Assignment : Python - For loop

November 19, 2023

1 Assignment : Python - For loop

1.1 Basic Level:

1.1.1 1. Write a Python program to print the numbers from 1 to 10 using a `for` loop.

[3]:
```
for i in range(1,11):
 print(i)
```

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

–> using range function and for loop to print numbers from 1 to 10

1.1.2 2. Create a program that calculates the sum of all numbers in a list using a `for` loop.

[12]:
```
l=[1,2,3,4]  # Sample list of numbers
s=0 # Initialize a variable to store the sum
for i in l: # Iterate over each element in the list
    s+=i # Add the current number to the sum
print("sum is",s) # Print the result
```

```
sum is 10
```

1.1.3 3. Write a program to print the characters of a string in reverse order using a for loop.

[21]:
```
# Take user input for a string
input_string = input("Enter a string: ")

# Initialize an empty string to store the reversed characters
reversed_string = ""

# Use a for loop to iterate through the characters of the input string in␣
 ↪reverse order
for char in reversed(input_string):
    reversed_string += char

# Print the reversed string
print("Reversed string:", reversed_string)
```

```
Enter a string:  hello
```

```
Reversed string: olleh
```

1.1.4 4. Develop a program that finds the factorial of a given number using a for loop.

[20]:
```
# Take user input for the number for which factorial is to be calculated
s = int(input("Enter a number for which factorial to be calculated: "))

# Initialize the factorial variable to 1
factorial = 1

# Use a for loop to iterate from 1 to the given number
for i in range(1, s + 1):
    # Multiply each iteration value with the current factorial
    factorial *= i

# Print the calculated factorial
print("Factorial of the number is", factorial)
```

```
Enter a number for which factorial to be calculated:  4
```

```
Factorial of the number is 24
```

1.1.5 5.Create a program to print the multiplication table of a given number using a for loop.

[29]:
```
num = int(input("Enter a number: "))

# Use a for loop to iterate from 1 to 10 (inclusive)
for i in range(1, 11):
    # Multiply the entered number by the current loop variable 'i'
    result = num * i
  # Print the result
    print(result)
```

```
Enter a number:  3

3
6
9
12
15
18
21
24
27
30
```

1.1.6 6. Write a program that counts the number of even and odd numbers in a list using a for loop.

[45]:
```
l = [1, 2, 3, 4, 5, 6]
e = 0 # variable to count even numbers
o = 0 # variable to count odd numbers

for i in l:
    if i % 2 == 0:
        e += 1 ##e += 1: Increments the even count (e) when an even number is␣
  ↪encountered.
    else:
        o += 1 ## o += 1: Increments the odd count (o) when an odd number is␣
  ↪encountered.

print("Even numbers in the list:", e) ##print("Even numbers in the list:", e):␣
  ↪Prints the count of even numbers
print("Odd numbers in the list:", o)  ##print("Odd numbers in the list:", o):␣
  ↪Prints the count of odd numbers.
```

```
Even numbers in the list: 3
Odd numbers in the list: 3
```

1.1.7 7. Develop a program that prints the squares of numbers from 1 to 5 using a for loop.

[32]:
```
print(" square of the number from 1 to 5 is")
for i in range(1,6):
    print(i*i)
```

```
##It uses a for loop to iterate through the range of numbers from 1 to 5  and␣
↪prints the square of each number using the expression i * i
```

```
 square of the number from 1 to 5 is
1
4
9
16
25
```

1.1.8 8. Create a program to find the length of a string without using the `len()` function.

[40]:
```
s=input(" input string ") # Take user input for a string
l=0 # Initialize a variable to store the length of the string
for i in s: #Use a for loop to iterate through each character in the string
    l+=1   # Increment the length variable for each character

# Print the length of the string
print("Length of the string is:", l)
```

```
 input string  asdf'

Length of the string is: 5
```

1.1.9 9. Write a program that calculates the average of a list of numbers using a `for` loop.

