**Python Assessment - 3**

**1. Multithreading**

**Problem Statement:**

Create a function `print\_numbers` that prints numbers from 1 to 5 with a 1-second delay between each number. Use the `threading` module to create two threads that run `print\_numbers` simultaneously.

**Example:**

# Expected output (order may vary depending on thread completion):

1

1

2

2

3

3

4

4

5

5

**2. Iterators**

**Problem Statement:**

Create a custom iterator class called `EvenNumbers` that generates even numbers starting from a given number up to a specified limit. Implement the `\_\_iter\_\_()` and `\_\_next\_\_()` methods.

**Example:**

evens = EvenNumbers(2, 10)

for number in evens:

print(number)

# Output: 2, 4, 6, 8, 10

**3. Generators**

**Problem Statement:**

Write a generator function called `fibonacci\_sequence` that generates the Fibonacci sequence up to a specified number of terms.

**Example:**

fib = fibonacci\_sequence(5)

print(list(fib))

# Output: [0, 1, 1, 2, 3]

**4. Decorators**

**Problem Statement:**

Create a decorator function `time\_logger` that logs the time taken by a function to execute. Use it to decorate a sample function `calculate\_sum` which takes a list of numbers and returns their sum.

**Example:**

@time\_logger

def calculate\_sum(numbers):

return sum(numbers)

calculate\_sum([1, 2, 3, 4, 5])

# Expected output:

# Execution time: <time\_taken> seconds

# 15

**5. Regex**

**Problem Statement:**

Write a function `validate\_email` that takes a string as input and uses regex to validate if it is a correctly formatted email address. Return `True` if it is valid, and `False` otherwise.

**Example:**

print(validate\_email("example@test.com")) # Output: True

print(validate\_email("example@test")) # Output: False

**6. JSON Processing**

Problem Statement:

Write a function `process\_json` that reads a JSON string of user information containing fields `name`, `age`, and `email`. Extract and print each field in a formatted output.

**Example:**

json\_data = '{"name": "John Doe", "age": 30, "email": "john@example.com"}'

process\_json(json\_data)

# Expected output:

# Name: John Doe

# Age: 30

# Email: john@example.com

**7. XML Processing**

**Problem Statement:**

Write a function `parse\_xml` that takes an XML string containing a list of books, each with `title`, `author`, and `year` elements. Extract and print each book's details.

**Example:**

xml\_data = '''

<library>

<book>

<title>Book One</title>

<author>Author One</author>

<year>2001</year>

</book>

<book>

<title>Book Two</title>

<author>Author Two</author>

<year>2005</year>

</book>

</library>

'''

parse\_xml(xml\_data)

# Expected output:

# Title: Book One, Author: Author One, Year: 2001

# Title: Book Two, Author: Author Two, Year: 2005

**8. Mini Project**

**Problem Statement:**

Create a mini project that simulates a basic to-do list application using JSON for data storage. Implement the following functionalities:

- Add a new task with a title and description.

- List all tasks.

- Mark a task as completed.

Create a `TodoList` class to handle these tasks, and store data in a JSON file.

**Example:**

todo = TodoList("tasks.json")

todo.add\_task("Buy groceries", "Milk, Eggs, Bread")

todo.add\_task("Study Python", "Complete multithreading chapter")

todo.list\_tasks()

# Expected output:

# 1. Buy groceries - Milk, Eggs, Bread (Incomplete)

# 2. Study Python - Complete multithreading chapter (Incomplete)

todo.mark\_task\_completed(1)

todo.list\_tasks()

# Expected output:

# 1. Buy groceries - Milk, Eggs, Bread (Completed)

# 2. Study Python - Complete multithreading chapter (Incomplete)