**🔹 Section A: Iterators (Q1–Q4)**

**1. Basic Iterator Creation**  
Write a Python program to create an iterator for a list of fruits (["apple", "banana", "cherry"]) and print each element using next().

**2. Custom Iterator Class**  
Create a custom iterator class CountDown that takes a number n and returns numbers from n to 1.  
Example:  
Input → CountDown(5)  
Output → 5 4 3 2 1

**3. Reverse Iterator**  
Write a custom iterator ReverseIterator that iterates through a list in reverse order without using reversed().

**4. Iterator from String**  
Write a program to iterate through each character in a string "PYTHON" using an iterator and print them one by one.

**🔹 Section B: Generators (Q5–Q9)**

**5. Simple Generator**  
Create a generator function number\_generator() that yields numbers from 1 to 5 and prints them using a loop.

**6. Fibonacci Generator**  
Write a generator that yields the first n numbers in the Fibonacci sequence.  
Example: For n = 6 → Output: 0 1 1 2 3 5

**7. Even Number Generator**  
Write a generator even\_numbers(limit) that yields even numbers up to the given limit.  
Example: limit = 10 → Output: 2 4 6 8 10

**8. Infinite Generator (Use with Break)**  
Create an infinite generator count\_up() that starts from 1 and keeps incrementing. Use a loop to print numbers only up to 10.

**9. Square Generator with List Comprehension**  
Write a generator expression that produces squares of numbers from 1 to 5.  
Example: Output → [1, 4, 9, 16, 25]

**🔹 Section C: any() and all() (Q10–Q11)**

**10. Use of any()**  
Given a list nums = [3, 7, 1, 9, 2], use a generator expression with any() to check if any number is greater than 8.

**11. Use of all()**  
Given nums = [2, 4, 6, 8], use a generator expression with all() to check if all numbers are even.

**🔹 Section D: Generators with Files & contextlib (Q12–Q13)**

**12. File Reader Generator**  
Create a generator function read\_file(filename) that reads lines from a text file one by one using yield.

**13. Context Manager with contextlib**  
Use @contextmanager to create a custom file handler that automatically opens and closes a file. Write data "Python is powerful!" into a text file using this context manager.

**🔹 Section E: Data Compression & Filtering (Q14–Q15)**

**14. Using itertools.compress()**  
Given

from itertools import compress

data = ['A', 'B', 'C', 'D', 'E']

selectors = [1, 0, 1, 0, 1]

Use compress() to print only the selected elements.

**15. Conditional Generator Compression**  
Given

numbers = [5, 12, 25, 7, 30, 18]

Use a generator expression to yield numbers greater than 10 and print the result as a list.