[49]:
```
l=[1,2,3,4,5,6] # Given list of numbers
s=0 # Initialize a variable to store the sum of numbers

for i in l: # Use a for loop to iterate through each number in the list
    s+=i   # Add the current number to the sum

avg=s/len(l) # Calculate the average by dividing the sum by the length of the␣
↪list
print("average is",avg)
# Print the calculated average
```

```
average is 3.5
```

1.1.10 10. Develop a program that prints the first `n` Fibonacci numbers using a `for` loop.

[2]:
```
n = int(input("Enter the number of Fibonacci numbers to print: "))# Get the␣
 ↪number of Fibonacci numbers to print

fib_sequence = [0, 1]# Initialize the first two numbers in the Fibonacci␣
 ↪sequence


for i in range(2, n): # Generate the Fibonacci sequence using for loop
    next_fibonacci = fib_sequence[i - 1] + fib_sequence[i - 2]
    fib_sequence.append(next_fibonacci)

print(f"First {n} Fibonacci numbers:") # Print the first n Fibonacci numbers
for fib_number in fib_sequence:
    print(fib_number, end=" ")
```

```
Enter the number of Fibonacci numbers to print:  4

First 4 Fibonacci numbers:
0 1 1 2
```

1.2 Intermediate Level:

1.2.1 11. Write a program to check if a given list contains any duplicates using a `for` loop.

[5]:
```
my_li = [1, 2, 3, 4, 5]

# Flag to track
has_duplicates = False

# Check for duplicates
for i in range(len(my_li)):
    for j in range(i + 1, len(my_li)):
        if my_li[i] == my_li[j]:
            has_duplicates = True
            break

# Print the result
if has_duplicates:
    print("The list contains duplicates.")
else:
    print("The list does not contain duplicates.")
```

```
The list does not contain duplicates.
```

[7]:
```
### the has_duplicates flag remains False unless a duplicate is found.
### The inner loop runs to completion, checking all pairs of elements,
### and if it finds a duplicate, it sets the has_duplicates flag to True.
```

1.2.2 12. Create a program that prints the prime numbers in a given range using a for loop.

[9]:
```
start_range = 1
end_range = 100

# Print prime numbers in the given range
print(f"Prime numbers in the range {start_range} to {end_range}:")
for number in range(start_range, end_range + 1):
    if number > 1:
        is_prime = True
        for i in range(2, int(number**0.5) + 1):
            if number % i == 0:
                is_prime = False
                break
        if is_prime:
            print(number, end=" ")
```

```
Prime numbers in the range 1 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
```

1.2.3 13. Develop a program that counts the number of vowels in a string using a for loop.

[14]:
```
str=input("enter the string") # Take user input for a string
v=0
# Initialize a variable to count vowels

for i in str:# Iterate over each character in the input string
    if i.lower() in ('a','e','i','o','u'):  # Check if the lowercase version of␣
 ↪the current character is a vowel
         v+=1 # If it is a vowel, increment the vowel count

print("number of vowels",v) # Print the total count of vowels
```

```
enter the string EeE

number of vowels 3
```

1.2.4 14. Write a program to find the maximum element in a 2D list using a nested `for` loop.

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

# Initialize max_element with the minimum possible integer value
max_element = float('-inf')

# Nested loops to iterate through each element in the 2D list
for row in matrix:
    for element in row:
        # Compare the current element with the current max_element
        if element > max_element:
            max_element = element

# Print the maximum element found in the 2D list
print("Maximum element:", max_element)
```

```
Maximum element: 9
```

1.2.5 15. Create a program that removes all occurrences of a specific element from a list using a `for` loop.

[4]:
```
# list
my_list = [1, 2, 3, 4, 2, 5, 2, 6,2,2]

# Element to be removed
element_to_remove = 2

# for loop to iterate over the list in reverse
for item in reversed(my_list):
                     ##Using reversed() ensures that you iterate over the list␣
 ↪in reverse order,
                     ##and it's particularly important when you are modifying␣
 ↪the list while iterating
    if item == element_to_remove:  # Check if the current item is equal to the␣
 ↪element to be removed
        my_list.remove(item)  # Remove the element if it matches
        # Note that this modifies the list during iteration, which is safe ␣
 ↪because we are iterating in reverse.

