Python Lecture: Arithmetic, Relational, Logical Operators, Branching, and Assignments

1 Arithmetic Operators in Python

Arithmetic operators perform mathematical operations on numbers (integers, floats, etc.). Here are the common ones:

- `+`: Addition (e.g., $5 + 3 = 8$)
- `-`: Subtraction (e.g., $5 - 3 = 2$)
- `*`: Multiplication (e.g., $5 \times 3 = 15$)
- `/`: Division (returns a float, e.g., $5/3 \approx 1.666$)
- `//`: Floor division (integer division, discards remainder, e.g., $5//3 = 1$)
- `%`: Modulus (remainder, e.g., $5\%3 = 2$)
- `**`: Exponentiation (power, e.g., $5 * *3 = 125$)

1.1 Operator Precedence

Python follows a precedence order (like PEMDAS in mathematics). Operators with higher precedence are evaluated first; same-precedence operators are evaluated left to right.

Precedence (highest to lowest):

1. `()`: Parentheses
2. `**`: Exponentiation
3. `*`, `/`, `//`, `%`: Multiplication, division, floor division, modulus
4. `+`, `-`: Addition, subtraction

Examples:

- `2 + 3 * 4`: Multiply first ($3 \times 4 = 12$), then add ($2 + 12 = 14$). Result: 14.
- `(2 + 3) * 4`: Parentheses first ($2 + 3 = 5$), then multiply ($5 \times 4 = 20$). Result: 20.
- `2 ** 3 + 1`: Exponent first ($2^3 = 8$), then add ($8 + 1 = 9$). Result: 9.
- `10 / 2 * 3`: Divide first ($10/2 = 5.0$), then multiply ($5.0 \times 3 = 15.0$). Result: 15.0.
- `10 % 3 // 2`: Modulus first ($10\%3 = 1$), then floor divide ($1//2 = 0$). Result: 0.

2 Sequential Programs Using Arithmetic Operators

Below are five simple sequential programs demonstrating arithmetic operators. Each program takes input, performs calculations, and displays output.

2.1 Program 1: Basic Arithmetic

Description: Takes two numbers and computes their sum, difference, product, and division.

```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
sum_result = num1 + num2
diff_result = num1 - num2
prod_result = num1 * num2
div_result = num1 / num2
print("Sum:", sum_result)
print("Difference:", diff_result)
print("Product:", prod_result)
print("Division:", div_result)
```

2.2 Program 2: Rectangle Area and Perimeter

Description: Calculates the area and perimeter of a rectangle using length and width inputs.

```
length = float(input("Enter length: "))
width = float(input("Enter width: "))
area = length * width
perimeter = 2 * (length + width)
print("Area:", area)
print("Perimeter:", perimeter)
```

2.3 Program 3: Simple Interest

Description: Computes simple interest using principal, rate, and time.

```
principal = float(input("Enter principal amount: "))
rate = float(input("Enter annual rate (as decimal): "))
time = float(input("Enter time in years: "))
interest = (principal * rate * time) / 100
print("Simple Interest:", interest)
```

2.4 Program 4: Powers and Roots

Description: Computes square, cube, and approximate square root of a number using exponentiation.

```
num = float(input("Enter a number: "))
square = num ** 2
cube = num ** 3
sqrt_approx = num ** 0.5
print("Square:", square)
print("Cube:", cube)
print("Square Root (approx):", sqrt_approx)
```

2.5 Program 5: Temperature Conversion

Description: Converts temperature from Celsius to Fahrenheit.

```python
celsius = float(input("Enter temperature in Celsius: "))
fahrenheit = (celsius * 9 / 5) + 32
print("Temperature in Fahrenheit:", fahrenheit)
```

3 Relational and Logical Operators

3.1 Relational Operators

These compare two values and return `True` or `False`:

- `==`: Equal to (e.g., $5 == 5 \rightarrow$ `True`)
- `!=`: Not equal to (e.g., $5! = 3 \rightarrow$ `True`)
- `>`: Greater than (e.g., $5 > 3 \rightarrow$ `True`)
- `<`: Less than (e.g., $5 < 3 \rightarrow$ `False`)
- `>=`: Greater than or equal to (e.g., $5 >= 5 \rightarrow$ `True`)
- `<=`: Less than or equal to (e.g., $5 <= 3 \rightarrow$ `False`)

