friendly. It uses simple syntax (like plain English) instead of complex symbols, making it perfect for beginners.

Commonly Used in Different Fields

Python is used to build websites, analyze data, create games, automate tasks, and even power advanced technologies like AI. Its versatility and large community support make it a go-to choice for solving problems in almost any field.

Q.2 What are basic programming concepts, and what are the main steps in writing a computer program? *[10503002]*

★ **Key Points:** *Basic Concept of Programming Language | Basic Step to Write Code*

1. Understanding 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.

Example: *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.*

2. Programming Basics

Computer programming involves the following basic steps to write a program:

- a. Write Code: Create a set of instructions in a programming language.
- b. Compile/Interpret: Translate the code into a form that the computer can understand.
- c. Execute: Run the code to perform the task.
- d. Output: Display the results or perform actions based on the code.

Source Code: Human readable code written in programming languages like C, C++, Python.

Object Code: Machine readable code generated by compiling source code.

Q.3 What steps are required to set up a Python development environment? *[10503003]*

★ **Key Points:** *Integrated Development Environment | How to Download Python?*

1. 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.

2. Steps of Python Installation

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

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). The installer will download automatically.

(Figure reference: Python Home Page — shows python.org navbar with Downloads menu open, listing All releases, Source code, Windows, macOS, Android, Other Platforms, License, Alternative Implementations, and a "Download for Windows" panel with the Python 3.13.7 button.)

b. 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 reference: Setup Process — Python 3.13.7 (64-bit) Setup window showing "Setup Progress" with the pip bootstrap installing and a progress bar.)

c. 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 reference: Installation Process — PyCharm Setup window showing "Installing" with progress bar and "Extract: IntelliJ platform.jewl.ui.jar", a Show details button, and Back/Next/Cancel buttons.)

DID YOU KNOW?

Q. What should you do when installing Python to make it easier to run from the command line, and what is another alternative for running Python programs? [10503003a]

Ans. 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.

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

(Figure reference: Python IDE — "Welcome to PyCharm" start screen with New Project, Open, Get from VCS options and a "Take a quick onboarding tour" prompt.)

e. 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 F5 in VS Code after setting up a task).
- If the output appears (e.g., "Hello, Python!"), your IDE is correctly configured.

3. 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 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.

(Figure reference: Google Colab IDE — a blank Colab notebook "Hello_Word.ipynb" with the toolbar (File, Edit, View, Insert, Runtime, Tools, Help), a single empty code cell prompting "Start coding or generate with AI.")

Q.4 Write the basic syntax of Python. *[10503004]*

★ **Key Points:** *Syntax of Python*

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

(Figure reference: Python Program in Colab — a Colab cell containing print("Hello, Students!") with output "Hello, Students!" shown beneath it.)

```
print("This is my first page")
```

Output: This is my first page

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.

Q.5 What is the purpose of using comments in Python? Explain its types. *[10503005]*

★ **Key Points:** *Comments | Single-Line | Multi-Line*

Comments are the lines that are not executed by the Python interpreter. They are used to provide explanations or notes for the programmer.

Types of Comments

- Start with the # symbol.
- It is used to add a comment that explains one line or step in the code.
- Python skips the part after # when running the program.

Example:

```
# This is a single-line comment
print("Hello,World!")
```

2. Multi-line Comments

- Multiline comments are used when you want to write a comment that covers more than one line.
- You can create them by using triple single quotation marks ''' at the start and end of the comment.
- They are helpful for writing longer notes or explanations that take up several lines.

Example:

```
'''This is a multi-line comment.
   It spans multiple lines.'''

print("Edhi Foundation operates the world's largest volunteer
ambulance network")
```

3. Difference between Single-line Comment and Multi-line Comment

Feature	Single-line Comment	Multi-line Comment
Used for	Explaining one line of code	Explaining multiple lines or long notes
Symbol used	# at the start of the line	Triple single quotes ''' at start and end
How it works	Python ignores everything after # on that line	Python ignores everything between the triple quotes
Best for	Short, quick notes	Longer explanations or blocks of text
Example	# This is a comment	''' This is a comment that takes multiple lines '''