# Print the updated list
```

```
print("List after removing", element_to_remove, ":", my_list)
```

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

1.2.6 16. Develop a program that generates a multiplication table for numbers from 1 to 5 using a nested for loop.

[9]:
```
for i in range(1,6): ##This loop iterates over values from 1 to 5
    for j in range(1,11): ## Inside the outer loop, there's another loop that␣
 ↪iterates over values from 1 to 10
        s=i*j #the variable s is assigned the product of i and j, representing␣
 ↪the result of the multiplication.
        print(s,end='\t')# Print the product, followed by a tab character (use␣
 ↪'\t' for indentation)
    print() # Move to the next line after printing each row
```

```
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      28      32      36      40
5       10      15      20      25      30      35      40      45      50
```

1.2.7 17. Write a program that converts a list of Fahrenheit temperatures to Celsius using a for loop.

[7]:
```
f=[98,99,100] # List of Fahrenheit temperatures
for i in f:
# Iterate over each temperature in the list
    c=((i-32)*5)/9 # Convert Fahrenheit to Celsius using the formula (C = (F -␣
 ↪32) * 5 / 9)
    print(round(c,1)) # Printing the result, rounding to one decimal place

```

```
36.7
37.2
37.8
```

1.2.8 18. Create a program to print the common elements from two lists using a for loop.

[12]:
```
# Two lists
l1=[1,2,3,4,5]
l2=[4,5,6,7]
# Nested loop to compare elements in both lists
for i in l1:
    for j in l2:
         # Check if the elements are equal
```

```
        if i==j:
            print(i)  # Print the common element
```

```
4
5
```

1.2.9 19. Develop a program that prints the pattern of right-angled triangles using a `for` loop. Use '*' to draw the pattern

[5]:
```
#  number of rows for the triangle
num_rows = 5

# Use afor loop to iterate over each row
for i in range(1, num_rows + 1):
    # Print '*' for each column in the row
    for j in range(i):
        print('*', end=' ')
    # Move to the next line after printing each row
    print()
```

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

1.2.10 20. Write a program to find the greatest common divisor (GCD) of two numbers using a `for` loop.

[6]:
```
x=int(input("enter first number"))
y=int(input("enter second number"))

# Determine the smaller of the two numbers
smaller = min(x,y)

# Initialize GCD to 1 (minimum possible GCD)
gcd = 1

# Use a for loop to iterate from 1 to the smaller number
for i in range(1, smaller + 1):
    # Check if both numbers are divisible by the current iterator
    if (x % i == 0) and (y % i == 0):
        # If divisible, update the GCD
        gcd = i

# Print the result
print(f"The GCD of {x} and {y} is: {gcd}")
```

```
enter first number 10
enter second number 20

The GCD of 10 and 20 is: 10
```

1.3 Advanced Level:

1.3.1 21. Create a program that calculates the sum of the digits of numbers in a list using a list comprehension.

[3]:
```
#  list of numbers
numbers = [123, 45, 678, 9]
# List comprehension offers a shorter syntax when you want to create a new list␣
  ↪based on the values of an existing list.
# list comprehension to calculate the sum of digits for each number
# newlist = [expression for item in iterable if condition == True]
sum_of_digits_list = [sum(int(digit) for digit in str(num)) for num in numbers]

# Print the result
print("Original List:", numbers)
print("Sum of Digits List:", sum_of_digits_list)
```

```
Original List: [123, 45, 678, 9]
Sum of Digits List: [6, 9, 21, 9]
```

1.3.2 22. Write a program to find the prime factors of a given number using a for loop and list comprehension.

[4]:
```
#Get a number from the user
number = int(input("Enter a number: "))

# prime factors using list comprehension
prime_factors = [i for i in range(2, number + 1) if (number % i == 0) and all(i␣
  ↪% j != 0 for j in range(2, int(i**0.5) + 1))]

# Print the result
print(f"Prime factors of {number}: {prime_factors}")
```

```
Enter a number:  12

Prime factors of 12: [2, 3]
```

1.3.3 23. Develop a program that extracts unique elements from a list and stores them in a new list using a list comprehension.

[5]:
```
# list
original_list = [1, 2, 2, 3, 4, 4, 5, 6, 6]

