Python Day 21

Inheritance & List Slicing

Why Day 21 Matters

Till now, you created classes
Now you learn how to:
- Reuse code
- Extend existing classes
- write cleaner, scalable program

This is core OOP, interview + real-world level

Part 1: Inheritance (Very Important)

What is Inheritance?

Inheritance allows one class to take features of another class

Think like this:-
"child class borrows from parent class and ~~cannot~~ a can add extra powers."

Basic Structure

```
class parent:
        pass
class child (parent):
        pass
```

- parent → Base / Super class
- Child → Derived / Sub class

Simple Example

```
class Animal:
    def __init__(self):
        self.num_eyes = 2
    def breathe(self):
        print("Inhale, Exhale.")
```

Now we inherit it

```
class Fish(Animal):
    def swim(self):
        print("Moving in water.")
```

What fish gets automatically:

(✓) num_eyes
(✓) breathe()
(✓) plus its own swim()

Using parent's constructor (super())

if child has its own constructor, parent constructor Does Not run automatically

(✗) wrong way:

```
class Fish(Animal):
    def __init__(self):
        self.fins = True
```

num_eyes will be missing (✗)

(✓) CORRECT way: super()

```
class Fish(Animal):
    def __init__(self):
        super().__init__()
        self.fins = True
```

Rule to remember

Always call super().init() if parent has important setups

(*) Method Overriding

child can change parent's behavior

parent method:

```
def breathe(self):
    print("Inhale exhale.")
```

child overrides it:

```
class Fish(Animal):
    def breathe(self):
        super().breathe()
        print("Doing this underwater.")
```

super() lets you reuse + extend parent log

Key Inheritance Rule

child class has access to:

- parent variables
- parent methods

(✓) Parent does not know about child

(✓) super() refers to immediate parent

(✓) Overriding replaces parent method

Why inheritance is Used in Game (Snake project)

- Create reusable logic
- Avoid repeating Code
- Make game object cleaner

Example

- Animal → base logic
- fish, Dog, Bird → Special behaviors

Part 2: List Slicing (Very important & Easy)

What is list Slicing?
Extracting a portion of a list or string

Syntax
    list [start : end]

- Start → included
- end → excluded

Basic Examples

```
numbers = [0, 1, 2, 3, 4, 5]
numbers [1:4] # [1, 2, 3]
numbers [:3]  # [0, 1, 2]
numbers [2:]  # [2, 3, 4, 5]
numbers [:]   # full copy
```

Negative Index Slicing

```
number [-3:] # [3,4,5]
number [:-2] # [0,1,2,3]
```

Negative index start from END

Step Size (Advanced but Important)

```
numbers [start : end : step]
numbers [::2] # [0,2,4]
numbers [::-1] # reversed list
```

- [::-1] is a python favorite

String Slicing (Same Rules!)

```
text = "Maggie"
text [0:3] # Mag
text [::-1] # eiggaM
```

why Slicing is powerful
- No loops needed
- cleaner code
- faster execution
- Used heavily in data handling

Part - 3 : Day 21 Snake Game logic

Snake Game using :
- Inheritance
- Cleaner class structure

Page No. ____
Date

Example Concept:

```
class Food (Turtle):
    def _init (self):
        super(). _init_()
        self. shape ("circle")
```

→ Food inherits from Turtle
→ No need to rewrite movement or graphic Code

Common Mistake (Very Important)
(x) forgetting super(). init ()
(x) Overriding without calling parent logic
(x) Confusing slicing end index
(x) Thinking child constructor runs parent automatically

final Revision cheat sheet
Inheritance:
- class child (parent)
- Use super() to call parent methods
- Override to customize behavior

Slicing:
- list [start : end]
- End index is excluded
- [: : -1] reverses sequence

One-line Memory Trick

Inheritance = reuse behavior
Slicing = cut without loops