PES Vision: To create professionally superior and ethically strong global workforce

PES UNIVERSITY, BENGALURU

Perseverance | Excellence | Service

UE25CS151A – PYTHON FOR COMPUTATIONAL PROBLEM SOLVING LAB MANUAL

WEEK 12

TOPICS:

Programs on Callbacks, Closures and Decorators

OBEJCTIVE:

To understand and implement the concepts of **callbacks, closures, and decorators** in Python.
Students will learn how functions can be passed as arguments, how inner functions can retain states, and how decorators can modify the behaviour of existing functions dynamically.

Problem Statement 1: (callback)

Write a function **process_data(data_list, callback)** that takes a list of numbers and a callback function. This function should iterate through each element in the list, apply the callback function to the element, and print the result.

Create two separate callback functions:

1. square_num(n): Returns the square of n.
2. double_num(n): Returns n multiplied by 2.

Demonstrate process_data using both callback functions on the list [1, 5, 9, 12].

Expected Output:

```
Function name:square_num
Original: 1, Processed: 1
Original: 5, Processed: 25
Original: 9, Processed: 81
Original: 12, Processed: 144
------------------------------
Function name:double_num
Original: 1, Processed: 2
Original: 5, Processed: 10
Original: 9, Processed: 18
Original: 12, Processed: 24
```

Problem Statement 2: (callback)

Write a function custom_sort(items, key_callback) that sorts a given list items based on the value returned by the key_callback.
You are given a list of tuples, where each tuple contains (item_name, price, quantity).
store_items = [('apple', 0.5, 10), ('banana', 0.2, 20), ('cherry', 2.0, 5), ('date', 1.5, 8)]
Demonstrate your custom_sort function by sorting this list based on two different criteria:

1. By **price** (lowest to highest).
2. By **quantity** (highest to lowest).

Expected Output:

```
Original list:
[('apple', 2000, 10), ('banana', 5000, 20), ('cherry', 1500, 5), ('date', 3500, 8)]

Sorted by price (low to high):
[('cherry', 1500, 5), ('apple', 2000, 10), ('date', 3500, 8), ('banana', 5000, 20)]

Sorted by quantity (high to low):
[('banana', 5000, 20), ('apple', 2000, 10), ('date', 3500, 8), ('cherry', 1500, 5)]
```

Problem Statement 3: (Closures)

Write a function make_power_of(n) that takes an exponent n. This function should return another function (a closure). The returned function should take a single argument x and return x raised to the power of n.

Demonstrate this by creating a squarer (power of 2) and a cuber (power of 3) and testing them with the number 4.

Expected Output:

```
4 squared is: 16
4 cubed is: 64
10 squared is: 100
```

Problem Statement 4: (Closures)

Write a function create_counter(start_value) that takes an integer start_value. It should return a new function (a closure) that, when called, returns the start_value on its first call, start_value + 1 on its second call, start_value + 2 on its third call, and so on.

Each counter created must be independent. Demonstrate this by creating two counters, counter_A starting at 10 and counter_B starting at 0, and calling them multiple times.

Expected Output:

```
--- Counter A ---
10
11
12
--- Counter B ---
0
1
--- Counter A (Again) ---
13
```

Problem Statement 5: (Decorators)

Write a decorator that measures and prints the execution time of any function

Note: Use time module

Expected Output:

```
start time:1762960116.5246
Hello, welcome to Python Decorators!
End time:1762960117.5258
Execution Time: 1.0012 seconds
```

Problem Statement 6: (Decorators)

Write a Python program using a **decorator** that converts the **output string of a function to uppercase**. The function should accept a person's name as input and return a greeting message in the format:

"Hello <name>, good morning!"

Use the decorator to automatically convert this greeting message to uppercase before displaying it.

Expected Output:

```
HELLO PES UNIVERSITY, GOOD MORNING!
```

Callbacks connect, closures preserve, and decorators transform — the trio that defines functional elegance in Python