INTERESTING INFORMATION

Q. How are triple quotes used in Python, and are they actually multiline comments? [10503004a]

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

Q.6 Define Variable. Write the rules for naming variables in Python. *[10503006]*

★ **Key Points:** *Variable | Rules for Naming Variable*

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=17
print("Areeda lived for",age,"years")
age=16
print("Hina lived for",age,"years")
```

Output:

```
Areeda lived for 17 years
Hina lived for 16 years
```

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

1. Rules for Naming Variables 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.

Example: *my_variable, _count, var2 are valid; 2var is invalid.*

- Subsequent characters can include letters, digits (0-9), or underscores (_).
- Variable names are case-sensitive, meaning age and Age are considered two different variables.

Example: *myVar, myvar, and MYVAR are treated as different variables.*

- Python's reserved keywords, such as for, while, if, etc., cannot be used as variable names.

Example: *if = 10 is invalid because if is a keyword.*

- Special characters like @, #, $, %, etc., are not allowed in variable names.

Example: *my@var is invalid.*

- There is no strict limit on the length of variable names, but overly long names can reduce readability.

2. Example of Valid and Invalid Variable Names

- Valid: score, player_1, _temp, totalSum
- Invalid: 2nd_place, my-var, for, my#score

DID YOU KNOW?

Q. What are some reserved words in Python, and can they be used as variable names? [10503006a]

Ans. 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 cannot use them as a variable name.

Q.7 What are the basic data types of variables? Explain integers, floats, strings, and Booleans with examples. *[10503007]*

★ **Key Points:** *Data Types | Integer | Float | Strings | Boolean*

In Python, you can create variables of different types to store various kinds of data. There are some common types of variables:

1. Integer (int)

Definition: Stores whole numbers without decimal points.

Example: age = 17

In this case, age is an integer variable storing the value 17.

2. Floating-Point (float)

Definition: Stores decimal numbers.

Example: price = 19.99

In this case, price is a float variable storing the value 19.99

3. String (str)

Definition: Stores text or characters. Strings are enclosed in quotes, either single (' ') or double (" ").

Example: name = "Arshad"

In this case, name is a string variable storing the text "Arshad"

4. Boolean (bool)

Definition: Stores True or False.

Example: is_student = True

In this case, is_student is a boolean variable that stores the value True.

Q.8 How do input() and print() operations work in Python, and how can you handle integer and float inputs? *[10503008]*

OR

Explain the input and print() / output functions with examples.

★ **Key Points:** *Input | Output | I/O Examples*

1. Input and Output Operations

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).

a. 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 key. The text entered by the user is then stored in a variable.

Example: name=input("Enter your name:")

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

b. Output: Use the print() function to display information on the screen. The print() function takes one or more arguments and displays them.

Example: print("Hello, "+name+"!")

This line of code displays a greeting message that includes the user's name.

DID YOU KNOW?

Q. In print() function, which symbol is used between multiple values / variables to be printed?

Ans. In the print() function, comma (,) is used between multiple values or variables to be printed.

DID YOU KNOW?

Q. In Python, what defines the structure or scope of code?

Ans. Indentation defines the structure or scope of code in Python. Incorrect indentation may cause errors.

2. Handling Integer and Float Inputs

To handle numeric inputs, convert the input using int() for integers or float() for floating-point numbers, as the input() function always returns a string.

```
user_age=int(input("Enter your age:"))
print("Your age is:",user_age)
```

Output: Enter your age: 16

Your age is: 16

```
user_height=float(input("Enter your height in meters: "))
print("Your height is",user_height,"meter")
```

Output: Enter your height in meters: 1.5

Your height is: 1.5 meter

Q.9 What is an operator in Python? Explain its types with examples. *[10503009]*

★ **Key Points:** *Operators | Arithmetic | Comparison | Assignment | Logical*

An operator is a symbol that performs a specific operation on one or more operands (values or variables). Operators are used to carry out arithmetic, comparison, logical, assignment, and many other operations in a program. An expression is a combination of variables, numbers, and operators that gives a result.

Example: *2 + 3, a * b, or x > y are all expressions. When the computer runs the code, it calculates the value of the expression. In the expression a + b, the + is an operator that performs addition, and a and b are operands.*

Types of Operators

There are four commonly used types of operators in the Python language.

1. 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.

Example:

```
# Define variables
a=10
b=3
# Perform all arithmetic operations
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 = 3.3333333333333335
```

```
print(a,"//", b,"=",a // b)      # Output: 10 // 3=3
print(a,"%", b,"=",a%b)          # Output: 10 % 3 = 1
print(a,"**", b,"=",a**b)        # Output: 10**3= 1000
```

2. 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.

Example:

```
#Define variable
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 >, <, ==, !=.

