BTES
bebo Educational Foundation

Assignment-1
Python Basics and Operators

1. Define and explain the difference between *variables* and *constants* in Python. Provide an example of each.

A *variable* is a storage location in memory with a name that can be used to store different values over time. The value of a variable can change during the execution of a program. In Python, variables are dynamically typed, meaning you don’t need to declare the data type explicitly.

Example:-

```
age = 25             # Here, 'age' is a variable
print(age)
```

A *constant* is a name for a value that should not change during the execution of a program. In Python, there is no strict way to enforce constants, but by convention, constants are written in uppercase letters, and they are not meant to be modified once assigned.

Example:-

```
PI = 3.14159     # PI is a constant (by convention, we use uppercase letters)
print(PI)
```

2. List and describe three different *data types* in Python. Provide an example for each.

The three different data types in Python are:

1.Integer(int):-

An **integer** is a whole number, positive or negative, without any decimal point. It is used to represent numerical data that doesn’t require fractions or decimals.

Example:-

```
age = 25
print(type(age))
```

2.Float(float):-

A **float** represents a number that includes a decimal point. It’s used for numbers that may require precision, like measurements or scientific calculations.

Example:-

```
age = 23.5
print(type(age))
```

3.String(str):-

A **string** is a sequence of characters used to represent text. Strings in Python are enclosed in single quotes ('...') or double quotes ("...").

Example:-

```
str = “Aman”
print(type(str))
```

3. Write a Python program that:

- Declares two integer variables with values 10 and 5.
- Performs addition, subtraction, multiplication, and division on these variables.
- Prints the result of each operation.

```
# Declaring two integer variables
num1 = 10
num2 = 5

# Perform basic arithmetic operations
addition = num1 + num2
subtraction = num1 - num2
multiplication = num1 * num2
division = num1 / num2

# Print the results of each operation
print("Addition:", addition)        # Output: 15
print("Subtraction:", subtraction)    # Output: 5
print("Multiplication:", multiplication)  # Output: 50
print("Division:", division)          # Output: 2.0
```

4. What are *literals* in Python? List and explain the four types of literals with one example each.

In Python, **literals** are fixed values that are directly assigned to variables or used in expressions. They represent the basic values of data types in a program, such as numbers, characters, and other simple values.

String Literals

- String literals represent text values. They can be created by enclosing text in single ('...') or double quotes ("...").
- **Example:**

```
greeting = "Hello, World!" # This is a string literal
```

Numeric Literals

- Numeric literals represent numbers. Python supports three types of numeric literals:
  - **Integer** (e.g., 10, -5)
  - **Float** (e.g., 3.14, -2.5)
  - **Complex** (e.g., 2 + 3j, where j represents the imaginary part)
- **Example:**

```
age = 25              # Integer literal
Pi = 3.14159          # Float literal
complex_num = 2 + 3j     # Complex number literal
```

Boolean Literals

- Boolean literals represent logical values and can be either True or False. These literals are often used in conditional statements to control program flow.
- **Example:**

```
is_student = True            # Boolean literal
```

Special Literal (None)

- Python has a special literal called None, which represents the absence of a value or a null value. It's commonly used to indicate that a variable has no value assigned to it or to reset a variable.

- **Example:**

```
data = None         # Special literal indicating the absence of a value
```

5. Declare a variable to store your age, and another variable to store your name. Print a message using these variables, for example: "My name is John and I am 25 years old."

```
# Declare variables

name = "John"

age = 25

# Print the message using the variables

print(f"My name is {name} and I am {age} years old.")
```

6. Explain the difference between *arithmetic operators* and *bitwise operators*. Provide an example of each type.

Arithmetic Operators

Arithmetic operators perform standard mathematical operations on numbers, like addition, subtraction, multiplication, and division. These operators work with whole numbers (integers) and floating-point numbers.

Example:-

```
a = 10
b = 3
# Using arithmetic operators
addition = a + b
modulus = a % b
print("Addition:", addition)      # Output: 13
print("Modulus:", modulus)        # Output: 1
```

Bitwise Operators

Bitwise operators perform operations on the binary representations of integers. They work directly at the bit level, manipulating individual bits within the numbers.

Example:-

```
x = 5       # Binary: 0101
y = 3       # Binary: 0011

# Using bitwise operators
bitwise_and = x & y       # Result: 1 (Binary: 0001)
```

```
bitwise_or = x | y        # Result: 7 (Binary: 0111)

print("Bitwise AND:", bitwise_and)  # Output: 1
print("Bitwise OR:", bitwise_or)    # Output: 7
```

7. Write a Python program that:

- Takes two integer inputs from the user.
- Uses *bitwise AND*, *bitwise OR*, and *bitwise XOR* on these two integers.
- Prints the results of each operation.

```
# Take two integer inputs from the user
num1 = int(input("Enter the first integer: "))
num2 = int(input("Enter the second integer: "))

# Perform bitwise operations
bitwise_and = num1 & num2
bitwise_or = num1 | num2
bitwise_xor = num1 ^ num2

# Print the results of each operation
print("Bitwise AND:", bitwise_and)
print("Bitwise OR:", bitwise_or)
print("Bitwise XOR:", bitwise_xor)
```

Output: -

```
Enter the first integer: 5
Enter the second integer: 3
Bitwise AND: 1
Bitwise OR: 7
Bitwise XOR: 6
```

8. Create a program that converts a given integer number into its binary, octal, and hexadecimal representations. The program should prompt the user to enter the number.

```
# Take an integer input from the user
```

```
num = int(input("Enter an integer: "))
# Convert the integer to binary, octal, and hexadecimal
binary_representation = bin(num)
octal_representation = oct(num)
hexadecimal_representation = hex(num)

# Print the results
print("Binary:", binary_representation)
print("Octal:", octal_representation)
print("Hexadecimal:", hexadecimal_representation)
```

9. What will be the output of the following code? Explain why.

```
a = 8
b = 3
result = a >> b
print(result)
```

□ **Binary Representation of a:**

- a = 8 in binary is 0000 1000 (8-bit representation).

□ **Right Shift by b = 3 Positions:**

- Shifting the bits of 0000 1000 three positions to the right removes the three rightmost bits and introduces zeros on the left:

```
0000 1000 >> 3 = 0000 0001
```

- In binary, 0000 0001 is equal to 1 in decimal.

10. Practical

Write a Python program that:

- Declares two variables x and y, assigns them the values 15 and 8, respectively.
- Swaps their values using only bit-wise XOR operation (without

using a third variable).

- Prints the new values of x and y after swapping.

```
x = 15
 y = 8
print("Before swapping:")
print("x =", x)
print("y =", y)
# Swap values using bitwise XOR
x = x ^ y
y = x ^ y
x = x ^ y
# Print values after swapping
print("\nAfter swapping:")
print("x =", x)
print("y =", y)
```