Python Coding Questions :

1.Write a Python program to check if a number is even or odd.

def check\_even\_odd(num):

if num % 2 == 0:

print(f"{num} is Even")

else:

print(f"{num} is Odd")

# Example

check\_even\_odd(7)

check\_even\_odd(10)

2.Write a Python function to reverse a string without using built-in functions.

def reverse\_string(s):

reversed\_s = ""

for char in s:

reversed\_s = char + reversed\_s

return reversed\_s

# Example

print(reverse\_string("hello")) # Output: "olleh"

3.Write a Python program to find the largest number in a given list.

def find\_largest(lst):

largest = lst[0]

for num in lst:

if num > largest:

largest = num

return largest

# Example

print(find\_largest([3, 7, 2, 9, 5])) # Output: 9

4.Write a Python function to check if a given string is a palindrome.

def is\_palindrome(s):

s = s.lower() # Making it case-insensitive

return s == s[::-1]

# Example

print(is\_palindrome("Madam")) # Output: True

print(is\_palindrome("Hello")) # Output: False

5.Write a Python program to count the occurrences of each character in a given string.

def count\_characters(s):

char\_count = {}

for char in s:

char\_count[char] = char\_count.get(char, 0) + 1

return char\_count

# Example

print(count\_characters("banana")) # Output: {'b': 1, 'a': 3, 'n': 2}

**Difference Between Class and Object** A class is a blueprint for creating objects, defining attributes and behaviors. An object is an instance of a class with specific values assigned to its attributes. Example:  
  
 class Car: pass

my\_car = Car() # Object of Car class

**Inheritance Example** Inheritance allows a class to inherit properties and methods from another class. Example:  
  
 class Animal:

def speak(self):

print("Animal speaks")

class Dog(Animal):

def speak(self):

print("Dog barks")

d = Dog()

d.speak() # Output: Dog barks

**Method Overriding Example** Method overriding occurs when a subclass provides a new implementation for a method inherited from a parent class.  
  
 class Parent:

def show(self):

print("Parent method")

class Child(Parent):

def show(self):

print("Child method")

c = Child()

c.show() # Output: Child method

1. **Instance, Class, and Static Methods**
   * **Instance method**: Operates on instance data.
   * **Class method**: Works on class-level data (@classmethod).
   * **Static method**: Independent function inside a class (@staticmethod).

class Demo:

def instance\_method(self): pass

@classmethod

def class\_method(cls): pass

@staticmethod

def static\_method(): pass

**Encapsulation Example** Encapsulation restricts access to certain data using private variables (\_\_var). Example:  
  
 class BankAccount:

def \_\_init\_\_(self, balance):

self.\_\_balance = balance

def get\_balance(self):

return self.\_\_balance

account = BankAccount(1000)

print(account.get\_balance()) # Output: 1000 .