🚀 21-Day Python Challenge

Day 15 – Advanced Object-Oriented Programming (Advanced OOP)

🎯 Day 15 Learning Objectives

By the end of today's class, students will be able to:

- Understand **Inheritance**.
- Understand **Polymorphism**.
- Understand **Encapsulation**.
- Build simple real-world OOP programs.

📘 What is Advanced OOP?

Advanced OOP helps us create programs that are:

- ✅ Easy to reuse
- ✅ Easy to maintain
- ✅ Easy to understand
- ✅ Used in real software companies

Today we will learn three important concepts:

- Inheritance
- Polymorphism
- Encapsulation

📘 1. Inheritance

What is Inheritance?

Inheritance means one class can use the properties and methods of another class.

Real-Life Example

Think about a family.

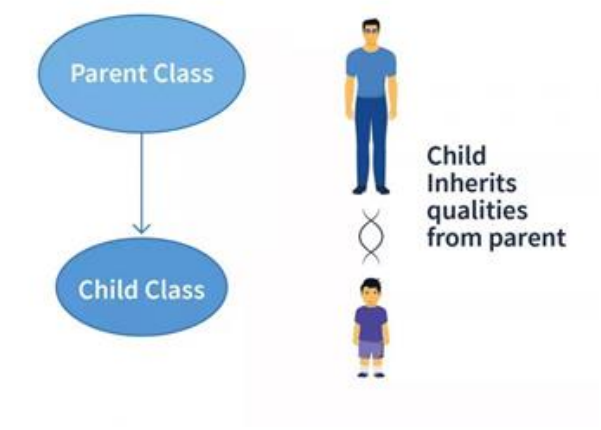
```
Father
↓
```

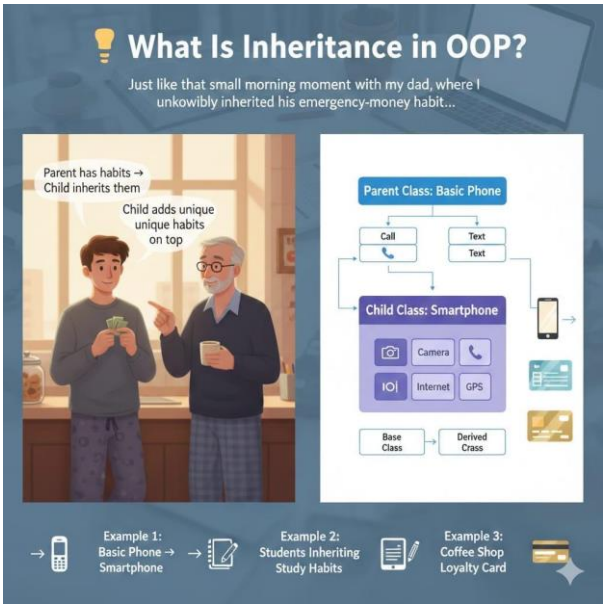
```
Son
↓
Grandson
```

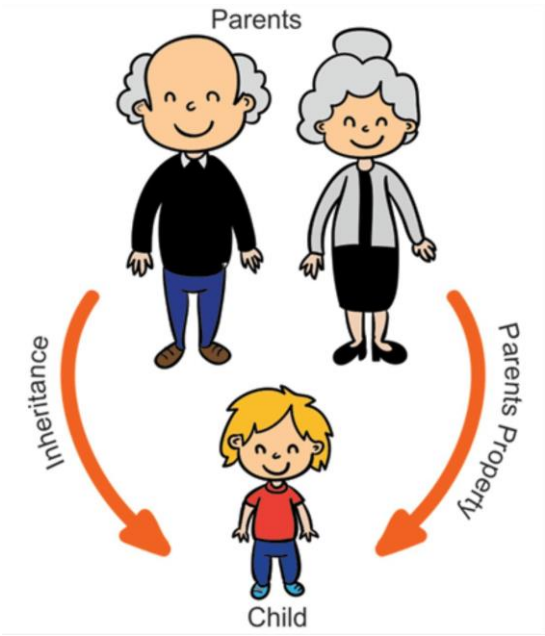
The son inherits many features from the father.

Python classes work the same way.

