for-loops

March 20, 2024

Simple:

Printing numbers from 1 to 10 using a for loop:

[1]:
```
for i in range(1, 11):
    print(i)
```

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

Calculating the sum of all numbers in a list using a for loop:

[2]:
```
numbers = [1, 2, 3, 4, 5]
total = 0

for num in numbers:
    total += num

print("Sum of all numbers:", total)
```

```
Sum of all numbers: 15
```

Printing the characters of a string in reverse order using a for loop:

[3]:
```
string = "hello"

for char in reversed(string):
    print(char)
```

```
o
l
l
```

```
e
h
```

Finding the factorial of a given number using a for loop:

[4]:
```
num = 5
factorial = 1

for i in range(1, num + 1):
    factorial *= i

print("Factorial of", num, "is:", factorial)
```

```
Factorial of 5 is: 120
```

Printing the multiplication table of a given number using a for loop:

[5]:
```
num = 7

for i in range(1, 11):
    print(num, "x", i, "=", num * i)
```

```
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
```

Counting the number of even and odd numbers in a list using a for loop:

[6]:
```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even_count = 0
odd_count = 0

for num in numbers:
    if num % 2 == 0:
        even_count += 1
    else:
        odd_count += 1

print("Number of even numbers:", even_count)
print("Number of odd numbers:", odd_count)
```

```
Number of even numbers: 4
Number of odd numbers: 5
```

Printing the squares of numbers from 1 to 5 using a for loop:

[8]:
```
for i in range(1, 6):
    print("Square of", i, "is:", i * i)
```

```
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
```

Finding the length of a string without using the len() function:

[9]:
```
string = "hello"
length = 0

for char in string:
    length += 1

print("Length of the string:", length)
```

```
Length of the string: 5
```

Calculating the average of a list of numbers using a for loop:

[10]:
```
numbers = [10, 20, 30, 40, 50]
total = 0

for num in numbers:
    total += num

average = total / len(numbers)
print("Average of the numbers:", average)
```

```
Average of the numbers: 30.0
```

Printing the first n Fibonacci numbers using a for loop:

[11]:
```
n = 10
fibonacci_sequence = [0, 1]

for i in range(2, n):
    next_num = fibonacci_sequence[-1] + fibonacci_sequence[-2]
    fibonacci_sequence.append(next_num)

print("First", n, "Fibonacci numbers:", fibonacci_sequence)
```

```
First 10 Fibonacci numbers: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Intermediate:

Checking if a given list contains any duplicates using a for loop:

[12]:
```
def has_duplicates(lst):
    seen = set()
    for item in lst:
        if item in seen:
            return True
        seen.add(item)
    return False

# Example usage:
my_list = [1, 2, 3, 4, 5, 1]
if has_duplicates(my_list):
    print("The list contains duplicates.")
else:
    print("The list does not contain duplicates.")
```

```
The list contains duplicates.
```

Printing the prime numbers in a given range using a for loop:

[13]:
```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

start = 10
end = 20

print("Prime numbers between", start, "and", end, "are:")
for num in range(start, end + 1):
    if is_prime(num):
        print(num)
```

```
Prime numbers between 10 and 20 are:
11
13
17
19
```

Counting the number of vowels in a string using a for loop:

[14]:
```
def count_vowels(string):
    vowels = "aeiouAEIOU"
    count = 0
    for char in string:
```

```
        if char in vowels:
            count += 1
    return count

# Example usage:
text = "Hello World"
print("Number of vowels:", count_vowels(text))
```

```
Number of vowels: 3
```

Finding the maximum element in a 2D list using a nested for loop:

[ ]:
```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

max_value = float('-inf')

for row in matrix:
    for element in row:
        if element > max_value:
            max_value = element

print("Maximum element in the 2D list:", max_value)
```

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

Removing all occurrences of a specific element from a list using a for loop:

[16]:
```
def remove_element(lst, element):
    return [x for x in lst if x != element]

