Q1.  Explain with an example each when to use a for loop and a while loop.

Certainly! Both for loops and while loops are control structures used in programming to repeat a block of code multiple times. Here's an example of when you would typically use each type of loop: 1. For Loop: A for loop is useful when you know the number of times you want to repeat a specific code block in advance. It is commonly used when you want to iterate over a collection of elements, such as an array, a list, or a range of numbers.

Example: Let's say you have an array of numbers, and you want to calculate the sum of all the elements in the array. In this case, you can use a for loop to iterate over each element and add them up.

numbers = [1, 2, 3, 4, 5]

sum = 0

for num in numbers:

sum += num

print("The sum is:", sum)

In this example, the for loop iterates over each element **‘num’**in the ‘n**umbers’**array and adds it to the **’sum’** variable. The loop runs for each element in the array, and once it finishes, the total sum is printed.

2.While Loop: A while loop is useful when you want to repeat a block of code until a certain condition is no longer true. It is commonly used when the number of iterations is not known in advance or when you want to repeatedly perform an action until a specific condition is met.

Example: Let's say you want to simulate a game where a player has to guess a randomly generated number. You want to keep asking the player for guesses until they guess the correct number.

import random

target\_number = random.randint(1, 100)

guessed = False

while not guessed:

guess = int(input("Enter your guess: "))

if guess == target\_number:

print("Congratulations! You guessed the correct number.")

guessed = True

elif guess < target\_number:

print("Too low! Try again.")

else:

print("Too high! Try again.")

In this example, the while loop continues to prompt the player for guesses until they guess the correct number (‘**guess == target number’**). The loop repeats as long as the condition ‘**not guessed’** is true. Once the correct guess is made, the loop terminates, and a congratulatory message is displayed.

Remember, when using loops, it's important to ensure that the condition(s) in the loop are appropriately defined to avoid infinite loops or prematurely exiting the loop.

Q2.  Write a python program to print the sum and product of the first 10 natural numbers using for and while loop.

Using a ‘for’ loop

# Calculate sum using a for loop

sum = 0

product = 1

for num in range(1, 11):

sum += num

product \*= num

print("Sum using for loop:", sum)

print("Product using for loop:", product)

Using a ‘while’ loop

# Calculate sum using a while loop

sum = 0

product = 1

num = 1

while num <= 10:

sum += num

product \*= num

num += 1

print("Sum using while loop:", sum)

print("Product using while loop:", product)

Both programs will produce the same output ,while will be:

Sum using for loop: 55

Product using for loop: 3628800

Sum using while loop: 55

Product using while loop: 3628800

Q3. Create a python program to compute the electricity bill for a household. The per-unit charges in rupees are as follows: For the first 100 units, the user will be charged Rs. 4.5 per unit, for the next 100 units, the user will be charged Rs. 6 per unit, and for the next 100 units, the user will be charged Rs. 10 per unit, After 300 units and above the user will be charged Rs. 20 per unit. You are required to take the units of electricity consumed in a month from the user as input. Your program must pass this test case: when the unit of electricity consumed by the user in a month is 310, the total electricity bill should be 2250.

def calculate\_electricity\_bill(units):

total\_bill = 0

if units <= 100:

total\_bill = units \* 4.5

elif units <= 200:

total\_bill = 100 \* 4.5 + (units - 100) \* 6

elif units <= 300:

total\_bill = 100 \* 4.5 + 100 \* 6 + (units - 200) \* 10

else:

total\_bill = 100 \* 4.5 + 100 \* 6 + 100 \* 10 + (units - 300) \* 20

return total\_bill

# Get the units of electricity consumed from the user

units\_consumed = int(input("Enter the units of electricity consumed: "))

# Calculate the electricity bill

bill\_amount = calculate\_electricity\_bill(units\_consumed)

print("The total electricity bill is Rs.", bill\_amount)

When you run this program and enter the units consumed as 310, it will output:

Enter the units of electricity consumed: 310

The total electricity bill is Rs. 2250.0

The program takes the units of electricity consumed as input from the user and calculates the total bill based on the given per-unit charges. It uses if-elif-else statements to determine the appropriate charge based on the consumption range. The final bill amount is then displayed as output.

Q4 . Create a list of numbers from 1 to 100. Use for loop and while loop to calculate the cube of each number and if the cube of that number is divisible by 4 or 5 then append that number in a list and print that list .

Using a for loop:

numbers = [ ]

for i in range(1, 101):

cube = i \*\* 3

if cube % 4 == 0 or cube % 5 == 0:

numbers.append(i)

print(numbers)

Using a while loop:

numbers = [ ]

i = 1

while i <= 100:

cube = i \*\* 3

if cube % 4 == 0 or cube % 5 == 0:

numbers.append(i)

i += 1

print(numbers)

Both implementations will produce the same output, which is a list of numbers from 1 to 100 whose cubes are divisible by either 4 or 5.

Q5.  Write a program to filter count vowels in the below-given string. string = "I want to become a data scientist"

def count\_vowels(string):

vowels = ['a', 'e', 'i', 'o', 'u']

count = 0

for char in string.lower():

if char in vowels:

count += 1

return count

string = "I want to become a data scientist"

vowel\_count = count\_vowels(string)

print("Number of vowels:", vowel\_count)

This program defines a function called ‘**count\_vowels’** that takes a string as input. It initializes a variable ‘**count’** to 0 and iterates through each character of the string after converting it to lowercase. If the character is present in the list of vowels, the count is incremented.

Finally, the program calls the ‘**count\_vowels’** function with the given string and prints the resulting vowel count. In this case, the output would be:

Number of vowels: 10

this program treats both lowercase and uppercase letters as vowels and counts them accordingly. If you want to count only lowercase vowels, you can modify the program by removing the ‘string.lower()’ conversion.