📷 Picture 1 – Inheritance







Parent Class

```
class Animal:

    def sound(self):
        print("Animals make sounds.")
```

Child Class

```
class Dog(Animal):

    def bark(self):
        print("Dog Barks")
```

Creating Object

```
dog = Dog()

dog.sound()

dog.bark()
```

Output

```
Animals make sounds.
Dog Barks
```

Step-by-Step

```
Animal Class

↓

Dog Class

↓

Dog gets sound()

↓

Dog adds bark()
```

Real-Time Example

```
class Person:

    def details(self):
        print("This is a Person")

class Student(Person):

    def study(self):
        print("Student is Studying")

student = Student()

student.details()
student.study()
```

Output

```
This is a Person
Student is Studying
```

📓 2. Polymorphism

What is Polymorphism?

One method can behave differently for different objects.

Example:

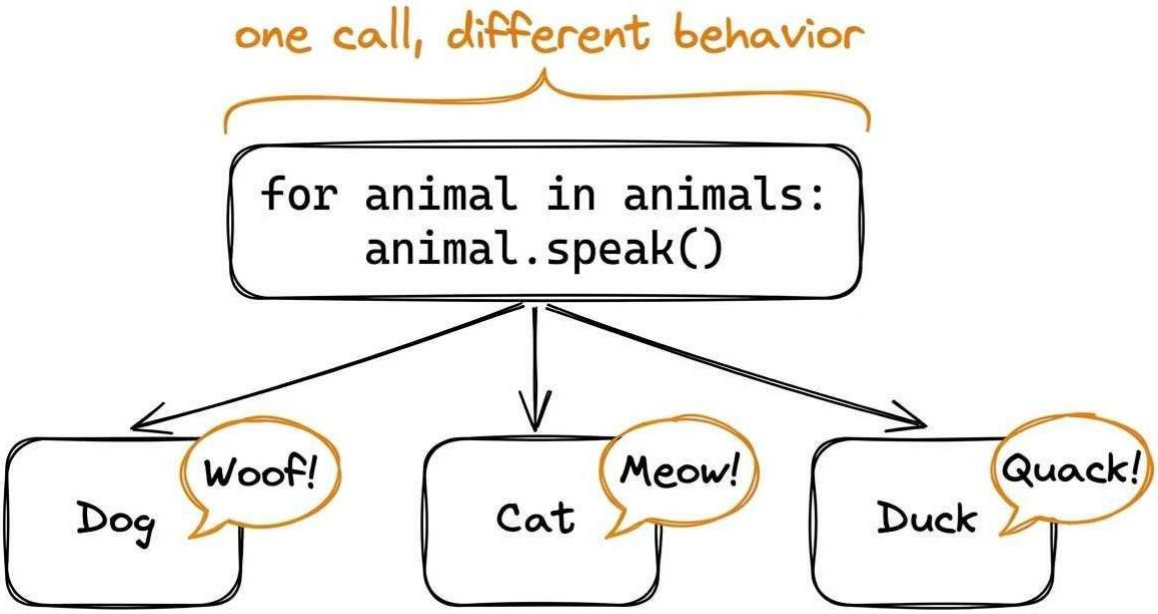
A person says **"Speak"**

- Dog → Bark
- Cat → Meow
- Cow → Moo

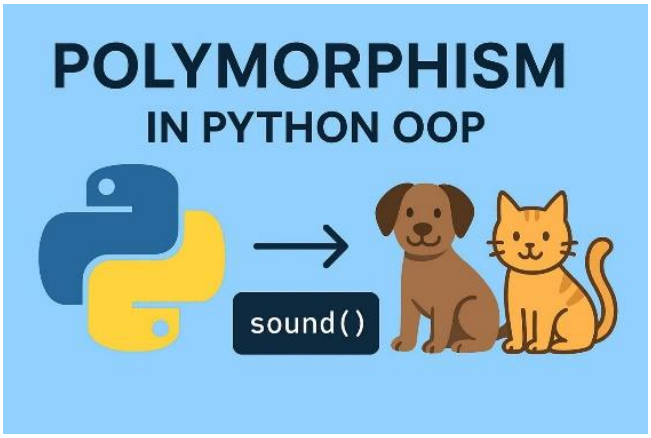
Same command…

Different responses.