3. 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.

Example:

```
# 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 assignments operations including addition, subtractions, 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.

4. Logical Operators

Logical operators are used to combine multiple conditions or expressions in a program. 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.

Example:

```
#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
```

Q.10 Explain how comparison and logical operators work together in conditional statements. Provide a practical example. *[10503010]*

★ **Key Points:** *Operators | Comparison | Logical Operators | and | or | not*

1. Comparison operators are used to compare two values or expressions and return a Boolean value (True or False). Examples include >, <, ==, !=, >=, <=.

2. Logical operators (and, or, not) are used to combine multiple conditions or expressions and return Boolean values based on the evaluation.

Example:

```
# 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
```

Note:

- and returns True only if both conditions are True.
- or returns True if at least one condition is True.
- not inverts a Boolean value.

Q.11 What are arithmetic operators in Python. Provide examples of each operator and explain when you would use them. *[10503011]*

★ **Key Points:** *Operators | Arithmetic | Use of Arithmetic Operators*

Arithmetic operators are used to perform basic mathematical operations on numbers.

Examples:

```
1. Addition (+): Adds two numbers.
a=10
b=3
print(a+b)                      # Output: 13
2. Subtraction (-): Subtracts the second number from the first.
print(a-b)                      # Output: 7
3. Multiplication (*): Multiplies two numbers.
print(a*b)                      # Output: 30
4. Division (/): Divides the first number by the second (returns float).
print(a/b)                      # Output: 3.3333333333333335
5. Floor Division (//): Divides and returns the integer part of the quotient.
print(a//b)                     # Output: 3
6. Modulus (%): Returns the remainder of division.
print(a%b)                      # Output: 1
7. Exponentiation (**): Raises the first number to the power of the second.
print(a**b)                     # Output: 1000
```

When to use:

- Addition, subtraction, multiplication, division used for general calculations.
- Modulus operators are used to find remainders. They are useful in loops, arrays, or checking even/odd numbers.
- Exponentiation used for powers or mathematical formulas.
- Floor division are used when only integer results are needed.

Q.12 What is operator precedence in Python? Explain with examples. *[10503012]*

★ **Key Points:** *Operator Precedence | Priority of Operators | Order of Operations*

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.

1. Parentheses ( ): Highest precedence. Operations inside parentheses are performed first. (3+2)*4 evaluates to 20.
2. Exponentiation (**): Performs power operations next. 2**3 evaluates to 8. i.e. 2+2*2**3=18
3. Multiplication '*', Division '/', and Modulus '%': These operations come next. 4*3 evaluates to 12, 10/2 evaluates to 5.0 and 11%3 evaluates 2.
4. Addition '+' and Subtraction '-': These have lower precedence compared to multiplication and division. i.e. 5+2 evaluates to 7, and 10-4 evaluates to 6.

Example: Consider the following code to show operator precedence:

