forloop

May 31, 2024

[1]:
```python
#Basic Level
#1.Write a Python program to print the numbers from 1 to 10 using a `for` loop
for i in range (1,11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

[4]:
```python
#2. Create a program that calculates the sum of all numbers in a list using a␣
↪`for` loop.

number=[5,10,15,20,25,30]
total=0
for num in number:
    total+=num
print("the sum of the number is:", total)
```

```
the sum of the number is: 105
```

[8]:
```python
#3.Write a program to print the characters of a string in reverse order using a␣
↪`for` loop.
def reverse_string(input_string):
    for char in reversed(input_string):
        print(char,end ='')
input_str="Hello World!"
reverse_string(input_str)
```

```
!dlroW olleH
```

[22]:
```
#4.Develop a program that finds the factorial of a given number using a `for`␣
↪loop
def factorial(n):
    result=1
    for i in range(1,n+1):
        result*=i
    return result
number= int(input("Enter a number:"))
print("factorial of", number, "is", factorial(number))
```

```
Enter a number: 5

factorial of 5 is 120
```

[28]:
```
#5.Create a program to print the multiplication table of a given number using a␣
↪`for` loop.
def multiplication_table(number):
    print("Multiplication table of", number)
    print("----------------------")
    for i in range(1,11):
        print(number ,"x", i , "=", number*i)
num=int(input("enter a number:"))
multiplication_table(num)
```

```
enter a number: 7

Multiplication table of 7
----------------------
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

[30]:
```
#6.Write a program that counts the number of even and odd numbers in a list␣
↪using a `for` loop
def count_even_odd(numbers):
    even_count=0
    odd_count=0
    for num in numbers:
        if num % 2==0:
            even_count+=1
        else:
            odd_count+=1
    return even_count,odd_count
num_list=[1,2,3,4,5,6,7,8,9,]
even,odd=count_even_odd(num_list)
print("Number of evern number : ", even)
print("number of odd number : ", odd)
```

```
Number of evern number :  4
number of odd number :  5
```

[32]:
```
#7.Develop a program that prints the squares of numbers from 1 to 5 using a␣
 ↪`for` loop.
def print_squares():
    for i in range(1,6):
        print("square of",i,"is",i*i)
print_squares()
```

```
square of 1 is 1
square of 2 is 4
square of 3 is 9
square of 4 is 16
square of 5 is 25
```

[43]:
```
#8.Create a program to find the length of a string without using the `len()`␣
 ↪function.
def string_length(string):
    lenght=0
    for _ in string:
        lenght+=1
    return lenght
input_str="Hello, World"
print("Lenght of the string :", string_length(input_str))
```

```
Lenght of the string : 12
```

[7]:
```
#9.Write a program that calculates the average of a list of numbers using a␣
 ↪`for` loop.
def calculate_average(numbers):
    total=0
    for num in numbers:
        total+=num
        average=total/len(numbers)
    return average
num_list=[10,20,30,40,50,60]
avg = calculate_average(num_list)
```

```
print("Average of the number :", avg)
```

```
Average of the number : 35.0
```

[16]:
```
#10  Develop a program that prints the first `n` Fibonacci numbers using a␣
 ↪`for` loop.
def fibonacci(n):
    fib_sequence=[]
    a, b= 0,1
    for _ in range(n):
        fib_sequence.append(a)
        a,b=b, a+b
    return fib_sequence
num_terms= int(input("Enter the number of fibonnacci numbers to print: "))
fibonacci_sequence = fibonacci(num_terms)
print("Fibonacci sequence: ")
for term in fibonacci_sequence:
    print(term, end= " ")
```

```
Enter the number of fibonnacci numbers to print:  5
```

```
Fibonacci sequence:
0 1 1 2 3
```