📷 Picture 2 – Polymorphism




```python
# The Parent class
class Animal:
    def speak(self):
        return "This animal makes a sound"

# Children classes overriding the parent
class Dog(Animal):
    def speak(self):
        return "Woof!"

class Cat(Animal):
    def speak(self):
        return "Meow!"

# Using polymorphism
whale = Animal()
dog = Dog()
cat = Cat()

print(whale.speak()) # This animal makes a sound
print(dog.speak())  # Woof!
print(cat.speak())  # Meow!
```



Example

```
class Dog:

    def sound(self):
        print("Dog says Bow Bow")

class Cat:

    def sound(self):
        print("Cat says Meow")

dog = Dog()
cat = Cat()

dog.sound()
cat.sound()
```

Output

```
Dog says Bow Bow
Cat says Meow
```

Notice:

Both classes have **sound()**

But the output is different.

Another Example

```
class Car:

    def move(self):
        print("Car is Running")

class Plane:

    def move(self):
        print("Plane is Flying")

car = Car()
plane = Plane()

car.move()
plane.move()
```

Output

```
Car is Running
Plane is Flying
```

📓 3. Encapsulation

What is Encapsulation?

Encapsulation means

Keeping data and methods together inside one class.

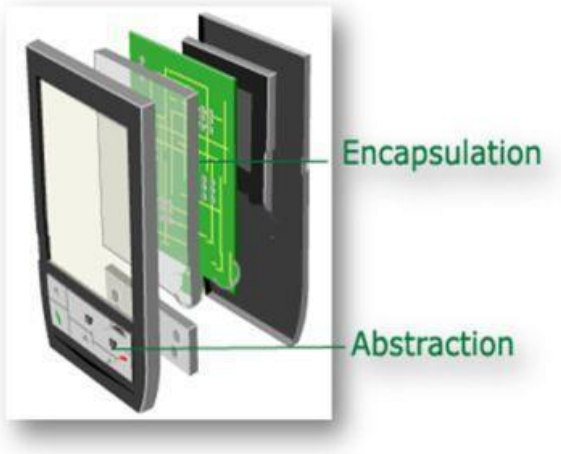
Think of a **Mobile Phone**.