```
print("=== PYTHON OPEATOR 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% 4 * 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
```

DID YOU KNOW?

Q. What does putting an f before a string do in Python?

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

DID YOU KNOW?

Q. What does it mean that exponentiation (**) is right associative in Python?

Ans. Right associative means Python evaluates ** from right to left. For example, in 2 ** 3 ** 2, it first calculates 3 ** 2 and then 2 ** 9. This is different from most operators, which are left to right.

Topic Wise Short Questions (Additional)

Introduction to Python Programming

Q.1 Why Python is considered beginner-friendly?

Ans. Python is considered beginner-friendly because it has a simple and readable syntax that resembles plain English, making it easy to understand. Its dynamic typing means you don't need to declare variable types explicitly, which reduces complexity for beginners.

Q.2 What role does Python's large community play in its popularity?

Ans. A strong community provides extensive support, tutorials, libraries, and frameworks, enabling problem-solving across diverse fields and accelerating development.

Q.3 How does Python's versatility benefit developers?

Ans. Python supports multiple domains like web development, data analysis, AI, automation, and game development, making it a one-stop solution for varied applications.

Q.4 Why is the name "Python" often misunderstood?

Ans. The name Python comes from the British comedy show Monty Python's Flying Circus, not the snake, reflecting the language's playful design philosophy.

Q.5 What are the core steps in computer programming?

Ans. Writing code, compiling/interpreting, executing, and producing output. These steps translate human logic into machine-executable instructions.

Basic Python Syntax and Structure

Q.6 Why is an IDE crucial for Python development?

Ans. IDEs like PyCharm or VS Code offer features like syntax highlighting, debugging, and code completion, streamlining development and reducing errors.

Q.7 How does Python's syntax differ from other languages?

Ans. Python uses indentation and minimal symbols (e.g., no semicolons), prioritizing readability over complex syntax.

Q.8 What are comments in Python, and why are they important?

Ans. Comments (single-line # or multi-line ''' ''') explain code logic, improving readability and aiding collaboration.

Q.10 Why are meaningful variable names critical in Python?

Ans. They make code self-documenting, reduce reliance on comments, and ease debugging and maintenance.

Q.11 What are the rules for naming a variable?

Ans. Rules for naming a variable: names must start with a letter or underscore, avoid reserved keywords, use snake_case for readability, and remember names are case-sensitive.

Q.12 How do integers and floats differ in Python?

Ans. Integers (int) are whole numbers, while floats (float) represent decimal values (e.g., age=17 vs. price=19.99).

Q.9 How does dynamic typing enhance Python's flexibility?
Ans. Dynamic typing enhances flexibility because variables can hold any data type without prior declaration, allowing rapid prototyping and adaptable code.

Q.13 Why is input validation important in Python programs?
Ans. It ensures data integrity by converting user inputs (strings) to appropriate types (e.g., int(input())).

Operators and Expressions

Q.14 What is an operator?
Ans. An operator is a symbol or sign used to perform a specific operation on data. For example, + is used for addition, - for subtraction, * for multiplication, and / for division. Operators in programming help us do calculations or compare values.

Q.15 What is an expression?
Ans. An expression is a combination of variables, numbers, and operators that gives a result. For example, 2 + 3, a * b, or x > y are all expressions. When the code runs, it calculates the value of the expression.

Q.16 How do arithmetic operators in Python handle division?
Ans. The / operator returns a float result (e.g., 10 / 3 = 3.33), while the // operator performs floor division, returning the integer part of the quotient (e.g., 10 // 3 = 3).

Q.17 How do comparison operators work in Python?
Ans. They evaluate relationships between values (e.g., ==, >, <=) and return Boolean results (True/False).

Q.18 What is the use of logical operators (and, or, not) in conditions?
Ans. Logical operators are used to combine multiple conditions in programming (e.g., x > 5 and x < 10) for complex decision-making.

Q.19 What is operator precedence in Python?
Ans. Operator precedence tells which operator is evaluated first in an expression. Python follows rules to decide the order of operations like *, +, and -. It helps to get the correct result from a complex expression.

Example (Q.19):

```
result=3+2*5
print(result)
```

Output: 13

Predict the Output

Q.20 *[10503032]*