3.2 Logical Operators

These combine boolean expressions:

- `and`: True if both operands are True (e.g., $(5 > 3)$ `and` $(2 < 4) \rightarrow$ `True`)
- `or`: True if at least one operand is True (e.g., $(5 > 3)$ `or` $(2 > 4) \rightarrow$ `True`)
- `not`: Inverts the boolean value (e.g., `not` $(5 > 3) \rightarrow$ `False`)

Example: `x > 0 and x < 10` checks if `x` is between 1 and 9.

4 Branching Structures in Python

Branching allows programs to execute different code based on conditions, using indentation to define blocks. Conditions use relational and logical operators.

4.1 if Statement

Executes code if the condition is `True`; skips it otherwise.

Structure:

```python
if condition:
    # code if True
# code after if (always runs)
```

How it works: Python evaluates the condition. If `True`, executes the indented block; if `False`, skips to the next non-indented line.

4.2 if-else Statement

Executes one block if `True`, another if `False`.

Structure:

```
if condition:
    # code if True
else:
    # code if False
```

How it works: Exactly one block runs based on the condition.

4.3 if-elif-else Statement

Checks multiple conditions sequentially, executing the first `True` block, or `else` if none are `True`.

Structure:

```
if condition1:
    # code if condition1 True
elif condition2:
    # code if condition1 False and condition2 True
elif condition3:
    # code if previous False and condition3 True
else:
    # code if all above False
```

How it works: Checks conditions in order, stops at the first `True`. `elif` can repeat; `else` is optional.

5 Branching Program Examples

5.1 if Statement Examples

5.1.1 Program 1: Positive Number Check

Description: Checks if a number is positive and prints a message if so.

```
num = float(input("Enter a number: "))
if num > 0:
    print("The number is positive.")
print("Program ended.")
```

5.1.2 Program 2: Voting Eligibility

Description: Checks if a person is eligible to vote (age ≥ 18).

```
age = int(input("Enter your age: "))
if age >= 18:
    print("You are eligible to vote.")
print("Thank you.")
```

5.2 if-else Statement Examples

5.2.1 Program 1: Even or Odd

Description: Determines if a number is even or odd.

```
num = int(input("Enter a number: "))
if num % 2 == 0:
    print("The number is even.")
```

```
else:
    print("The number is odd.")
```

5.2.2 *Program 2: Freezing Check*

Description: Checks if temperature is above freezing (0°C).

```
temp = float(input("Enter temperature in Celsius: "))
if temp > 0:
    print("Above freezing.")
else:
    print("At or below freezing.")
```

5.3 if-elif-else Statement Examples

5.3.1 *Program 1: Grading System*

Description: Assigns a grade based on a score: A (90+), B (80-89), C (70-79), else Fail.

```
score = float(input("Enter score (0-100): "))
if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
else:
    print("Fail")
```

5.3.2 *Program 2: Number Classification*

Description: Classifies a number as positive, negative, or zero.

```
num = float(input("Enter a number: "))
if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")
```

6 Assignments

Below are five assignments to test your understanding of arithmetic operators, their behavior with data types, and their use in sequential and branching structures.

6.1 Assignment 1: Testing Arithmetic Operators

Task: Write a Python program that takes two numbers (integers or floats) as input and performs all arithmetic operations (+, -, *, /, //, %, **). Print the result of each operation with a clear label. Ensure the program handles division by zero by checking if the second number is zero before performing division or modulus.

```
# Assignment 1: Test arithmetic operators
num1 = float(input("Enter first number: "))
```

```
num2 = float(input("Enter second number: "))
print("Addition:", num1 + num2)
print("Subtraction:", num1 - num2)
print("Multiplication:", num1 * num2)
if num2 != 0:
    print("Division:", num1 / num2)
    print("Floor Division:", num1 // num2)
    print("Modulus:", num1 % num2)
else:
    print("Division: Cannot divide by zero")
    print("Floor Division: Cannot divide by zero")
    print("Modulus: Cannot divide by zero")
print("Exponentiation:", num1 ** num2)
```

6.2 Assignment 2: True/False - Arithmetic Operators and Data Types

Task: For each statement below, determine if it is `True` or `False` based on whether arithmetic operators work with the given data types. Write a Python program that tests each statement by attempting the operation and catching any errors using `try-except`. Print whether each statement is `True` or `False` based on the result.

