Python's `map`, `filter`, and `lambda`: A Practical Mini-Lesson

Objective

After this lesson, students will be able to:

- Understand what **higher-order functions** are in Python
- Use `map()` to transform lists efficiently
- Use `filter()` to select elements based on a condition
- Write concise anonymous functions using `lambda`
- Combine all three to write more expressive code

---

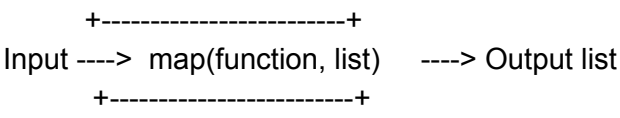
1. What Are Higher-Order Functions?

In Python, a *higher-order function* is any function that:

1. Takes another function as input
2. Returns a function as output

`map()` and `filter()` both take a function + an iterable (like a list), and apply that function to items in the iterable.

Diagram:



---

2. `lambda`: Creating Small, Anonymous Functions

A `lambda` function is a quick way to create a tiny function without using `def`.

Syntax

lambda inputs: output_expression

Example

```
square = lambda x: x * x
print(square(5))   # 25
```

Lambdas are perfect when you need a tiny function only once — like inside `map()` or `filter()`.

---

3. `map()`: Transforming Data

`map()` applies a function to **every element** in an iterable.

Syntax

map(function, iterable)

Example: Convert Celsius to Fahrenheit

```
temps_c = [0, 10, 20, 30]
temps_f = list(map(lambda c: 9/5 * c + 32, temps_c))

print(temps_f)  # [32.0, 50.0, 68.0, 86.0]
```

Why use `map()`?

- Cleaner than loops
- Often faster
- Works well with lambda for one-line transformations

---

4. `filter()`: Selecting Elements

`filter()` keeps only elements for which the function returns **True**.

Syntax

filter(function, iterable)

Example: Filter Even Numbers

```
nums = [1, 2, 3, 4, 5, 6]
evens = list(filter(lambda x: x % 2 == 0, nums))

print(evens)  # [2, 4, 6]
```

---

5. Combining `map()` + `filter()`

You can chain them logically:

Example: From a list, select the even numbers, then square them.

```
nums = [1, 2, 3, 4, 5, 6]

evens = filter(lambda x: x % 2 == 0, nums)
squared = map(lambda x: x * x, evens)

print(list(squared))  # [4, 16, 36]
```

---

6. Real-World Mini Case Study

Imagine you're cleaning a dataset of student quiz scores.

Raw data:

```
scores = ["90", "85", "70", "", "88", "95", "abc", "77"]
```

Goal:

1. Remove invalid entries
2. Convert to integers
3. Curve scores by +5 points

Solution:

```
cleaned = filter(lambda s: s.isdigit(), scores)
ints = map(lambda s: int(s), cleaned)
curved = map(lambda x: x + 5, ints)

print(list(curved))
# [95, 90, 75, 93, 100, 82]
```

This example shows how powerful these tools become when chained.

---

7. Summary Table

Concept	Purpose	Example
`lambda`	small anonymous function	`lambda x: x+1`
`map()`	transform all elements	`map(lambda x: x*2, nums)`
`filter()`	select elements by condition	`filter(lambda x: x>0, nums)`

---

8. Optional Exercises

These can be assigned or included in the video.

Exercise 1

Given:

words = ["apple", "hi", "book", "to", "computer"]

Use `filter()` to keep only words with length ≥ 4.

Exercise 2

Given:

nums = [1, 3, 5, 7]

Use `map()` to compute the cube of each number.

Exercise 3

Chain `filter()` and `map()`:

- keep only multiples of 3
- then double them

---

9. Additional References

- Real Python tutorial: https://realpython.com/python-map-function/