```
# print("Hello")
print("World")
```

Ou
World

Q.21 *[10503033]*

```
print("Python")
```

Ou
Python

Q.22 *[10503034]*

```
#This is a comment
#print(100)
print(50)
```

Ou
50

Q.23 *[10503035]*

```
print("A")
#print("B")
```

Ou
A

Q.24 *[10503036]*

```
#print("Y")
print("Z")
```

Ou
Z

Q.25 *[10503037]*

```
x=10
y=5
print(x+y)
```

Ou
15

Q.26 *[10503038]*

```
a="Hello"
b="World"
print(a+" "+b)
```

Ou
Hello World

Q.27 *[10503039]*

```
num=7
num=num+3
print(num)
```

Ou
10

Q.28 *[10503040]*

```
x=5
y=x
x=10
print(y)
```

Ou
5

Q.29 *[10503041]*

```
name="Ali"
age=20
print(name,age)
```

Ou
Ali 20

Q.30 *[10503042]*

```
x=y=z=2
print(x,y,z)
```

Ou
2 2 2

Q.31 *[10503043]*

```
a=10
b="10"
print(a+int(b))
```

Ou
20

Q.32 *[10503044]*

```
x=None
print(x)
```

Ou
None

Q.33 *[10503045]*

```
print(10+5)
```

Ou
15

Q.34 *[10503046]*

```
print(10-3*2)
```

Ou
4

Q.35 *[10503047]*

```
print(2**3)
```

Ou
8

Q.36 *[10503048]*

```
y=10
y -=4
print(y)
```

Ou
6

Q.37 *[10503049]*

```
a=5
b=2
print(a//b)
```

Ou
2

Q.38 *[10503050]*

```
print((3+2)*4)
```

Ou
20

Find the Errors

Error Program	Error / How to Fix
Q.39 *[10503051]* `print("World"`	**Syntax Error: The print statement is missing the closing parenthesis.** Fix: print("World")

Q.40 *[10503052]*
`x=10`
`y="5"`
`print(x+y)`
	Type Error: Cannot add int and str.
	Fix: print(x+int(y))

Q.41 *[10503053]*
`a=10`
`b=20`
`print(a<=>b)`
	Syntax Error: <=> invalid in Python.
	Fix: Use a<b, a>b or a==b.

Q.42 *[10503054]* `print(Hello)`	**Name Error: Hello undefined.** Fix: print("Hello")

Q.43 *[10503055]*
`x=10`
`x++`
	Syntax Error: ++ not valid in Python.
	Fix: x+=1

Q.44 *[10503056]* `x="Python`	**Syntax Error: Unclosed string.** Fix: x="Python"
Q.45 *[10503057]* `print("Hi)`	**Syntax Error: Missing closing quote.** Fix: print("Hi")

Q.46 *[10503058]*
`x=5`
`y=2`
`print(x**)`
	Syntax Error: Invalid exponent.
	Fix: print(x**y)

Q.47 *[10503059]*
`if 5>3`
`  print("Yes")`
	Syntax Error: Missing colon.
	Fix: if 5>3:
	`print("Yes")`

Q.48 *[10503060]*
`a=10`
`b=20`
`print(a<>b)`
	Syntax Error: <> invalid. Use !=.
	Fix: print(a!=b)

Q.49 *[10503061]* `print("Hello"+5)`	**Type Error: Cannot add str and int.** Fix: print("Hello"+str(5))

Q.50 *[10503062]*
`a=10`
`b=0`
`c=a%b`
	Zero Division Error: Modulus by zero.
	Fix: b!=0

Q.51 *[10503063]* `print(2**)`	**Syntax Error: Missing exponent.** Fix: print(2**3)
Q.52 *[10503064]*	**Syntax Error: Wrong operator.**

Error Program	Error / How to Fix

`a=10`
`b=20`
`if a=>b:`
	Fix: if a>=b:

Q.53 *[10503065]* `print("Python"+)`	**Syntax Error: Missing value.** Fix: print("Python"+"3")

Q.54 *[10503066]*
`x=5`
`y=2`
`print(x**y))`
	Syntax Error: Extra parenthesis.
	Fix: print(x**y)

Q.55 *[10503067]* `if x=5:`	**Syntax Error: Use == for comparison.** Fix: if x==5:

Q.56 *[10503068]*
`x=10`
`y=20`
`print(x<>y)`
	Syntax Error: <> invalid.
	Fix: print(x!=y)

Topic Wise Multiple Choice Questions (Additional)

Introduction to Python Programming

1. **What is Python?**
   (a) A complex system language
   (b) A high-level, easy-to-learn programming language
   (c) A web design tool
   (d) A database system

2. **Which of the following fields use Python?**
   (a) Game development
   (b) Data analysis
   (c) Artificial Intelligence
   (d) All of these

3. **What is the main purpose of programming?**
   (a) To create instructions for a computer
   (b) To watch videos
   (c) To draw pictures
   (d) To send emails

4. **Which step is part of writing a program?**
   (a) Compile/Interpret
   (b) Play music
   (c) Browse the internet
   (d) Delete files

5. **Where should you download Python from?**
   (a) Any app store
   (b) Third-party site
   (c) https://www.python.org
   (d) Random download links

6. **What does an IDE help with?**
   (a) Watching movies
   (b) Writing and debugging code
   (c) Playing games
   (d) Sending messages

7. **Which tool is used to install Python packages?**
   (a) npm
   (b) apt
   (c) git
   (d) pip

Basic Python Syntax and Structure

8. **What is the purpose of comments in Python?**
   (a) To run code faster
   (b) To delete code permanently
   (c) To explain or note code for better understanding
   (d) To change variable names

9. **Which symbol is used for a single-line comment in Python?**
   (a) #
   (b) //
   (c) /* */

15. **Which is allowed in Python variable names?**
   (a) Spaces
   (b) Hyphens (-)
   (c) Underscores (_)
   (d) Special symbols (#)

16. **What is the recommended case for constants?**
   (a) UPPERCASE with underscores
   (b) Snake_case
   (c) CamelCase

(d) --

10. How can you write a multi-line comment in Python?
(a) Using // at each line
(b) Using triple quotes (''')
(c) Using square brackets [[[ ]]]
(d) Using curly braces { }

11. Which of the following is a valid variable name in Python?
(a) 2ndPlace
(b) @name
(c) my-var
(d) my_variable