[1]:
```
#Intermediate Level
#11.Write a program to check if a given list contains any duplicates using a␣
 ↪`for` loop.
def has_duplicates(lst):
    seen=set()
    for item in lst:
        if item in seen:
            return True
        seen.add(item)
    return False
my_list=[1,2,3,4,5,6]
print("List has duplicates:", has_duplicates(my_list))
my_list_with_duplicates=[1,2,3,2,3]
print("list has duplicates:", has_duplicates(my_list_with_duplicates))
```

```
List has duplicates: False
list has duplicates: True
```

[2]:
```
#12. Create a program that prints the prime numbers in a given range using a␣
 ↪`for` loop
def is_prime(num):
    if num<=1:
        return False
    for i in range(2, int (num**0.5)+1):
        if num%i==0:
            return False
    return True
def print_primes_in_range(start,end):
    print("Prime numbers in the range", start,"to", end,"are:")
    for num in range(start, end+1):
        if is_prime(num):
            print(num,end=" ")
start_num=int(input("enter teh starting number of the range:"))
end_num=int(input("enter the ending number of the range:"))
print_primes_in_range(start_num,end_num)
```

```
enter teh starting number of the range: 6
enter the ending number of the range: 9

Prime numbers in the range 6 to 9 are:
7
```

[5]:
```
#13.Develop a program that counts the number of vowels in a string using a␣
 ↪`for` loop.
def count_vowels(input_string):
    vowels="aeiouAEIOU"
    count=0
    for char in input_string:
        if char in vowels:
            count+=1
    return count
input_str=input("enter a string:")
print("Number of vowels in the string:", count_vowels(input_str))
```

```
enter a string: pooja najardhane

Number of vowels in the string: 7
```

[7]:
```
#14.Write a program to find the maximum element in a 2D list using a nested␣
 ↪`for` loop.
def find_max_element(matrix):
    max_element=float('-inf')
    for row in matrix:
        for element in row:
            if element>max_element:
                max_element=element
    return max_element
matrix=[
    [1,2,3],
    [4,5,6],
    [7,8,9]
]
print("Maximum element in the 2D list:", find_max_element(matrix))
```

```
Maximum element in the 2D list: 9
```

[12]:
```
#15.Create a program that removes all occurrences of a specific element from a␣
↪list using a `for` loop.
def remove_element(lst, target):
    new_lst = []
    for item in lst:
        if item != target:
            new_lst.append(item)
    return new_lst

def find_max_element(matrix):
    max_element = float('-inf')
    for row in matrix:
        for element in row:
            if element > max_element:
                max_element = element
    return max_element

my_list = [1, 2, 3, 3, 4, 5, 6, 2, 3]
target_element = 3
new_list = remove_element(my_list, target_element)
print("List after removing all occurrences of", target_element, ":", new_list)
```

```
List after removing all occurrences of 3 : [1, 2, 4, 5, 6, 2]
```

[18]:
```
#16.Develop a program that generates a multiplication table for numbers from 1␣
↪to 5 using a nested `for` loop
def generate_multiplication_table():
    for i in range(1, 6):
        print("Multiplication table for", i)
        print("----------------------")
        for j in range(1, 11):
            print(i, "x", j, "=", i * j)
        print()

generate_multiplication_table()
```

```
Multiplication table for 1
----------------------
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
```

```
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Multiplication table for 2
----------------------
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

Multiplication table for 3
----------------------
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Multiplication table for 4
----------------------
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

Multiplication table for 5
----------------------
5 x 1 = 5
```

```
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

[20]:
```
#17. Write a program that converts a list of Fahrenheit temperatures to Celsius␣
 ↪using a `for` loop
def fahrenheit_to_celsius(fahrenheit_temps):
    celsius_temps = []
    for fahrenheit in fahrenheit_temps:
        celsius = (fahrenheit - 32) * 5 / 9
        celsius_temps.append(celsius)
    return celsius_temps

fahrenheit_temperatures = [32, 68, 86, 104, 122]
celsius_temperatures = fahrenheit_to_celsius(fahrenheit_temperatures)

print("Fahrenheit temperatures:", fahrenheit_temperatures)
print("Celsius temperatures:", celsius_temperatures)
```

