Advanced Functions in Python

Lambda Functions

Definition

A **lambda function** is a **small anonymous function** (no name) written in **one line**.
It is used for short, simple operations.

Syntax

```
lambda arguments: expression
```

Simple Example

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

print(add(5, 3))
```

Output

```
8
```

Practical Example

```
# square of a number

square = lambda x: x * x

print(square(4))
```

---

map() Function

Definition

map() applies a function to **each element of an iterable** (list, tuple, etc.).

Syntax

```
map(function, iterable)
```

Simple Example

```
numbers = [1, 2, 3, 4]
result = map(lambda x: x * 2, numbers)
print(list(result))
```

Output

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

Practical Example

```
# Convert temperatures from Celsius to Fahrenheit
celsius = [0, 10, 20, 30]

fahrenheit = list(map(lambda c: (c * 9/5) + 32, celsius))
print(fahrenheit)
```

---

filter() Function

Definition

filter() selects elements from an iterable **based on a condition**.

Syntax

```
filter(function, iterable)
```

Simple Example

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

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

Output

```
[2, 4, 6]
```

Practical Example

```
# Filter students who passed
marks = [45, 78, 32, 90, 60]

passed = list(filter(lambda m: m >= 50, marks))
print(passed)
```

---

reduce() Function

Definition

reduce() **reduces** a list to a **single value** by applying a function repeatedly.

reduce() is available in functools module.

Syntax

```
from functools import reduce
reduce(function, iterable)
```

Simple Example

```
from functools import reduce

numbers = [1, 2, 3, 4]
result = reduce(lambda a, b: a + b, numbers)
print(result)
```

Output

```
10
```

Practical Example

```
# Find product of all numbers
from functools import reduce

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

```
product = reduce(lambda a, b: a * b, nums)
print(product)
```

Activity: Data Transformation Using Advanced Functions

Problem

Given a list of salaries:

1. Increase each salary by 10%
2. Filter salaries above 50,000
3. Find total salary expense

Complete Practical Code

```
from functools import reduce

salaries = [30000, 45000, 60000, 80000]

# Step 1: Increase salary by 10%
updated_salaries = list(map(lambda s: s + s * 0.10, salaries))

# Step 2: Filter salaries above 50,000
high_salaries = list(filter(lambda s: s > 50000, updated_salaries))

# Step 3: Total expense
total = reduce(lambda a, b: a + b, high_salaries)

print("Updated Salaries:", updated_salaries)
print("High Salaries:", high_salaries)
print("Total Expense:", total)
```

Output

```
Updated Salaries: [33000.0, 49500.0, 66000.0, 88000.0]

High Salaries: [66000.0, 88000.0]

Total Expense: 154000.0
```

Quick Comparison Table

Function	Purpose
lambda	Short anonymous function
map()	Transform data
filter()	Select data
reduce()	Aggregate data

PRACTICE PROBLEMS: Advanced Functions

Lambda Functions

Square of a Number

Problem: Find square using lambda.

```
square = lambda x: x * x

print(square(6))
```

Add Two Numbers

Problem: Add two numbers using lambda.

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

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

Check Even or Odd

Problem: Check even number.

```
is_even = lambda x: x % 2 == 0

print(is_even(8))
```

Maximum of Two Numbers

Problem: Find maximum.

```
maximum = lambda a, b: a if a > b else b
print(maximum(5, 9))
```

String Length

Problem: Find length of string.

```
length = lambda s: len(s)
print(length("Python"))
```

---

map() Function (6–10)

Double Each Element

```
nums = [1, 2, 3, 4]
result = list(map(lambda x: x * 2, nums))
print(result)
```

Convert to Uppercase

```
names = ["athar", "python", "code"]
result = list(map(lambda x: x.upper(), names))
print(result)
```

Square List Elements

```
nums = [2, 4, 6]
squares = list(map(lambda x: x ** 2, nums))
print(squares)
```

Convert Celsius to Fahrenheit

```python
celsius = [0, 10, 20]

fahrenheit = list(map(lambda c: (c * 9/5) + 32, celsius))

print(fahrenheit)
```

Add 5 to Each Element

```python
nums = [10, 20, 30]

result = list(map(lambda x: x + 5, nums))

print(result)
```

---

filter() Function (11–15)

Filter Even Numbers

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

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

print(evens)
```

Filter Odd Numbers

```python
odds = list(filter(lambda x: x % 2 != 0, nums))

print(odds)
```

Numbers Greater Than 50

```python
marks = [45, 78, 32, 90, 60]

passed = list(filter(lambda x: x > 50, marks))

print(passed)
```

Filter Positive Numbers

```python
nums = [-5, 10, -2, 8]

positive = list(filter(lambda x: x > 0, nums))

print(positive)
```

Filter Names Starting with 'A'

```
names = ["Athar", "Ali", "John", "Aman"]
result = list(filter(lambda x: x.startswith("A"), names))
print(result)
```

---

reduce() Function (16–20)

Sum of All Numbers

```
from functools import reduce

nums = [1, 2, 3, 4]
total = reduce(lambda a, b: a + b, nums)
print(total)
```

Product of Numbers

```
product = reduce(lambda a, b: a * b, nums)
print(product)
```

Find Maximum Value

```
maximum = reduce(lambda a, b: a if a > b else b, nums)
print(maximum)
```

Find Minimum Value

```
minimum = reduce(lambda a, b: a if a < b else b, nums)
print(minimum)
```

Total Salary After Bonus

Problem: Add 10% bonus and calculate total.

```
from functools import reduce

salaries = [30000, 45000, 60000]

updated = list(map(lambda s: s + s * 0.10, salaries))
total = reduce(lambda a, b: a + b, updated)

print(updated)
print(total)
```

QUICK REVISION

Function	Use
lambda	Short functions
map()	Transform
filter()	Select
reduce()	Aggregate