**Lab Exercise 2 – Python Generator**

Objective: In this lab exercise, you will learn about Python generators, how they differ from regular functions, and how to create and use them.

**Instructions:**

* Part 1: Understanding Generators
* Open a Python IDE or text editor.
* Create a Python function that generates a list of square numbers from 1 to 5. Print the list.

def generate\_squares():

squares = []

for i in range(1, 6):

squares.append(i \*\* 2)

return squares

square\_list = generate\_squares()

print(square\_list)

Now, create a generator function that generates the same list of square numbers. Use the yield keyword to yield each square one at a time.

def generate\_squares\_generator():

for i in range(1, 6):

yield i \*\* 2

square\_gen = generate\_squares\_generator()

print(square\_gen)

**Part 2: Lazy Evaluation**

Implement a generator function that generates an infinite sequence of random numbers between 1 and 100. Use the random module for generating random numbers.

import random

def random\_number\_generator():

while True:

yield random.randint(1, 100)

rand\_gen = random\_number\_generator()

for \_ in range(10):

print(next(rand\_gen))

**Part 3: Applying Generators**

Create a text file named "sample.txt" (you can create this file with some sample text).

Implement a generator function that yields lines from the text file one by one.

def read\_text\_file(filename):

with open(filename, 'r') as file:

for line in file:

yield line.strip()

text\_gen = read\_text\_file('sample.txt')

for \_ in range(10):

print(next(text\_gen))

Use the generator to iterate through the text file and print the first 10 lines.

Modify the generator to skip lines that start with a specific character, such as "#". Print the first 10 non-comment lines.

def read\_text\_file\_no\_comments(filename, comment\_char='#'):

with open(filename, 'r') as file:

for line in file:

if not line.startswith(comment\_char):

yield line.strip()

text\_gen\_no\_comments = read\_text\_file\_no\_comments('sample.txt')

for \_ in range(10):

print(next(text\_gen\_no\_comments))

**Conclusion:** In this lab exercise, you learned about Python generators, lazy evaluation, and how to create and use them for efficient data processing and manipulation. Generators are a powerful tool for working with large or infinite sequences of data in a memory-efficient manner.