12. Which of the following cannot be used as a variable name in Python?
(a) for
(b) score
(c) age
(d) user_name

13. Why should we use meaningful variable names?
(a) To make code shorter
(b) To improve readability and understanding
(c) To increase program speed
(d) To hide data from others

14. Which variable name is invalid?
(a) my_var
(b) _count
(c) MAX_VALUE
(d) 2nd_place

(d) Lowercase

17. What is the data type of the variable in this line: price = 19.99?
(a) float
(b) str
(c) int
(d) bool

18. Which data type represents text in Python?
(a) int
(b) str
(c) float
(d) bool

19. What does the input() function do in Python?
(a) Displays output to the screen
(b) Waits for user input and stores it as a string
(c) Converts data to integer
(d) Ends the program

20. How can you convert user input to an integer?
(a) Using str()
(b) Using float()
(c) Using int()
(d) Using bool()

Operators and Expressions

21. What is an operator in Python?
(a) A value
(b) A symbol that performs an operation
(c) A comment
(d) A function

26. What does the == operator check?
(a) Greater than
(b) Less than
(c) Equality
(d) Assignment

22. Which of the following is an arithmetic operator?
(a) and
(b) ==
(c) +
(d) =

23. Which operator is used for exponentiation?
(a) **
(b) `
(c) //
(d) %

24. Which operator compares two values and returns True or False?
(a) Arithmetic
(b) Assignment
(c) Comparison
(d) Logical

25. Which of the following is a logical operator?
(a) not
(b) -
(c) +
(d) =

27. What is an expression in Python?
(a) A comment
(b) A combination of values, variables, and operators
(c) A variable name
(d) An error message

28. What is the result of the expression 3+ 4 * 2?
(a) 10
(b) 9
(c) 14
(d) 11

29. Which has the highest precedence in Python expressions?
(a) Addition
(b) Parentheses ()
(c) Multiplication
(d) Exponentiation

30. What will be the output of (3 + 2) * 2?
(a) 6
(b) 7
(c) 10
(d) 8

Answers

1	b	**2**	d	**3**	a	**4**	a	**5**	c	**6**	b	**7**	d	**8**	c
9	a	**10**	b	**11**	d	**12**	a	**13**	b	**14**	d	**15**	c	**16**	a
17	a	**18**	b	**19**	b	**20**	c	**21**	b	**22**	c	**23**	a	**24**	c
25	a	**26**	c	**27**	b	**28**	d	**29**	d	**30**	c				

Solved 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) --

Answers

1	d	**2**	c	**3**	b	**4**	c	**5**	d	**6**	b	**7**	c	**8**	a

9	c														

Short Questions

Q.1 Why are comments important in programming?

Ans. Comments are important because it helps to explain the code, make it easier to understand, and improve readability for programmers. They are also useful for documenting logic and making future maintenance easier.

Q.2 Discuss three basic data types in Python.

Ans. int: Used to store whole numbers (e.g., 10, -5). float: Used to store decimal numbers (e.g., 3.14, 19.99). str: Used to store text or characters (e.g., "Hello", "Python").

Q.3 Describe the difference between integer and float data types in Python.

Ans. An integer (int) stores whole numbers without decimals, while a float stores numbers with decimal points. e.g. (age=17, height=5.9)

Q.4 What are logical operators and name all three?

Ans. Logical operators are used to combine or compare conditions. The three logical operators in Python are and, or, and not.

Q.5 How do you get user input in Python?

Ans. User input is obtained using the input() function, which reads input from the user as a string.

Q.6 What is the purpose of print() statement in Python?

Ans. The print() statement is used to display output or information on the screen.

Q.7 What is operator precedence? Give an example.

Ans. Operator precedence determines the order in which operations are performed in an expression. Example: 3 + 4 * 2 results in 11 because multiplication is done before addition.

Q.8 Write any three main rules for naming variables in Python.

Ans. Variable names must start with a letter or an underscore (_). Variable names cannot start with a number. Variable names cannot be Python keywords (e.g., for, if).

Long Questions

Q.1 What are the basic data types in Python? Explain integers, floats, strings, and Booleans with examples. *[10503116]*

Ans. See Long Question No. 7 (detailed answer in Unit content, Q.7).

Q.2 What are arithmetic operators in Python. Provide examples of each operator and explain when you would use them. *[10503117]*

Ans. See Long Question No. 11 (detailed answer in Unit content, Q.11).

Q.3 Explain the input() and print() functions with examples. *[10503118]*

Ans. See Long Question No. 8 (detailed answer in Unit content, Q.8).

Q.4 Explain how comparison and logical operators work together in conditional statements. Provide a practical example. *[10503119]*

Ans. See Long Question No. 10 (detailed answer in Unit content, Q.10).

Q.5 Write a program in Python that will take a name from user and display greeting message to the given name. *[10503120]*

Ans.