```
Fahrenheit temperatures: [32, 68, 86, 104, 122]
Celsius temperatures: [0.0, 20.0, 30.0, 40.0, 50.0]
```

[28]:
```
#18. Create a program to print the common elements from two lists using a `for`␣
 ↪loop.
def find_common_elements(list1, list2):
    common_elements = []
    for item in list1:
        if item in list2:
            common_elements.append(item)
    return common_elements

list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]
common_elements = find_common_elements(list1, list2)
print("Common elements:", common_elements)
```

```
Common elements: [4, 5]
```

[31]:
```
#19. Develop a program that prints the pattern of right-angled triangles using␣
 ↪a `for` loop. Use '*' to draw the
```

```
pattern
def print_right_angle_triangle(rows):
    for i in range(1, rows + 1):
        for j in range(1, i + 1):
            print("*", end="")
        print()

num_rows = int(input("Enter the number of rows for the triangle: "))
print_right_angle_triangle(num_rows)
```

```
Enter the number of rows for the triangle:  3
```

```
*
**
***
```

[34]:
```
#20.Write a program to find the greatest common divisor (GCD) of two numbers␣
 ↪using a `for` loop
def gcd(a, b):
    gcd_value = 1
    for i in range(1, min(a, b) + 1):
        if a % i == 0 and b % i == 0:
            gcd_value = i
    return gcd_value

num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
print("GCD of", num1, "and", num2, "is:", gcd(num1, num2))
```

```
Enter the first number:  5
Enter the second number:  8
```

```
GCD of 5 and 8 is: 1
```

[1]:
```
# Advanced Level:
#21. Create a program that calculates the sum of the digits of numbers in a␣
 ↪list using a list comprehension.
def sum_of_digits(numbers):
    return [sum(int(digit) for digit in str(number)) for number in numbers]


numbers = [123, 456, 789, 101]
sums = sum_of_digits(numbers)
print(sums)
```

```
[6, 15, 24, 2]
```

[3]:
```python
#22. Write a program to find the prime factors of a given number using a `for`␣
↪loop and list comprehension.
def prime_factors(n):
    factors = []
    divisor = 2

    while n > 1:
        while n % divisor == 0:
            factors.append(divisor)
            n //= divisor
        divisor += 1

    return factors


number = 56
factors = prime_factors(number)
print(f"The prime factors of {number} are: {factors}")
```

```
The prime factors of 56 are: [2, 2, 2, 7]
```

[4]:
```python
#23. Develop a program that extracts unique elements from a list and stores␣
↪them in a new list using a list comprehension.
def extract_unique_elements(input_list):
    seen = set()
    unique_elements = [x for x in input_list if x not in seen and not seen.
↪add(x)]
    return unique_elements


input_list = [1, 2, 2, 3, 4, 4, 5, 5, 5, 6]
unique_elements = extract_unique_elements(input_list)
print(f"The unique elements are: {unique_elements}")
```

```
The unique elements are: [1, 2, 3, 4, 5, 6]
```

[9]:
```python
#24. Create a program that generates a list of all palindromic numbers up to a␣
↪specified limit using a list comprehension.
def generate_palindromic_numbers(limit):
     return [num for num in range(limit + 1) if str(num) == str(num)[::-1]]


limit = 200
palindromic_numbers = generate_palindromic_numbers(limit)
print(f"Palindromic numbers up to {limit}: {palindromic_numbers}")
```

```
Palindromic numbers up to 200: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44,
55, 66, 77, 88, 99, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191]
```

[10]:
```
# 25. Write a program to flatten a nested list using list comprehension.
def flatten_nested_list(nested_list):
    return [item for sublist in nested_list for item in sublist]


nested_list = [[1, 2, 3], [4, 5], [6, 7, 8, 9]]
flattened_list = flatten_nested_list(nested_list)
print(f"Flattened list: {flattened_list}")
```

