9 September Notes

What is a Dictionary?

- **A dictionary is a collection of key–value pairs.**
- **Keys must be unique and immutable (e.g., string, number, tuple).**
- **Values can be any data type (string, list, int, float, etc.).**
- **Syntax:**

dictionary_name = {key1: value1, key2: value2, ...}

🔹 **Example: Creating and Accessing Dictionary**

```
student = {"name": "Alice", "age": 20, "course": "Python"}


print(student["name"])   # Access value using key
```

Explanation:

- **student["name"] → fetches value of key "name".**
- **Output: Alice.**

🔹 **Modifying Dictionary**

```
student["age"] = 21        # Update existing value
student["city"] = "Delhi"   # Add new key-value pair


print(student)
```

Explanation:

- student["age"] = 21 → updates "age" from 20 to 21.
- student["city"] = "Delhi" → adds a new key "city".
- Now dictionary becomes:
- {"name": "Alice", "age": 21, "course": "Python", "city": "Delhi"}

🔹 **Dictionary Methods**

1. Getting Keys, Values, Items

```
d = {"a": 1, "b": 2, "c": 3}

print(d.keys())    # dict_keys(['a', 'b', 'c'])

print(d.values())  # dict_values([1, 2, 3])

print(d.items())   # dict_items([('a', 1), ('b', 2), ('c', 3)])
```

Explanation:

- .keys() → returns all keys.
- .values() → returns all values.
- .items() → returns key-value pairs as tuples.

---

2. Removing Elements

```
d.pop("b")   # removes key "b" and its value

print(d)
```

Explanation:

- pop("b") removes the entry "b": 2.
- Dictionary becomes: {"a": 1, "c": 3}.

---

3. Iterating Over Dictionary

```
for k, v in d.items():
    print(k, v)
```

Explanation:

- .items() returns key-value pairs.
- Loop prints each key with its value.
- Output:
- a 1 c 3

Bitwise Operators, Nested If-Else & Loops in Python

🔹 Bitwise Operators

Bitwise operators work on numbers at the **binary level**.

Example:

```
# Binary representation:
# 12 = 1100
#  5 = 0101


print(12 & 5)   # AND → 0100 = 4
print(12 | 5)   # OR  → 1101 = 13
print(3 >> 2)   # Right shift
print(3 << 2)   # Left shift
```

Explanation:

- 12 & 5 → 1100 & 0101 = 0100 → 4
- 12 | 5 → 1100 | 0101 = 1101 → 13
- 3 >> 2 → shift right → 0000 → 0
- 3 << 2 → shift left → 1100 → 12

🔹 Nested If-Else

Syntax:

```
if condition1:        # Outer if
   if condition2:     # Inner if
      statement
   else:
      statement
```

```
else:
    statement
```

Example:

```
n = int(input("Enter a number: "))


if n >= 0:
    if n > 0:
        print("+ve")
    else:
        print("zero")
else:
    print("-ve")
```

Explanation:

- First check → if number is non-negative.
- Inside, another check → if greater than 0 → print +ve, else → zero.
- If outer condition fails → print -ve.

🔹 Loops in Python

Loops are used to execute code **repeatedly** until a condition is met.

Types:

1. **For loop**
2. **While loop**
3. **Nested loop** (loop inside loop)

1) For Loop with Sequence

```
sub = "python"
for i in sub:
    print(i)
```

Explanation:

- Iterates through each character of "python".
- Prints:

p

y

t

h

o

n

2) For Loop with List

```
colors = ['black', 'white', 'red', 'yellow', 'orange']
for i in colors:
  print(i)
```

Explanation:

- Iterates through list colors.
- Prints each color name.

3) For Loop with enumerate()

```
sub = 'python'
for i in enumerate(sub):
  print(i)
```

Explanation:

- enumerate() → gives index + value.
- Output:

(0, 'p')

(1, 'y')

(2, 't')

(3, 'h')

(4, 'o')

(5, 'n')

4) For Loop with range()

```
for i in range(1, 11, 1):   # start=1, stop=11, step=1
   print(i)
```

Explanation:

- Prints numbers from **1 to 10**.

5) Printing with Extra Statement

```
for i in range(5):
   print(i)
   print("good morning")
```

Explanation:

- Runs loop 5 times → prints i and "good morning" each time.

6) Accessing Index & Value

```
sub = 'python'
for i in range(0, 6, 1):
   print(i, sub[i])
```

Explanation:

- Iterates index from 0 → 5.
- Prints index and character.

0 p

1 y

2 t

3 h

4 o

5 n

7) Printing Even Numbers (1 to 20)

```
for i in range(2, 21, 2):
  print(i)
```

Explanation:

- Start at 2, step = 2 → prints only even numbers.

8) Check Odd/Even (1 to 10)

```
for i in range(1, 11, 1):
  if i % 2 == 0:
    print(i, "= even number")
  else:
    print(i, "= odd number")
```

Explanation:

- Checks remainder with % 2.
- Even → remainder 0, Odd → remainder 1.

🔹 **Key Takeaways**

- **Bitwise operators** work on binary numbers.
- **Nested if-else** allows decision making inside another decision.
- **Loops** help repeat tasks efficiently.
- range() is powerful for number sequences.
- enumerate() is useful to get index + value together.