
📘 Lecture 11: Higher-Order Functions in Python

🧠 What Are Higher-Order Functions?

A **Higher-Order Function** is a function that:

✅ Takes **another function** as an argument
✅ Or **returns** a function as output

🍽️ Real-Life Analogy

👨‍🍳 A **head chef** doesn't cook himself
➡️ He gives the recipe to a **junior chef (function)** to prepare
✔️ Just like higher-order functions delegate work!

🔹 1. `map()` Function

Applies a function to **every item** in a list or iterable.

```
python
CopyEdit
def double(x):
    return x * 2

numbers = [1, 2, 3, 4]
result = list(map(double, numbers))
print(result)  # → [2, 4, 6, 8]
```

🔹 2. `filter()` Function

Filters elements **based on a condition**.

```
python
CopyEdit
def is_even(x):
    return x % 2 == 0
```

```
numbers = [1, 2, 3, 4, 5, 6]
result = list(filter(is_even, numbers))
print(result)  # → [2, 4, 6]
```

🔷 3. `reduce()` Function

🔁 Applies a function **cumulatively** (like folding).
🔂 Needs to be **imported** from `functools`.

```
python
CopyEdit
from functools import reduce

def multiply(x, y):
    return x * y

numbers = [1, 2, 3, 4, 5]
result = reduce(multiply, numbers)
print(result)  # → 120
```

🔷 4. `zip()` Function

Combines two or more iterables into **tuples**.

```
python
CopyEdit
names = ['Alice', 'Bob', 'Charlie']
ages = [25, 30, 35]

result = list(zip(names, ages))
print(result)  # → [('Alice', 25), ('Bob', 30), ('Charlie', 35)]
```

🔷 5. `enumerate()` Function

Gives **index + value** during iteration.

```
python
CopyEdit
fruits = ['apple', 'banana', 'cherry']

for index, fruit in enumerate(fruits, start=1):
    print(f"{index}. {fruit}")
```

🖨️ Output:

```
markdown
CopyEdit
1. apple
2. banana
3. cherry
```

✅ Best Practices

Function	When to Use
`map()`	When applying simple transformation
`filter()`	When filtering based on conditions
`reduce()`	For cumulative results (use carefully!)
`zip()`	Pairing multiple lists
`enumerate()`	Accessing index + item in loops

💡 **List Comprehensions** can replace simple `map()`/`filter()`.

🎯 Final Thoughts

- Higher-order functions = **powerful + flexible**
- Useful for **clean, modular, reusable** code
- Practice is 🔑 — try combining them with `lambda`, loops, and conditions!