21 May

Python Basic - 2

* 1. Create two int type variables, apply addition, subtraction, division and multiplications and store the results in variables. Then print the data in the following format by calling the variables:

First variable is & second variable is . Addition: + =

Subtraction: - = Multiplication: \* = Division: / =

Ans # Create two int type variables

first\_variable = 10

second\_variable = 5

# Perform addition, subtraction, multiplication, and division

addition\_result = first\_variable + second\_variable

subtraction\_result = first\_variable - second\_variable

multiplication\_result = first\_variable \* second\_variable

division\_result = first\_variable / second\_variable

# Print the results in the desired format

print(f"First variable is {first\_variable} & second variable is {second\_variable}.")

print(f"Addition: + = {addition\_result}")

print(f"Subtraction: - = {subtraction\_result}")

print(f"Multiplication: \* = {multiplication\_result}")

print(f"Division: / = {division\_result}")

* 1. What is the difference between the following operators:

(i) ‘/’ & ‘//’

(ii) ‘\*\*’ & ‘^’

Ans

(i) ‘/’ & ‘//’

1. `/` (Division Operator):

- The `/` operator performs regular division between two numbers and always returns a floating-point result.

- If both operands are integers, the result will be a floating-point number even if the division yields an integer value.

- Example: `10 / 5` returns `2.0`, `10 / 3` returns `3.3333333333333335`, and `8 / 4` returns `2.0`.

2. `//` (Floor Division Operator):

- The `//` operator performs floor division between two numbers, which means it returns the result rounded down to the nearest integer (towards negative infinity).

- If both operands are integers, the result will be an integer.

- Example: `10 // 5` returns `2`, `10 // 3` returns `3`, and `8 // 4` returns `2`.

(ii) `\*\*` and `^`:

1. `\*\*` (Exponentiation Operator):

- The `\*\*` operator is used for exponentiation, raising a number to a power.

- Example: `2 \*\* 3` returns `8`, `3 \*\* 2` returns `9`, and `2 \*\* 0` returns `1`.

2. `^` (Bitwise XOR Operator):

- In Python, the `^` operator is used for bitwise XOR (exclusive OR) operations on integers.

- It performs bitwise XOR of the binary representation of two integers.

- Example: `10 ^ 5` returns `15` because `10` in binary is `1010` and `5` in binary is `0101`, and `1010 ^ 0101` results in `1111`, which is `15` in decimal.

* 1. List the logical operators.

Ans

1. `and`: The logical AND operator returns True if both operands are True, otherwise, it returns False.

2. `or`: The logical OR operator returns True if at least one of the operands is True, otherwise, it returns False.

3. `not`: The logical NOT operator returns the opposite boolean value of the operand. If the operand is True, it returns False, and if the operand is False, it returns True.

* 1. Explain right shift operator and left shift operator with examples

Ans

1. Right Shift Operator (`>>`):

The right shift operator shifts the bits of a binary number to the right by a specified number of positions. It essentially divides the number by 2 raised to the power of the shift amount (rounded down to the nearest integer).

Syntax: `number >> n`

Example:

```

# Shifting 8 (binary 1000) two positions to the right

result = 8 >> 2

print(result)

# Output: 2

```

Explanation:

- Binary representation of `8` is `1000`.

- Shifting `8` two positions to the right results in `10` in binary, which is `2` in decimal.

2. Left Shift Operator (`<<`):

The left shift operator shifts the bits of a binary number to the left by a specified number of positions. It essentially multiplies the number by 2 raised to the power of the shift amount.

Syntax: `number << n`

Example:

```

# Shifting 5 (binary 101) three positions to the left

result = 5 << 3

print(result)

# Output: 40

```

Explanation:

- Binary representation of `5` is `101`.

- Shifting `5` three positions to the left results in `101000` in binary, which is `40` in decimal.

.

* 1. Create a list containing int type data of length 15. Then write a code to check if 10 is present in the list or not.

Ans

my\_list = [1, 5, 3, 8, 10, 20, 15, 12, 17, 25, 6, 9, 13, 11, 7]

# Check if 10 is present in the list

if 10 in my\_list:

print("10 is present in the list.")

else:

print("10 is not present in the list.")