# unique elements using list comprehension
```

```
unique_elements = list(set(original_list))

# Print the result
print("Original List:", original_list)
print("Unique Elements List:", unique_elements)
```

```
Original List: [1, 2, 2, 3, 4, 4, 5, 6, 6]
Unique Elements List: [1, 2, 3, 4, 5, 6]
```

1.3.4 24. Create a program that generates a list of all palindromic numbers up to a specified limit using a list comprehension.

[7]:
```
# Function to check if a number is a palindrome
def is_palindrome(num):
    return str(num) == str(num)[::-1]

# Specify the limit
limit = int(input("Enter the limit: "))

# Generate a list of palindromic numbers up to the limit using list␣
↪comprehension
palindromic_numbers = [i for i in range(limit + 1) if is_palindrome(i)]

# Print the result
print(f"Palindromic numbers up to {limit}: {palindromic_numbers}")
```

```
Enter the limit:  89

Palindromic numbers up to 89: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55,
66, 77, 88]
```

1.3.5 25. Write a program to flatten a nested list using list comprehension.

[8]:
```
# nested list
nested_list = [[1, 2, 3], [4, [5, 6]], [7, 8, 9]]

# Flatten the nested list using list comprehension
flattened_list = [element for sublist in nested_list for element in (sublist if␣
↪isinstance(sublist, list) else [sublist])]

# Print the result
print("Nested List:", nested_list)
print("Flattened List:", flattened_list)
```

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

1.3.6 26. Develop a program that computes the sum of even and odd numbers in a list separately using list comprehension.

[9]:
```
# list of numbers (replace it with your own list)
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Separate even and odd numbers using list comprehension
even_numbers = [num for num in numbers if num % 2 == 0]
odd_numbers = [num for num in numbers if num % 2 != 0]

# Compute the sum of even and odd numbers
sum_even = sum(even_numbers)
sum_odd = sum(odd_numbers)

# Print the result
print("Original List:", numbers)
print("Even Numbers:", even_numbers)
print("Sum of Even Numbers:", sum_even)
print("Odd Numbers:", odd_numbers)
print("Sum of Odd Numbers:", sum_odd)
```

```
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Even Numbers: [2, 4, 6, 8]
Sum of Even Numbers: 20
Odd Numbers: [1, 3, 5, 7, 9]
Sum of Odd Numbers: 25
```

1.3.7 27. Create a program that generates a list of squares of odd numbers between 1 and 10 using list comprehension.

[10]:
```
# Generate a list of squares of odd numbers between 1 and 10 using list␣
  ↪comprehension
squares_of_odd_numbers = [num**2 for num in range(1, 11) if num % 2 != 0]

# Print the result
print("Squares of Odd Numbers:", squares_of_odd_numbers)
```

```
Squares of Odd Numbers: [1, 9, 25, 49, 81]
```

1.3.8 28. Write a program that combines two lists into a dictionary using list comprehension.

[11]:
```
# lists (replace them with your own lists)
keys = ['a', 'b', 'c']
values = [1, 2, 3]

# Combine lists into a dictionary using list comprehension
combined_dict = {key: value for key, value in zip(keys, values)}
```

```
# Print the result
print("Keys:", keys)
print("Values:", values)
print("Combined Dictionary:", combined_dict)
```

```
Keys: ['a', 'b', 'c']
Values: [1, 2, 3]
Combined Dictionary: {'a': 1, 'b': 2, 'c': 3}
```

1.3.9 29. Develop a program that extracts the vowels from a string and stores them in a list using list comprehension.

[14]:
```
# input a string from the user
input_string = input("Enter a string: ")

# Extract vowels using list comprehension
vowels = [char for char in input_string if char.lower() in 'aeiou']

# Print the result
print("Input String:", input_string)
print("Vowels:", vowels)
```

```
Enter a string:  qwertyui

Input String: qwertyui
Vowels: ['e', 'u', 'i']
```

1.3.10 30. Create a program that removes all non-numeric characters from a list of strings using list comprehension.

[15]:
```
# list of strings
string_list = ['abc123', '45def', '678ghi', '9jkl']