```
Flattened list: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[11]:
```
#26. Develop a program that computes the sum of even and odd numbers in a list␣
 ↪separately using list comprehension.
def sum_even_odd(numbers):
    sum_even = sum([num for num in numbers if num % 2 == 0])
    sum_odd = sum([num for num in numbers if num % 2 != 0])
    return sum_even, sum_odd


numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sum_even, sum_odd = sum_even_odd(numbers)
print(f"Sum of even numbers: {sum_even}")
print(f"Sum of odd numbers: {sum_odd}")
```

```
Sum of even numbers: 30
Sum of odd numbers: 25
```

[12]:
```
#27. Create a program that generates a list of squares of odd numbers between 1␣
 ↪and 10 using list comprehension.
squares_of_odd_numbers = [x**2 for x in range(1, 11) if x % 2 != 0]
print(squares_of_odd_numbers)
```

```
[1, 9, 25, 49, 81]
```

[14]:
```
#28. Write a program that combines two lists into a dictionary using list␣
 ↪comprehension.
keys=['a','b','c']
value=[1,2,3]
combined_dict={key: value for key, value in zip(keys,value)}
print("combined Dictionary:", combined_dict)
```

```
combined Dictionary: {'a': 1, 'b': 2, 'c': 3}
```

[18]:
```
#29. Develop a program that extracts the vowels from a string and stores them␣
 ↪in a list using list comprehension
input_string="Hello,world!"
```

```
vowel=[char for char in input_string if char.lower()in 'aeiou']
print("vowel in the string:", vowel)
```

```
vowel in the string: ['e', 'o', 'o']
```

[20]:
```
# 30. Create a program that removes all non-numeric characters from a list of␣
↪strings using list comprehension.
list_of_string=["abc123","456def","ghi789","10jkl"]
numeric_string=[''.join(char for char in string if char.isdigit())for string in␣
↪list_of_string]
print("list of string with only numberic characters:", numeric_string)
```

```
list of string with only numberic characters: ['123', '456', '789', '10']
```

[10]:
```
# Challenge Level
#31. Write a program to generate a list of prime numbers using the Sieve of␣
↪Eratosthenes algorithm and list comprehension.
def sieve_of_eratosthenes(limit):
    primes = [True] * (limit + 1)
    primes[0] = primes[1] = False
    for i in range(2, int(limit**0.5) + 1):
        if primes[i]:
            primes[i*i:limit+1:i] = [False] * len(primes[i*i:limit+1:i])
    return [i for i in range(2, limit+1) if primes[i]]

limit = int(input("Enter the upper limit for prime numbers: "))
prime_numbers = sieve_of_eratosthenes(limit)
print("Prime numbers up to", limit, ":", prime_numbers, sep=", ")
```

```
Enter the upper limit for prime numbers:  45

Prime numbers up to, 45, :, [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43]
```