- Statement 1: Addition (+) works on strings (e.g., `"5" + "3"`).
- Statement 2: Multiplication (*) works on strings and integers (e.g., `"hello" * 3`).
- Statement 3: Division (/) works on strings (e.g., `"10" / "2"`).
- Statement 4: Addition (+) works on integers and floats (e.g., `5 + 3.2`).
- Statement 5: Exponentiation (**) works on floats (e.g., `2.5 ** 2`).

```
# Assignment 2: Test arithmetic operators with data types
print("Statement 1: Addition (+) works on strings")
try:
    result = "5" + "3"  # String concatenation
    print("Result:", result, "-> True (concatenates strings)")
except:
    print("Result: False")

print("\nStatement 2: Multiplication (*) works on strings and integers")
try:
    result = "hello" * 3  # String repetition
    print("Result:", result, "-> True (repeats string)")
except:
    print("Result: False")

print("\nStatement 3: Division (/) works on strings")
try:
    result = "10" / "2"  # Should raise TypeError
    print("Result:", result)
except:
    print("Result: False (division not supported for strings)")

print("\nStatement 4: Addition (+) works on integers and floats")
try:
    result = 5 + 3.2  # Integer + Float
    print("Result:", result, "-> True")
except:
    print("Result: False")
```

```
29
30 print("\nStatement 5: Exponentiation (**) works on floats")
31 try:
32     result = 2.5 ** 2  # Float exponentiation
33     print("Result:", result, "-> True")
34 except:
35     print("Result: False")
```

6.3 Assignment 3: Sequential Structure Program

Task: Write a sequential program that calculates the total cost of items purchased, including tax. Ask the user for the price of an item and the quantity purchased. Calculate the subtotal (price × quantity), apply a 5% tax, and display the subtotal, tax amount, and total cost.

```
1 # Assignment 3: Calculate total cost with tax
2 price = float(input("Enter item price: "))
3 quantity = int(input("Enter quantity: "))
4 subtotal = price * quantity
5 tax_rate = 0.05
6 tax = subtotal * tax_rate
7 total = subtotal + tax
8 print("Subtotal: $", subtotal)
9 print("Tax (5%): $", tax)
10 print("Total: $", total)
```

6.4 Assignment 4: Branching Structure Program

Task: Write a program using an `if-elif-else` structure to determine a discount based on the purchase amount. If the amount is over $100, apply a 10% discount; if over $50, apply a 5% discount; otherwise, no discount. Display the original amount, discount amount, and final amount after discount.

```
1 # Assignment 4: Apply discount based on purchase amount
2 amount = float(input("Enter purchase amount: $"))
3 if amount > 100:
4     discount = amount * 0.10
5 elif amount > 50:
6     discount = amount * 0.05
7 else:
8     discount = 0
9 final_amount = amount - discount
10 print("Original Amount: $", amount)
11 print("Discount: $", discount)
12 print("Final Amount: $", final_amount)
```

6.5 Assignment 5: Combined Sequential and Branching Program

Task: Write a program that combines sequential and branching structures. Ask the user for their annual income and the number of dependents. Calculate a tax amount using a sequential formula (tax = income × 0.15). Then, use branching to adjust the tax: if the number of dependents is 3 or more, reduce the tax by 20%; if 1 or 2, reduce by 10%; otherwise, no reduction. Display the income, initial tax, tax reduction, and final tax.

```
1 # Assignment 5: Calculate tax with dependent-based reduction
```

```
income = float(input("Enter annual income: $"))
dependents = int(input("Enter number of dependents: "))
tax = income * 0.15  # Sequential calculation
if dependents >= 3:
    reduction = tax * 0.20
elif dependents >= 1:
    reduction = tax * 0.10
else:
    reduction = 0
final_tax = tax - reduction
print("Income: $", income)
print("Initial Tax (15%): $", tax)
print("Tax Reduction: $", reduction)
print("Final Tax: $", final_tax)
```