Note: Solution for the exercises will be on GitHub.

Day 3: Functions & Modules

- Args/kwargs, default parameters
- Lambda, map, filter, reduce
- Decorators (basics)
- Creating and importing modules

✅ **1. Function Parameters: Default, *args, **kwargs**

🔹 **Basic + Default Parameters**

```
def greet(name="Guest"):
    print(f"Hello, {name}!")


greet("Alice")
greet()
```

🔹 ***args (Variable-length positional arguments)**

```
def add_all(*numbers):
    return sum(numbers)


print(add_all(1, 2, 3, 4))
```

🔹 ****kwargs (Variable-length keyword arguments)**

```
def print_info(**details):
    for key, value in details.items():
        print(f"{key}: {value}")


print_info(name="Alice", age=25, city="Paris")
```

✅ **2. Lambda, map, filter, reduce**

🔹 **Lambda (Anonymous Function)**

Note: Solution for the exercises will be on GitHub.

```
square = lambda x: x * x

print(square(5))
```

🔹 **map() – Apply a function to all items**

```
nums = [1, 2, 3, 4]

squared = list(map(lambda x: x**2, nums))

print(squared)
```

🔹 **filter() – Keep items where function returns True**

```
even = list(filter(lambda x: x % 2 == 0, nums))

print(even)
```

🔹 **reduce() – Repeatedly apply a function (needs functools)**

```
from functools import reduce


product = reduce(lambda x, y: x * y, nums)

print(product)
```

✅ **3. Creating and Importing a Module**

If you create a file math_utils.py:

```
def square(x):
    return x * x
```

Then in another file:

```
import math_utils

print(math_utils.square(4))
```

🧠 **Mini Exercises**

1. Write a function that takes any number of numbers and returns their average using *args.
2. Use map() to convert a list of strings to uppercase.
3. Use filter() to get all numbers > 10 from a list.
4. Use reduce() to find the sum of a list.