DAY 2 **[26 - JUNE - 2025]**

PYTHON OPERATORS

An operator is a symbol that represents an operation that may be performed on one or more operands.

An operand is a value that a given operator is applied to.

Example: 4+ (3*k)

+,* are operator and 4, 3, k are operands.

Different forms of operators:

- Unary operator:
  A Unary arithmetic operator performs mathematical operations on one operand only.
- Binary operators:
  A binary operator operates on two operands.

Types of Operators

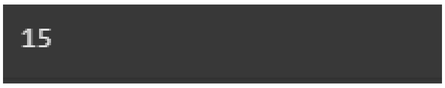
1. Arithmetic Operator
   Arithmetic operators are basic mathematical operations.
2. Relational Operator
   Relational operators are also called as Comparison operators. It is used to compare values. Its either return True or False according to condition.
3. Logical Operator
   Logical operator are typically used with Boolean (logical) values. They allow a program to make a decision based on multiple condition.
4. Bitwise Operator
   Bitwise operator act on operands as if they are string of binary digits. It operates bit by bit.
5. Assignment Operator
   Assignment operators are used to assign values tovariables.
6. Special Operator
   Python offers some special operators like identity operator and the membership operator.
   - Identity Operator – is and is not are the identity operators.
   - Membership operator – in and not in are the membership operators.

PRACTICE QUESTIONS:

1. You're creating a program to manage a zoo's animal population. Declare a variable lion_population with an initial value of 10. The zoo welcomes 5 new lion cubs. Update the lion_population variable and print the total lion population.

```
Code: loin_population = 10
      loin_population += 5
      print(loin_population)
```

Output:

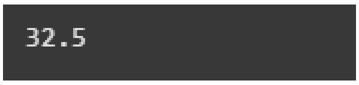


```
15
```

2. You're developing a weather monitoring system. Declare a variable temperature with a value of 25.5 degrees Celsius. Due to a sudden heatwave, the temperature increases by 8 degrees. Update and print the new temperature.

```
Code: temperature = 25.5
      temperature += 8
      print(temperature)
```

Output:

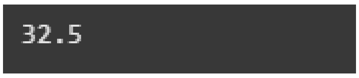


```
32.5
```

3. A science experiment involves tracking the growth of a plant. Declare a variable plant_height with an initial value of 15 centimeters. Over a week, the plant grows 2.5 centimeters taller each day. Update and print the final height of the plant after the week.

```
Code: plant_height =15
      daily_growth = 2.5
      plant_height+= daily_growth*7
      print(plant_height)
```

Output:

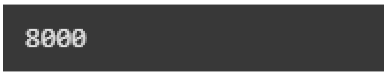


```
32.5
```

4. You're designing a space mission trajectory. Declare variables initial_velocity and acceleration with values 3000 meters per second and 500 meters per second squared respectively. Calculate and print the final velocity after 10 seconds.

```
Code: initial_velocity=3000
      acceleration=500
      time=10
      final_velocity=initial_velocity+(acceleration*time)
      print(final_velocity)
```

Output:

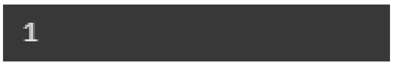


```
8000
```

5. A group of friends is sharing a pizza. Declare a variable pizza_slices with a value of 8. Each friend wants to have an equal number of slices, and there are 5 friends. Calculate and print the maximum number of slices each friend can have without cutting the pizza.

```
Code: pizza_slices = 8
      friends = 5
      print(pizza_slices//friends)
```

Output:

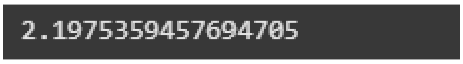


```
1
```

6. You're modeling the movement of a pendulum. Declare a variable pendulum_length with a value of 1.2 meters. Calculate and print the period of oscillation (time taken for one complete swing) using the formula \(T = 2\pi \sqrt{\frac{L}{g}} \), where \(\pi \) is pi (approximately 3.14159) and \(g \) is the acceleration due to gravity (approximately \(9.81 \) meters per second squared).

```
Code: import math
      pendulum_length=1.2
      g=9.81
      t=2*math.pi*math.sqrt(pendulum_length/g)
      print(t)
```

Output:



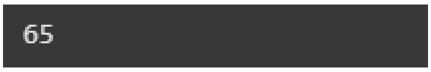
```
2.1975359457694705
```

7. A software company is tracking the number of bugs in their codebase. Declare a variable bug_count with an initial value of 100. After a round of debugging, 35 bugs are fixed. Update and print the new bug_count.

Code:

```
bug_count = 100
fixed =35
new_bugs= bug_count-fixed
print(new_bugs)
```

Output:



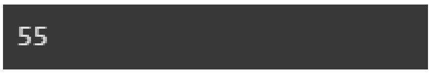
```
65
```

8. You're building a game where players collect gems. Declare a variable gem_count with an initial value of 50. Each time a player finds a gem, 5 gems are added to their collection. Update and print the new gem_count.

Code:

```
gem_count =50
gem_count+=5
print(gem_count)
```

Output:



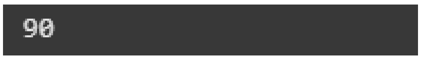
```
55
```

9. A fitness tracker is monitoring a user's heart rate variability (HRV). Declare a variable hrv_index with an initial value of 80. After a relaxation session, the user's HRV improves by 10 points. Update and print the new hrv_index.

Code:

```
hrv_index =80
hrv_index += 10
print(hrv_index)
```

Output:



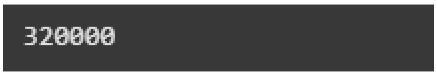
```
90
```

10. You're simulating the growth of a bacterial colony. Declare a variable bacteria_count with an initial value of 5000. Over a day, the colony doubles in size every 4 hours. Update and print the new bacteria_count.

Code:

```
bacteria_count =5000
doubuling=24//4
bacteria_count*=2**doubuling
print(bacteria_count)
```

Output:



```
320000
```