# Example usage:
my_list = [1, 2, 3, 4, 3, 5]
element_to_remove = 3
new_list = remove_element(my_list, element_to_remove)
print("List after removing", element_to_remove, ":", new_list)
```

```
List after removing 3 : [1, 2, 4, 5]
```

Generating a multiplication table for numbers from 1 to 5 using a nested for loop:

[17]:
```
for i in range(1, 6):
    for j in range(1, 11):
        print(i, "x", j, "=", i * j)
    print()  # Print an empty line after each table
```

```
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

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

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

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

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
```

Converting a list of Fahrenheit temperatures to Celsius using a for loop:

[18]:
```
fahrenheit_temperatures = [32, 68, 86, 104]
celsius_temperatures = []

for temp in fahrenheit_temperatures:
    celsius = (temp - 32) * 5/9
    celsius_temperatures.append(celsius)

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

```
Celsius temperatures: [0.0, 20.0, 30.0, 40.0]
```

Printing the common elements from two lists using a for loop:

[19]:
```
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]

common_elements = []

for item in list1:
    if item in list2:
        common_elements.append(item)

print("Common elements:", common_elements)
```

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

Printing the pattern of right-angled triangles using a for loop:

[20]:
```
rows = 5

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

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

Finding the greatest common divisor (GCD) of two numbers using a for loop:

[21]:
```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

num1 = 24
num2 = 36

print("GCD of", num1, "and", num2, "is:", gcd(num1, num2))
```

```
GCD of 24 and 36 is: 12
```

Advanced Level:

Calculating the sum of the digits of numbers in a list using a list comprehension:

[22]:
```
numbers = [123, 456, 789]

digit_sums = [sum(int(digit) for digit in str(num)) for num in numbers]

print("Sum of digits for each number:", digit_sums)
```

```
Sum of digits for each number: [6, 15, 24]
```

Finding the prime factors of a given number using a for loop and list comprehension:

[23]:
```
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
print("Prime factors of", number, "are:", prime_factors(number))
```

```
Prime factors of 56 are: [2, 2, 2, 7]
```

Extracting unique elements from a list and storing them in a new list using a list comprehension

[24]:
```
original_list = [1, 2, 2, 3, 4, 4, 5, 5]

unique_elements = list(set(original_list))

print("Unique elements:", unique_elements)
```

```
Unique elements: [1, 2, 3, 4, 5]
```

Generating a list of all palindromic numbers up to a specified limit using a list comprehension:

[25]:
```
limit = 100

palindromic_numbers = [num for num in range(1, limit + 1) if str(num) ==␣
 ↪str(num)[::-1]]

print("Palindromic numbers up to", limit, ":", palindromic_numbers)
```

```
Palindromic numbers up to 100 : [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55,
66, 77, 88, 99]
```

Flattening a nested list using list comprehension:

[26]:
```
nested_list = [[1, 2, 3], [4, 5], [6, 7, 8]]

flattened_list = [item for sublist in nested_list for item in sublist]

print("Flattened list:", flattened_list)
```

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

Computing the sum of even and odd numbers in a list separately using list comprehension:

[27]:
```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

even_sum = sum(num for num in numbers if num % 2 == 0)
odd_sum = sum(num for num in numbers if num % 2 != 0)

print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
```

```
Sum of even numbers: 20
Sum of odd numbers: 25
```

Generating a list of squares of odd numbers between 1 and 10 using list comprehension:

[28]:
```
squares_of_odds = [num**2 for num in range(1, 11) if num % 2 != 0]

print("Squares of odd numbers between 1 and 10:", squares_of_odds)
```

```
Squares of odd numbers between 1 and 10: [1, 9, 25, 49, 81]
```

Combining two lists into a dictionary using list comprehension:

[29]:
```
keys = ['a', 'b', 'c']
values = [1, 2, 3]

combined_dict = {key: value for key, value in zip(keys, values)}
```

```
print("Combined dictionary:", combined_dict)
```

```
Combined dictionary: {'a': 1, 'b': 2, 'c': 3}
```

Extracting vowels from a string and storing them in a list using list comprehension:

[30]:
```
text = "Hello World"
vowels = [char for char in text if char.lower() in 'aeiou']