# Remove non-numeric characters using list comprehension
numeric_strings = [''.join(char for char in s if char.isdigit()) for s in␣
 ↪string_list]

# Print the result
print("Original List:", string_list)
print("Numeric Strings:", numeric_strings)
```

```
Original List: ['abc123', '45def', '678ghi', '9jkl']
Numeric Strings: ['123', '45', '678', '9']
```

1.4 Challenge Level:

1.4.1 31. Write a program to generate a list of prime numbers using the Sieve of Eratosthenes algorithm and list comprehension.

[17]:
```
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(range(i*i, limit+1, i))

    return [num for num in range(2, limit + 1) if primes[num]]

# Input a limit from the user
limit = int(input("Enter a limit: "))

# Generate a list of prime numbers using Sieve of Eratosthenes and list␣
 ↪comprehension
prime_numbers = sieve_of_eratosthenes(limit)

# Print the result
print(f"Prime numbers up to {limit}: {prime_numbers}")
```

```
Enter a limit:  34

Prime numbers up to 34: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]
```

1.4.2 32. Create a program that generates a list of all Pythagorean triplets up to a specified limit using list comprehension.

[20]:
```
# Function to check if a set of three numbers forms a Pythagorean triplet
def is_pythagorean_triplet(a, b, c):
    return a**2 + b**2 == c**2

# Input a limit from the user
limit = int(input("Enter a limit: "))

# Generate a list of Pythagorean triplets using list comprehension
pythagorean_triplets = [
    (a, b, c)
    for a in range(1, limit + 1)
    for b in range(a, limit + 1)
    for c in range(b, limit + 1)
    if is_pythagorean_triplet(a, b, c)
]
```

```
# Print the result
print(f"Pythagorean triplets up to {limit}: {pythagorean_triplets}")
```

```
Enter a limit:  10
```

```
Pythagorean triplets up to 10: [(3, 4, 5), (6, 8, 10)]
```

1.4.3 33. Develop a program that generates a list of all possible combinations of two lists using list comprehension.

[21]:
```
# lists
list1 = [1, 2, 3]
list2 = ['a', 'b', 'c']

# all possible combinations of two lists using list comprehension
combinations = [(x, y) for x in list1 for y in list2]

# Print the result
print("List 1:", list1)
print("List 2:", list2)
print("Combinations:", combinations)
```

```
List 1: [1, 2, 3]
List 2: ['a', 'b', 'c']
Combinations: [(1, 'a'), (1, 'b'), (1, 'c'), (2, 'a'), (2, 'b'), (2, 'c'), (3,
'a'), (3, 'b'), (3, 'c')]
```

1.4.4 34. Write a program that calculates the mean, median, and mode of a list of numbers using list comprehension.

[23]:
```
from statistics import mean, median, mode

# list of numbers
numbers = [2, 4, 4, 4, 5, 5, 7, 9]

# Calculate mean, median, and mode using list comprehension
list_mean = mean(numbers)
list_median = median(numbers)
list_mode = mode(numbers)

# Print the result
print("Original List:", numbers)
print("Mean:", list_mean)
print("Median:", list_median)
print("Mode:", list_mode)
```

```
Original List: [2, 4, 4, 4, 5, 5, 7, 9]
Mean: 5
```

```
Median: 4.5
Mode: 4
```

1.4.5 35. Create a program that generates Pascals triangle up to a specified number of rows using list comprehension.

[24]:
```
def generate_pascals_triangle(rows):
    triangle = [[1] * (row + 1) for row in range(rows)]

    for i in range(2, rows):
        for j in range(1, i):
            triangle[i][j] = triangle[i-1][j-1] + triangle[i-1][j]

    return triangle

# Input number of rows from the user
num_rows = int(input("Enter the number of rows for Pascal's triangle: "))

# Generate Pascal's triangle using list comprehension
pascals_triangle = generate_pascals_triangle(num_rows)

# Print the result
print(f"Pascal's Triangle up to {num_rows} rows:")
for row in pascals_triangle:
    print(row)
```

```
Enter the number of rows for Pascal's triangle:  5
```

```
Pascal's Triangle up to 5 rows:
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
```

1.4.6 36. Develop a program that calculates the sum of the digits of a factorial of numbers from 1 to 5 using list comprehension.

[26]:
```
from math import factorial

