Q1. Function to Return Odd Numbers in a Range

To define a function in Python, the keyword def is used.

# Function to return odd numbers from 1 to 25

def get\_odd\_numbers():

return [num for num in range(1, 26) if num % 2 != 0]

print(get\_odd\_numbers())

---

Q2. Using \*args and \*\*kwargs in Functions

\*args is used to pass a variable number of arguments to a function. It collects extra arguments as a tuple.

\*\*kwargs is used to pass a variable number of keyword arguments. It collects extra arguments as a dictionary.

Example with \*args:

def sum\_all(\*args):

return sum(args)

print(sum\_all(1, 2, 3, 4, 5)) # Output: 15

Example with \*\*kwargs:

def print\_details(\*\*kwargs):

for key, value in kwargs.items():

print(f"{key}: {value}")

print\_details(name="Alice", age=25, location="New York")

---

Q3. Iterator in Python

An iterator in Python is an object that can be iterated (looped) upon.

The method used to initialize an iterator object is iter().

The method used to get the next item from an iterator is next().

Example:

# Given list

numbers = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

# Creating an iterator

iterator = iter(numbers)

# Using next() to get the first five elements

for \_ in range(5):

print(next(iterator))

---

Q4. Generator Function in Python and the yield Keyword

A generator function is a function that yields a sequence of values instead of returning them all at once.

The yield keyword is used to produce a value in a generator, allowing the function to return an intermediate result and resume later.

Example:

def simple\_generator():

yield 1

yield 2

yield 3

for value in simple\_generator():

print(value)

---

Q5. Generator Function for Prime Numbers Less Than 1000

# Prime number generator function

def prime\_generator():

for num in range(2, 1000):

is\_prime = True

for i in range(2, int(num \*\* 0.5) + 1):

if num % i == 0:

is\_prime = False

break

if is\_prime:

yield num

# Using next() to get the first 20 prime numbers

prime\_gen = prime\_generator()

for \_ in range(20):

print(next(prime\_gen))

---

Q6. Program to Print the First 10 Fibonacci Numbers Using a While Loop

# Printing the first 10 Fibonacci numbers

a, b = 0, 1

count = 0

while count < 10:

print(a)

a, b = b, a + b

count += 1

---

Q7. List Comprehension to Extract Characters in 'pwskills'

# Given string

string = 'pwskills'

# List comprehension to get the expected output

output = [char for char in string if char in 'pwskills']

print(output)

# Output: ['p', 'w', 's', 'k', 'i', 'l', 'l', 's']

---

Q8. Check if a Number is Palindrome Using a While Loop

# Function to check if a number is palindrome

def is\_palindrome(number):

original = number

reverse = 0

while number > 0:

digit = number % 10

reverse = reverse \* 10 + digit

number //= 10

return original == reverse

# Checking a number

num = int(input("Enter a number: "))

if is\_palindrome(num):

print(f"{num} is a palindrome")

else:

print(f"{num} is not a palindrome")

---

Q9. Print Odd Numbers from 1 to 100 Using List Comprehension

# Creating a list of numbers from 1 to 100

numbers = [i for i in range(1, 101)]

# Filtering out odd numbers using list comprehension

odd\_numbers = [num for num in numbers if num % 2 != 0]

print(odd\_numbers)