print("Vowels in the string:", vowels)
```

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

Removing all non-numeric characters from a list of strings using list comprehension:

[31]:
```
strings = ['123', 'abc', '456', 'def', '789']

numeric_strings = [s for s in strings if s.isdigit()]

print("Numeric strings:", numeric_strings)
```

```
Numeric strings: ['123', '456', '789']
```

Challenge level

Generating a list of prime numbers using the Sieve of Eratosthenes algorithm and list comprehension:

[32]:
```
def sieve_of_eratosthenes(n):
    primes = [True] * (n + 1)
    primes[0], primes[1] = False, False
    p = 2
    while p * p <= n:
        if primes[p]:
            for i in range(p * p, n + 1, p):
                primes[i] = False
        p += 1
    return [i for i in range(n + 1) if primes[i]]

limit = 100
prime_numbers = sieve_of_eratosthenes(limit)
print("Prime numbers up to", limit, ":", prime_numbers)
```

```
Prime numbers up to 100 : [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Generating a list of all Pythagorean triplets up to a specified limit using list comprehension:

[33]:
```
limit = 10
```

```
pythagorean_triplets = [(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]
print("Pythagorean triplets up to", limit, ":", pythagorean_triplets)
```

```
Pythagorean triplets up to 10 : [(3, 4, 5)]
```

Generating a list of all possible combinations of two lists using list comprehension:

[34]:
```
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:", combinations)
```

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

Calculating the mean, median, and mode of a list of numbers using list comprehension:

[35]:
```
import statistics

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

mean = sum(numbers) / len(numbers)
median = statistics.median(numbers)
mode = statistics.mode(numbers)

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

```
Mean: 5.0
Median: 5
Mode: 1
```

Generating Pascal's triangle up to a specified number of rows using list comprehension:

[37]:
```
def generate_pascals_triangle(rows):
    triangle = [[1]]
    for i in range(1, rows):
        row = [1]
        for j in range(1, i):
            row.append(triangle[i-1][j-1] + triangle[i-1][j])
        row.append(1)
        triangle.append(row)
    return triangle

num_rows = 5
pascals_triangle = generate_pascals_triangle(num_rows)
```

```
for row in pascals_triangle:
    print(row)
```

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
```

Calculating the sum of the digits of a factorial of numbers from 1 to 5 using list comprehension:

[38]:
```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

sum_of_digits = [sum(int(digit) for digit in str(factorial(num))) for num in␣
 ↪range(1, 6)]
print("Sum of digits of factorials from 1 to 5:", sum_of_digits)
```

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

Finding the longest word in a sentence using list comprehension:

[39]:
```
sentence = "This is a sample sentence to find the longest word"
longest_word = max(sentence.split(), key=lambda word: len(word))
print("Longest word in the sentence:", longest_word)
```

```
Longest word in the sentence: sentence
```

Filtering a list of strings to include only those with more than three vowels using list comprehension:

[40]:
```
strings = ["hello", "world", "programming", "python", "artificial",␣
 ↪"intelligence"]

filtered_strings = [s for s in strings if sum(1 for char in s if char.lower()␣
 ↪in 'aeiou') > 3]
print("Strings with more than three vowels:", filtered_strings)
```

```
Strings with more than three vowels: ['artificial', 'intelligence']
```

Generating a list of prime palindromic numbers using list comprehension:

[42]:
```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
```

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

limit = 1000
prime_palindromes = [num for num in range(2, limit) if is_prime(num) and␣
 ↪is_palindrome(num)]
print("Prime palindromic numbers up to", limit, ":", prime_palindromes)
```

```
Prime palindromic numbers up to 1000 : [2, 3, 5, 7, 11, 101, 131, 151, 181, 191,
313, 353, 373, 383, 727, 757, 787, 797, 919, 929]
```