8 September Notes

Tuples

- Ordered, **immutable**.
- Defined with ( ).

```
t1 = (10, 20, 30)
print(t1[0], t1[-1])

person = ("Alice", 21, "Student")
name, age, role = person
print(name, age, role)
```

Sets

- **Unordered, unique values**.
- Defined with { }.

```
s = {1, 2, 3, 3, 2}
print(s)   # {1,2,3}
s.add(4)
s.remove(2)

a, b = {1,2,3}, {3,4,5}
print(a|b)  # union
print(a&b)  # intersection
print(a-b)  # difference
```

Notes – Relational, Logical, and Assignment Operators in Python

🔹 1. Relational Operators

- Used to **compare values**.
- Return result as **True or False**.

Operator	Meaning	Example	Result
>	Greater than	5 > 3	True
<	Less than	5 < 3	False
>=	Greater than or equal to	5 >= 5	True
<=	Less than or equal to	4 <= 6	True
==	Equal to	7 == 7	True
!=	Not equal to	7 != 3	True

🔹 2. Logical Operators

- Used to combine multiple conditions.

Operator	Meaning	Example	Result
and	True if **both** conditions are True	(5 > 3 and 2 < 4)	True
or	True if **any one** condition is True	(5 > 3 or 2 > 4)	True
not	Reverses the result	not (5 > 3)	False

✅ **Example:**

```
print(3 > 5 and 1 <= 7)  # False (since 3>5 is False)
```

Truth Table – AND & OR

A	B	A AND B	A OR B
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

✅ Example with NOT:

NOT(True) = False

NOT(False) = True

◆ 3. Assignment Operators

- Used to assign values to variables.

Operator	Meaning	Example	Equivalent
=	Assign value	a = 5	a = 5
+=	Add and assign	a += 2	a = a + 2
-=	Subtract and assign	a -= 2	a = a - 2
*=	Multiply and assign	a *= 2	a = a * 2
/=	Divide and assign	a /= 2	a = a / 2
//=	Floor divide and assign	a //= 2	a = a // 2
%=	Modulus and assign	a %= 2	a = a % 2
**=	Exponent and assign	a **= 2	a = a ** 2

◆ 4. Program 1: Eligibility for Voting in India

```
age = int(input("Enter the age: "))

nat = input("Enter nationality: ")
```

```
region = nat.lower()


if age >= 18 and region == "india":
  print("Eligible")
else:
  print("Not Eligible")
```

Explanation:

- age >= 18 ensures person is adult.
- region == "india" ensures nationality is Indian.
- Both conditions combined with and.
- If both are True → "Eligible", else "Not Eligible".

🔹 5. Program 2: Check if Number is Positive, Negative, or Zero

```
num = float(input("Enter the number: "))


if num > 0:
  print("Positive")
else:
  if num < 0:
    print("Negative")
  else:
    print("Zero")
```

Explanation:

- If num > 0 → Positive.
- Else check → If num < 0 → Negative.
- Else (remaining case) → Zero.

🔹 6. Program 3: Student Result Classification

Rules:

- 85–100 → Distinction
- 60–84 → First Class
- 50–59 → Second Class
- 35–49 → Pass
- 0–34 → Fail

✅ Example Code:

```
n = int(input("Enter the number of inputs: "))
marks_list = []  # store all marks

# Step 1: Take inputs
for i in range(1, n+1):
    marks = int(input(f"Enter the marks M{i}: "))
    marks_list.append(marks)

# Step 2: Process and print results
for i, marks in enumerate(marks_list, start=1):
    print(f"Result for M{i} ({marks}): ", end="")
    if 85 <= marks <= 100:
        print("Distinction")
    elif 60 <= marks < 85:
        print("First Class")
    elif 50 <= marks < 60:
        print("Second Class")
    elif 35 <= marks < 50:
        print("Pass")
    elif 0 <= marks < 35:
        print("Fail")
    else:
        print("Invalid Marks")
```

Explanation:

1. User enters how many students (n).
2. Marks of each student are stored in marks_list.
3. Loop checks each student's marks with conditions:
    - if 85 <= marks <= 100 → Distinction
    - elif 60 <= marks < 85 → First Class
    - elif 50 <= marks < 60 → Second Class
    - elif 35 <= marks < 50 → Pass
    - elif 0 <= marks < 35 → Fail
    - Else → Invalid marks entered.

💡 **Sample Output:**

```
Enter the number of inputs: 5
Enter the marks M1: 30
Enter the marks M2: 45
Enter the marks M3: 55
Enter the marks M4: 75
Enter the marks M5: 90
Result for M1 (30): Fail
Result for M2 (45): Pass
Result for M3 (55): Second Class
Result for M4 (75): First Class
Result for M5 (90): Distinction
```

🔹 **Key Takeaways**

1. **Relational operators** compare values (True/False output).
2. **Logical operators** combine multiple conditions (and, or, not).
3. **Assignment operators** are shorthand updates for variables.
4. Programs demonstrate **real-life condition checks** like voting, number checking, and grading system.