```
name = input("Enter your name: ")
print("Hello,", name)
```

Output

```
Enter your name: Arshad Mehmood Shah
Hello, Arshad Mehmood Shah
```

Explanation: This program takes the user's name as input and displays a greeting message using the print() function.

Q.6 Write a program in python that takes two numbers from user and display their sum. *[10503121]*

Ans.

```
num1  = int(input("Enter first number: "))
num2  = int(input("Enter second number: "))
total = num1 + num2
print ("Sum:", total)
```

Output

```
Enter first number: 5
Enter second number: 7
Sum: 12
```

Q.7 Write a program in python that takes five numbers from user and display their average. *[10503122]*

Ans.

```
n1 = float(input("Enter first number: "))
n2 = float(input("Enter second number: "))
n3 = float(input("Enter third number: "))
n4 = float(input("Enter fourth number: "))
n5 = float(input("Enter fifth number: "))
average = (n1 + n2 + n3 + n4 + n5) / 5
print("Average:", average)
```

Output:

```
Enter first number: 10
Enter second number: 20
Enter third number: 30
Enter fourth number: 40
Enter fifth number: 50
Average: 30.0
```

Q.8 Solve the following expressions: *[10503123]*

- a) 8+3*4 -2**2
- b) 20%7*3+10//3
- c) 2*3**2%4+10-6/2

Ans. Solved Expressions (using Python operator precedence):

```
8+3*4 -2**2      #Exponentiation
8+3*4 -4         #Multiplication
8+12- 4          #Addition
20 - 4           #Subtraction
16
```

Ans. 16

```
20%7*3+10//3     #Modulus
6*3+10//3        #Floor Division
6*3+3            #Multiplication
18+3             #Addition
21
```

Ans. 21

```
2*3**2%4+10-6/2  #Exponentiation
2*9%4+10-6/2     #Multiplication
18%4+10-6/2      #Modulus
2+10-6/2         #Division
2+10-3           #Addition
12-3             #Subtraction
```

Ans. 9.0