# Calculate the sum of digits of the factorial of numbers from 1 to 5 using␣
 ↪list comprehension
sum_of_digits = [sum(int(digit) for digit in str(factorial(num))) for num in␣
 ↪range(1, 6)]

# Print the result
print("Factorials:", [factorial(num) for num in range(1, 6)])
print("Sum of Digits:", sum_of_digits)
```

```
Factorials: [1, 2, 6, 24, 120]
Sum of Digits: [1, 2, 6, 6, 3]
```

1.4.7 37. Write a program that finds the longest word in a sentence using list comprehension.

[29]:
```python
# Input a sentence from the user
sentence = input("Enter a sentence: ")

# Find the longest word using list comprehension
longest_word = max((word.strip(".,!?") for word in sentence.split()), key=len)

# Print the result
print(f"The longest word in the sentence is: {longest_word}")
```

```
Enter a sentence:  hsuh ssssss sssswwjwuw snjssjsjsjsjnsjnj

The longest word in the sentence is: snjssjsjsjsjnsjnj
```

1.4.8 38. Create a program that filters a list of strings to include only those with more than three vowels using list comprehension.

[30]:
```python
#  list of strings from the user
string_list = input("Enter a list of strings (comma-separated): ").split(',')

# Filter strings with more than three vowels using list comprehension
filtered_strings = [s for s in string_list if sum(1 for char in s.lower() if␣
 ↪char in 'aeiou') > 3]

# Print the result
print(f"Original List: {string_list}")
print(f"Filtered List (more than three vowels): {filtered_strings}")
```

```
Enter a list of strings (comma-separated):  hbhb,vgyg,gyvu

Original List: ['hbhb', 'vgyg', 'gyvu']
Filtered List (more than three vowels): []
```

1.4.9 39. Develop a program that calculates the sum of the digits of numbers from 1 to 1000 using list comprehension.

[35]:
```python
# Calculate and print the sum of digits for numbers from 1 to 1000
print("Sum of Digits for Numbers 1 to 1000:")
for num in range(1, 1001):
    print(f"{num}: {sum(int(digit) for digit in str(num))}")
```

```
Sum of Digits for Numbers 1 to 1000:
1: 1
2: 2
```

```
3: 3
4: 4
5: 5
6: 6
7: 7
8: 8
9: 9
10: 1
11: 2
12: 3
13: 4
14: 5
15: 6
16: 7
17: 8
18: 9
19: 10
20: 2
21: 3
22: 4
23: 5
24: 6
25: 7
26: 8
27: 9
28: 10
29: 11
30: 3
31: 4
32: 5
33: 6
34: 7
35: 8
36: 9
37: 10
38: 11
39: 12
40: 4
41: 5
42: 6
43: 7
44: 8
45: 9
46: 10
47: 11
48: 12
49: 13
50: 5
```

```
51: 6
52: 7
53: 8
54: 9
55: 10
56: 11
57: 12
58: 13
59: 14
60: 6
61: 7
62: 8
63: 9
64: 10
65: 11
66: 12
67: 13
68: 14
69: 15
70: 7
71: 8
72: 9
73: 10
74: 11
75: 12
76: 13
77: 14
78: 15
79: 16
80: 8
81: 9
82: 10
83: 11
84: 12
85: 13
86: 14
87: 15
88: 16
89: 17
90: 9
91: 10
92: 11
93: 12
94: 13
95: 14
96: 15
97: 16
98: 17
```

```
99: 18
100: 1
101: 2
102: 3
103: 4
104: 5
105: 6
106: 7
107: 8
108: 9
109: 10
110: 2
111: 3
112: 4
113: 5
114: 6
115: 7
116: 8
117: 9
118: 10
119: 11
120: 3
121: 4
122: 5
123: 6
124: 7
125: 8
126: 9
127: 10
128: 11
129: 12
130: 4
131: 5
132: 6
133: 7
134: 8
135: 9
136: 10
137: 11
138: 12
139: 13
140: 5
141: 6
142: 7
143: 8
144: 9
145: 10
146: 11
```