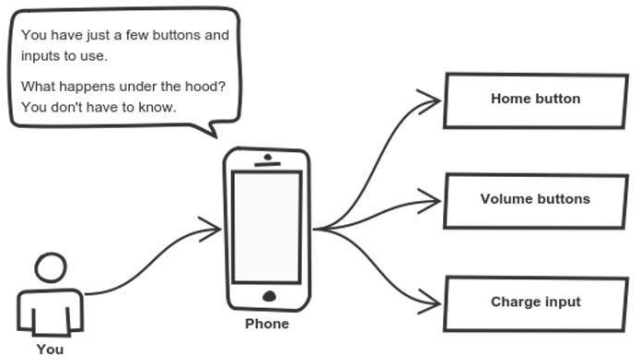
You can press buttons and use apps,

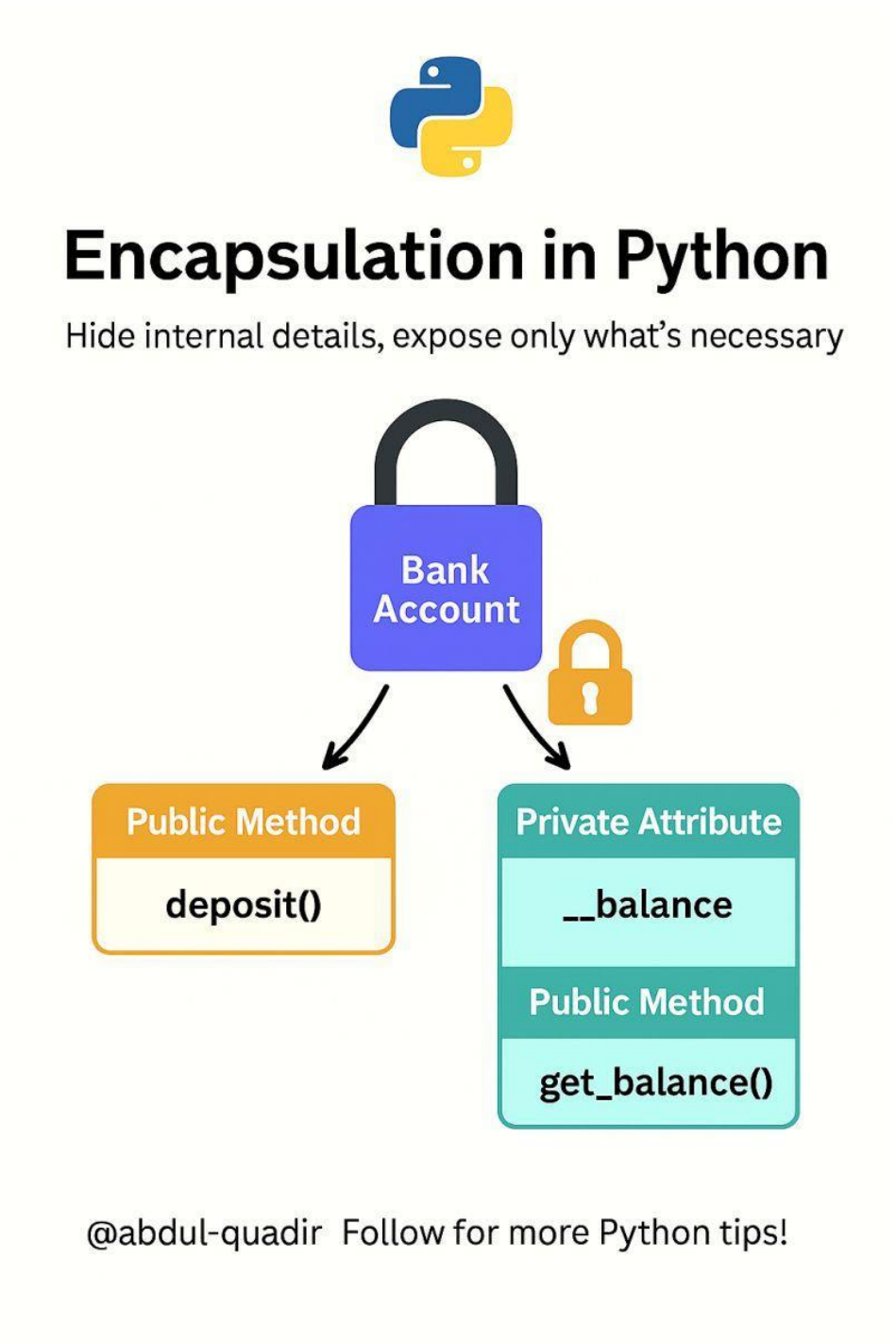
But you cannot directly touch the internal circuits.

The phone hides its internal details.

📷 Picture 3 – Encapsulation







Example

```
class Student:
    def __init__(self):
        self.name = "Rahul"
student = Student()
print(student.name)
```

Private Variable

```
class Bank:

    def __init__(self):

        self.__balance = 5000

bank = Bank()

# print(bank.__balance)
```

Output

```
AttributeError
```

Python protects the variable because it starts with __.

Mini Project

Student Information

```
class Person:

    def display(self):
        print("Person Details")

class Student(Person):

    def study(self):
        print("Learning Python")

student = Student()

student.display()
student.study()
```

Summary Table

Concept	Meaning
Inheritance	One class uses another class's features
Polymorphism	Same method, different behavior
Encapsulation	Protect data inside a class

Assignment

Write programs for:

- Animal → Dog
- Vehicle → Bike
- Person → Teacher
- Student Information
- Employee Information

Interview Questions

1. What is Inheritance?
2. What is a Parent Class?
3. What is a Child Class?
4. What is Polymorphism?
5. What is Encapsulation?
6. Why do we use OOP?

🌟 Day 15 Success Message

Congratulations! 🎉

Today you learned the three important pillars of Object-Oriented Programming:

- ✅ Inheritance
- ✅ Polymorphism
- ✅ Encapsulation

"Professional software is built using reusable classes. Advanced OOP helps programmers write cleaner, smarter, and more maintainable code."

