# Day 10 Notes

## Advanced Python Concepts- Enumerate

In python, index starts from 0

### 1. enumerate()

enumerate() adds a counter to an iterable and returns it as an enumerate object.

| **Syntax:** |
|---|
| enumerate(iterable, start=0) |

### Example:

```
fruits = ["apple", "banana", "cherry"]

for index, value in enumerate(fruits):
   print(index, value)
```

### Output:

```
0 apple
1 banana
2 cherry
```

### Why use it?

- When you need both index and value while looping.
- Cleaner than using range(len(list)).

---

### 2. delattr()

**delattr()** deletes an attribute from an object.

| **Syntax:** |
|---|
| delattr(object, attribute_name) |

**Example:**

```
class Student:
    name = "Sriya"

delattr(Student, "name")
```

---

### 3. isinstance()

isinstance() checks whether an object belongs to a specific class.

**Syntax:**

```
isinstance(object, class)
```

**Example:**

```
x = 10
print(isinstance(x, int))  # True
```

---

### 4. issubclass()

issubclass() checks whether a class is a subclass of another class.

**Syntax:**

```
issubclass(child_class, parent_class)
```

**Example:**

```
class Animal:
    pass

class Dog(Animal):
    pass
```

```
print(issubclass(Dog, Animal))  # True
```

---

## 5. Miscellaneous Built-in Functions

### I. globals()

Returns a dictionary of the current global symbol table.

```
x = 10
print(globals())
```

Used to access global variables dynamically.

---

### II. locals()

Returns a dictionary of the current local symbol table.

```
def test():
    a = 5
    print(locals())

test()
```

---

### III. callable()

Checks whether an object can be called like a function.

```
def greet():
    pass

print(callable(greet))  # True
print(callable(10))     # False
```

---

### IV. eval()

Evaluates a string as a Python expression.

```
x = "5 + 3"
print(eval(x))  # 8
```

Be careful: eval() can be dangerous if used with user input.

---

## 6. Exceptional Handling

Exception handling prevents program crashes and handles runtime errors gracefully.

---

### I. try

Code that may cause an error is placed inside the try block.

### II. except

Handles the error if it occurs.

### III. finally

Always executes, whether an error occurs or not.

---

### Example:

```
try:
   num = int(input("Enter a number: "))
   result = 10 / num
except ZeroDivisionError:
   print("Cannot divide by zero!")
except ValueError:
   print("Invalid input!")
finally:
   print("Execution completed.")
```

---

## 7. del

Deletes objects or variables.

```
x = 10
del x
```

After deletion, accessing x will cause an error.

---

## 8. Garbage Collection (gc)

Python automatically manages memory using **Garbage Collection (GC)**.

Garbage collection removes unused objects to free memory.

---

### I. gc.collect()

Manually triggers garbage collection.

```
import gc
gc.collect()
```

---

### II. gc

The gc module provides functions to control garbage collection.

```
import gc
print(gc.get_count())
```

---