4. Understanding Packages – Use Case Scenarios

1. `shapes` Package (circle.py, rectangle.py)

Scenario:
A design company builds CAD (Computer-Aided Design) software. They frequently need to calculate the area and perimeter of shapes. To avoid rewriting formulas everywhere, they create a **`shapes` package**.

- `circle.py` → `area(radius)`, `perimeter(radius)`
- `rectangle.py` → `area(length, width)`, `perimeter(length, width)`

Implementation idea:
Architects can import `shapes.circle` to quickly compute land area or `shapes.rectangle` for floor plans.

2. `ecommerce` Package (cart.py, payment.py)

Scenario:
An online shopping platform needs to manage carts and payments.

- `cart.py` → Add items, remove items, calculate total.
- `payment.py` → Simulate payment (success/failure).

Implementation idea:
When a user checks out, the system imports `ecommerce.cart` to get the final bill and then calls `ecommerce.payment` to process payment.

3. `utilities` Package (string_ops.py, math_ops.py, file_ops.py)

Scenario:
A data analytics team often needs to clean text, perform math, and handle files. Instead of writing these functions every time, they create a `utilities` package.

- `string_ops.py` → Remove punctuation, count vowels, etc.
- `math_ops.py` → Mean, median, standard deviation.
- `file_ops.py` → Read, write, search in files.

Implementation idea:
Analysts import `utilities.string_ops` for cleaning survey responses,

`utilities.math_ops` for calculating averages, and `utilities.file_ops` to read/write reports.

4. `school` Package (students.py, teachers.py, results.py)

Scenario:
A school management system needs to keep track of students, teachers, and results.

- `students.py` → Add students, view student list.
- `teachers.py` → Assign subjects, view teacher info.
- `results.py` → Calculate grades from marks.

Implementation idea:
When generating a report card, the program pulls student info from `school.students`, teacher details from `school.teachers`, and calculates grades using `school.results`.

5. `banking` Package (accounts.py, transactions.py)

Scenario:
A banking app needs to manage customer accounts and perform transactions.

- `accounts.py` → Create accounts, check balance.
- `transactions.py` → Deposit, withdraw, transfer money.

Implementation idea:
When a customer deposits money, the program calls `banking.transactions.deposit()` and then updates balance using `banking.accounts.get_balance()`.

📌 5. Powerful Lambda Function in Python – Use Case Scenarios

1. Sort a list of tuples by the second element (descending)

Scenario:
An HR team has a list of `(employee_name, performance_score)`. They want to sort employees by performance for promotions.

```
employees = [("Asha", 85), ("Bala", 92), ("Chitra", 78)]
```
Use: `sorted(employees, key=lambda x: x[1], reverse=True)`

2. Use `map()` and lambda to square numbers

Scenario:
A data scientist needs to normalize sensor readings by squaring each value for variance calculations.

```
readings = [2, 3, 4]
```
Use: `map(lambda x: x**2, readings)`

3. Filter out even numbers using `filter()` and lambda

Scenario:
A game requires only odd-numbered IDs for a lottery draw. From a list of player IDs, select only odd ones.

```
player_ids = [101, 102, 103, 104, 105]
```
Use: `filter(lambda x: x % 2 != 0, player_ids)`

4. Use `reduce()` and lambda to calculate product of numbers

Scenario:
A warehouse system needs to calculate the **total volume** of stacked boxes by multiplying dimensions (length × width × height).

```
dimensions = [2, 3, 5]  # l × w × h
```
Use: `reduce(lambda x, y: x * y, dimensions)`

5. Sort a dictionary of students by marks (values)

Scenario:
A teacher has student marks stored in a dictionary. They want to print the rank list.

```
students = {"Asha": 78, "Bala": 90, "Chitra": 65}
```
Use: `sorted(students.items(), key=lambda x: x[1], reverse=True)`