🚀 21-Day Python Challenge

Day 16 – Python Collections

🎯 Learning Objectives

Students will learn:

- List Comprehension
- Lambda Functions

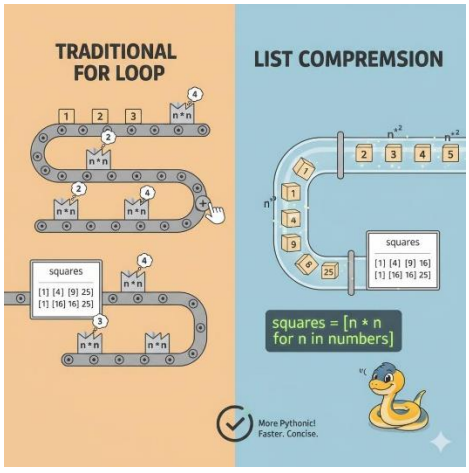
📘 What is List Comprehension?

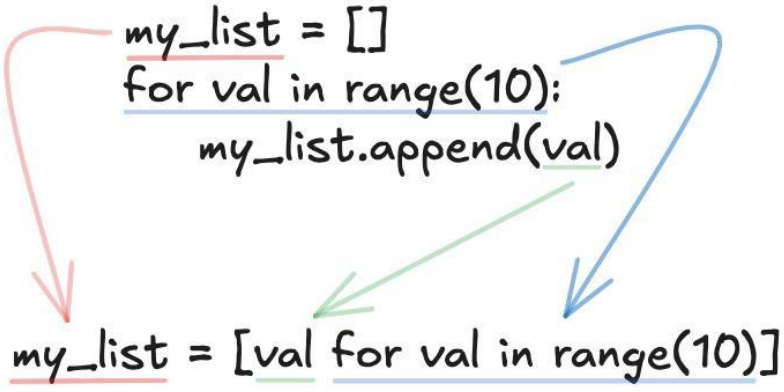
List Comprehension is a **short and easy way** to create a new list.

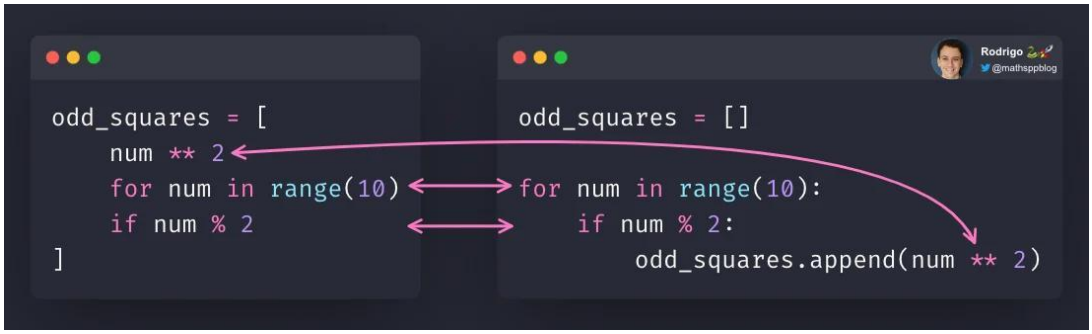
Instead of writing many lines of code,

We can write everything in **one line**.

📷 Picture 4 – List Comprehension







Normal Method

```
numbers = []

for i in range(1,6):

    numbers.append(i)

print(numbers)
```

Output

```
[1, 2, 3, 4, 5]
```

List Comprehension

```
numbers = [i for i in range(1,6)]

print(numbers)
```

Output

```
[1, 2, 3, 4, 5]
```

Notice how much shorter it is.