[15]:
```
#32.Create a program that generates a list of all Pythagorean triplets up to a␣
↪specified limit using list comprehension
def pythogorean_triplets(limit):
    return[(a,b,c)for a in range (1,limit)for b in range(a,limit) for c in␣
↪range(b, limit) if a **2+b**2==c**2]
limit=int(input("enter the upper limit for pythogoren triplets:"))
triplets= pythogorean_triplets(limit)
print("pythogorean triplets up to ", limit, ":", triplets)
```

```
enter the upper limit for pythogoren triplets: 30

pythogorean triplets up to  30 : [(3, 4, 5), (5, 12, 13), (6, 8, 10), (7, 24,
25), (8, 15, 17), (9, 12, 15), (10, 24, 26), (12, 16, 20), (15, 20, 25), (20,
21, 29)]
```

```
[20]: #33. Develop a program that generates a list of all possible combinations of␣
       ↪two lists using list comprehension
      list1 = [1, 2, 3]
      list2 = ['a', 'b', 'c']
      combinations = [(x, y) for x in list1 for y in list2]
      print("All possible combinations of two lists:")
      for combination in combinations:
          print(combination)
```

```
All possible combinations of two lists:
(1, 'a')
(1, 'b')
(1, 'c')
(2, 'a')
(2, 'b')
(2, 'c')
(3, 'a')
(3, 'b')
(3, 'c')
```

```
[22]: #34. Write a program that calculates the mean, median, and mode of a list of␣
       ↪numbers using list comprehension.
      import statistics

      numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

      mean = sum(numbers) / len(numbers)

      median = statistics.median(numbers)

      print("Mean:", mean)
      print("Median:", median)
```

```
Mean: 5.5
Median: 5.5
```

```
[23]: #35.Create a program that generates Pascal's triangle up to a specified number␣
       ↪of rows using list comprehension.
      def generate_pascals_triangle(rows):
          triangle = [[1]]
          for _ in range(1, rows):
              row = [1] + [triangle[-1][i] + triangle[-1][i+1] for i in␣
       ↪range(len(triangle[-1]) - 1)] + [1]
              triangle.append(row)
          return triangle

      def print_pascals_triangle(triangle):
```

```
    for row in triangle:
        print(" ".join(map(str, row)).center(len(triangle[-1]) * 3))

rows = int(input("Enter the number of rows for Pascal's triangle: "))
pascals_triangle = generate_pascals_triangle(rows)
print_pascals_triangle(pascals_triangle)
```

```
Enter the number of rows for Pascal's triangle:  3

    1
   1 1
  1 2 1
```

[24]:
```
# 36. Develop a program that calculates the sum of the digits of a factorial of␣
↪numbers from 1 to 5 using list comprehension.
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

factorials = [factorial(i) for i in range(1, 6)]

sums_of_digits = [sum(int(digit) for digit in str(factorial)) for factorial in␣
↪factorials]

print("Sums of digits of factorials from 1 to 5:", sums_of_digits)
```

```
Sums of digits of factorials from 1 to 5: [1, 2, 6, 6, 3]
```

[25]:
```
# 37. Write a program that finds the longest word in a sentence using list␣
↪comprehension.
sentence = input("Enter a sentence: ")

words = sentence.split()

word_lengths = [len(word) for word in words]

max_length = max(word_lengths)

longest_words = [word for word in words if len(word) == max_length]

print("Longest word(s) in the sentence:", longest_words)
```

```
Enter a sentence:  my name is pooja najardhane
```

```
Longest word(s) in the sentence: ['najardhane']
```

[26]:
```
#38.Create a program that filters a list of strings to include only those with␣
↪more than three vowels using list comprehension
def count_vowels(word):
    return sum(1 for char in word if char.lower() in 'aeiou')

strings = ["hello", "world", "python", "programming", "ai", "openai",␣
↪"example", "vowels"]

filtered_strings = [word for word in strings if count_vowels(word) > 3]

print("Strings with more than three vowels:", filtered_strings)
```

```
Strings with more than three vowels: ['openai']
```

[27]:
```
# 39. Develop a program that calculates the sum of the digits of numbers from 1␣
↪to 1000 using list comprehension.
def sum_of_digits(n):
    return sum(int(digit) for digit in str(n))

total_sum_of_digits = sum(sum_of_digits(number) for number in range(1, 1001))

print("Total sum of digits for numbers from 1 to 1000:", total_sum_of_digits)
```

```
Total sum of digits for numbers from 1 to 1000: 13501
```

[28]:
```
#40. Write a program that generates a list of prime palindromic numbers using␣
↪list comprehension
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

def is_palindrome(n):
    return str(n) == str(n)[::-1]
```

```
prime_palindromic_numbers = [num for num in range(2, 1000) if is_prime(num) and␣
↪is_palindrome(num)]

print("Prime palindromic numbers:", prime_palindromic_numbers)
```

```
Prime palindromic numbers: [2, 3, 5, 7, 11, 101, 131, 151, 181, 191, 313, 353,
373, 383, 727, 757, 787, 797, 919, 929]
```

```
[ ]:
```