Step-by-Step

```
range(1,6)

↓

1 2 3 4 5

↓

Store in List

↓

[1,2,3,4,5]
```

Example 2

Create Squares

```
square = [x*x for x in range(1,6)]

print(square)
```

Output

```
[1,4,9,16,25]
```

Example 3

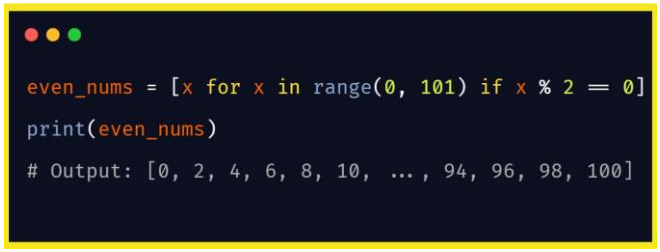
Even Numbers

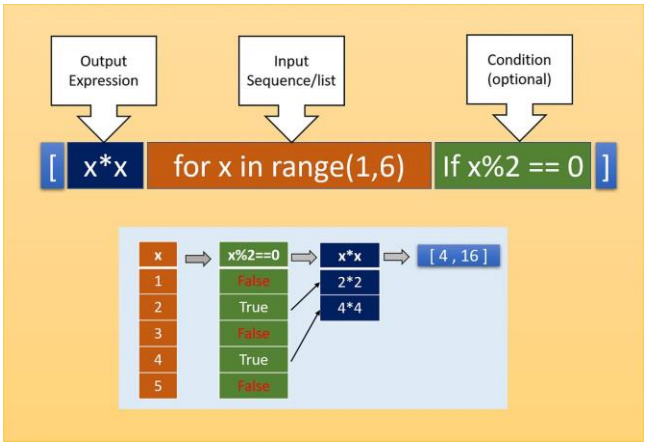
```
even = [x for x in range(1,11) if x%2==0]

print(even)
```

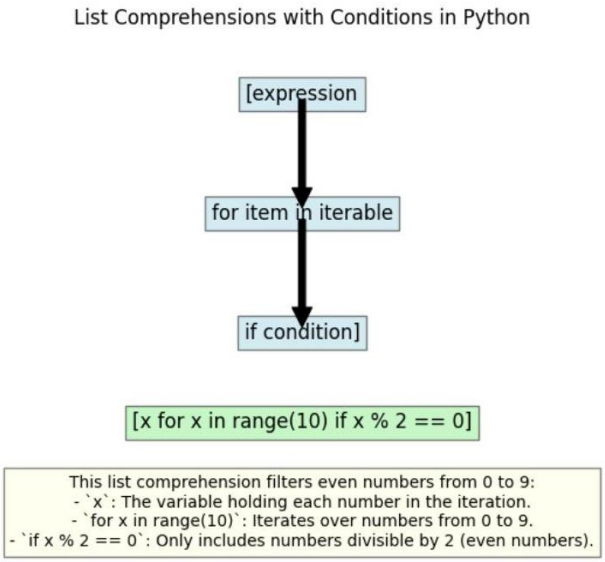
Output

```
[2,4,6,8,10]
```

📷 Picture 5 – Even Numbers







📄 Lambda Functions

What is a Lambda Function?

A Lambda Function is a **small anonymous function**.

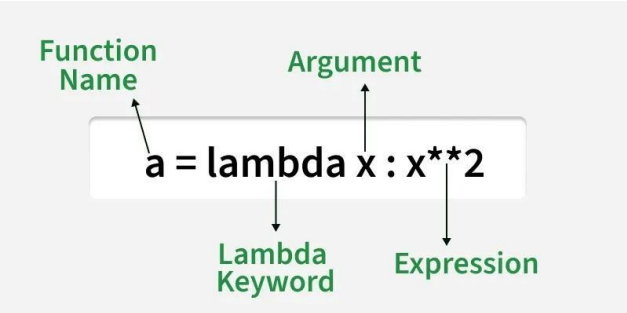
Instead of writing

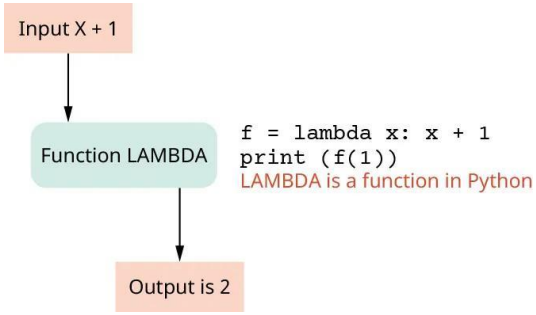
```
def add(a,b):
    return a+b
```

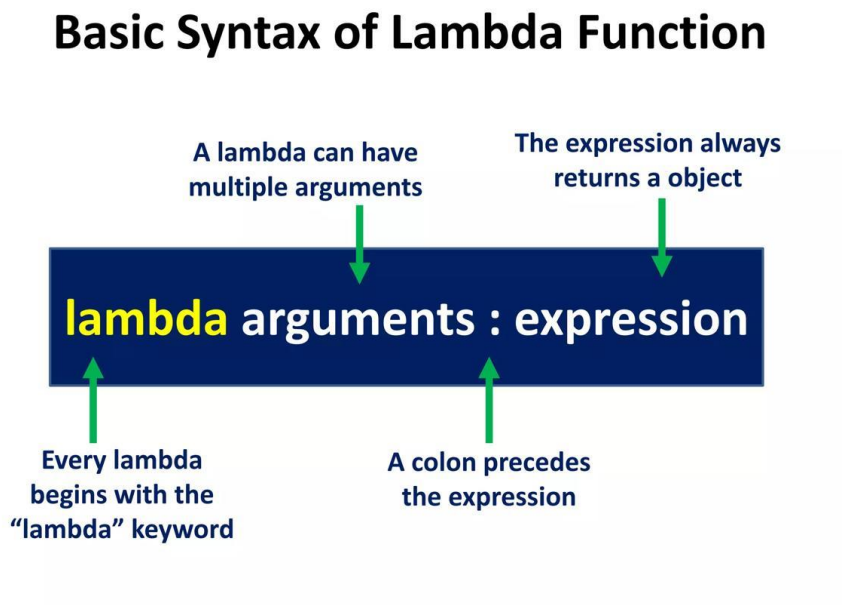
We can write

```
lambda a,b:a+b
```

📷 Picture 6 – Lambda Function







Example 1

```
add = lambda a,b:a+b

print(add(10,20))
```

Output

```
30
```

Example 2

Square Number

```
square = lambda x:x*x

print(square(5))
```

Output

```
25
```

Example 3

Largest Number

```
largest = lambda a,b:a if a>b else b

print(largest(20,15))
```

Output

```
20
```

Real-Time Program

Student Marks

```
marks = [45,60,72,81,95]

passed = [m for m in marks if m>=35]

print("Passed Students Marks")

print(passed)
```

Output

```
Passed Students Marks

[45,60,72,81,95]
```

Mini Project

Square Calculator

```
square = lambda number:number*number

number = int(input("Enter Number : "))

print("Square =", square(number))
```

Comparison

Normal Function	Lambda Function
Uses `def`	Uses `lambda`
Multiple Lines	One Line
Can contain many statements	Single expression

Assignment

Write programs to:

1. Create a list using List Comprehension.
2. Create a list of squares.
3. Create a list of even numbers.
4. Create a Lambda function to add numbers.
5. Create a Lambda function to find the cube of a number.
6. Create a Lambda function to find the largest of two numbers.

Interview Questions

1. What is List Comprehension?
2. Why do we use List Comprehension?
3. What is Lambda Function?
4. Why is Lambda called an Anonymous Function?
5. Difference between `def` and `lambda`?

🏆 Day 16 Challenge – Student Marks Analyzer

Write a Python program that:

- Accepts marks for **5 students**.
- Uses **List Comprehension** to create a list of students who scored **50 or above**.
- Uses a **Lambda Function** to calculate the square of each student's mark.
- Displays:
  - Original Marks
  - Passed Marks
  - Squares of Marks

Sample Output

```
Original Marks : [45, 67, 82, 30, 91]

Passed Marks : [67, 82, 91]

Square of 45 = 2025
Square of 67 = 4489
Square of 82 = 6724
Square of 30 = 900
Square of 91 = 8281
```

🌟 Day 16 Success Message

Congratulations! 🎉

Today you learned two powerful Python features:

- ✅ **List Comprehension** – Write cleaner and shorter code for creating lists.
- ✅ **Lambda Functions** – Create simple one-line functions quickly.

These concepts are widely used in **Data Science, Artificial Intelligence (AI), Machine Learning, Data Analysis, Web Development, and Automation**.

"A beginner writes code that works. An experienced programmer writes code that is simple, clean, and efficient. List Comprehensions and Lambda Functions are important steps toward writing professional Python code."