```
147: 12
148: 13
149: 14
150: 6
151: 7
152: 8
153: 9
154: 10
155: 11
156: 12
157: 13
158: 14
159: 15
160: 7
161: 8
162: 9
163: 10
164: 11
165: 12
166: 13
167: 14
168: 15
169: 16
170: 8
171: 9
172: 10
173: 11
174: 12
175: 13
176: 14
177: 15
178: 16
179: 17
180: 9
181: 10
182: 11
183: 12
184: 13
185: 14
186: 15
187: 16
188: 17
189: 18
190: 10
191: 11
192: 12
193: 13
194: 14
```

```
195: 15
196: 16
197: 17
198: 18
199: 19
200: 2
201: 3
202: 4
203: 5
204: 6
205: 7
206: 8
207: 9
208: 10
209: 11
210: 3
211: 4
212: 5
213: 6
214: 7
215: 8
216: 9
217: 10
218: 11
219: 12
220: 4
221: 5
222: 6
223: 7
224: 8
225: 9
226: 10
227: 11
228: 12
229: 13
230: 5
231: 6
232: 7
233: 8
234: 9
235: 10
236: 11
237: 12
238: 13
239: 14
240: 6
241: 7
242: 8
```

```
243: 9
244: 10
245: 11
246: 12
247: 13
248: 14
249: 15
250: 7
251: 8
252: 9
253: 10
254: 11
255: 12
256: 13
257: 14
258: 15
259: 16
260: 8
261: 9
262: 10
263: 11
264: 12
265: 13
266: 14
267: 15
268: 16
269: 17
270: 9
271: 10
272: 11
273: 12
274: 13
275: 14
276: 15
277: 16
278: 17
279: 18
280: 10
281: 11
282: 12
283: 13
284: 14
285: 15
286: 16
287: 17
288: 18
289: 19
290: 11
```

```
291: 12
292: 13
293: 14
294: 15
295: 16
296: 17
297: 18
298: 19
299: 20
300: 3
301: 4
302: 5
303: 6
304: 7
305: 8
306: 9
307: 10
308: 11
309: 12
310: 4
311: 5
312: 6
313: 7
314: 8
315: 9
316: 10
317: 11
318: 12
319: 13
320: 5
321: 6
322: 7
323: 8
324: 9
325: 10
326: 11
327: 12
328: 13
329: 14
330: 6
331: 7
332: 8
333: 9
334: 10
335: 11
336: 12
337: 13
338: 14
```

```
339: 15
340: 7
341: 8
342: 9
343: 10
344: 11
345: 12
346: 13
347: 14
348: 15
349: 16
350: 8
351: 9
352: 10
353: 11
354: 12
355: 13
356: 14
357: 15
358: 16
359: 17
360: 9
361: 10
362: 11
363: 12
364: 13
365: 14
366: 15
367: 16
368: 17
369: 18
370: 10
371: 11
372: 12
373: 13
374: 14
375: 15
376: 16
377: 17
378: 18
379: 19
380: 11
381: 12
382: 13
383: 14
384: 15
385: 16
386: 17
```

```
387: 18
388: 19
389: 20
390: 12
391: 13
392: 14
393: 15
394: 16
395: 17
396: 18
397: 19
398: 20
399: 21
400: 4
401: 5
402: 6
403: 7
404: 8
405: 9
406: 10
407: 11
408: 12
409: 13
410: 5
411: 6
412: 7
413: 8
414: 9
415: 10
416: 11
417: 12
418: 13
419: 14
420: 6
421: 7
422: 8
423: 9
424: 10
425: 11
426: 12
427: 13
428: 14
429: 15
430: 7
431: 8
432: 9
433: 10
434: 11
```

```
435: 12
436: 13
437: 14
438: 15
439: 16
440: 8
441: 9
442: 10
443: 11
444: 12
445: 13
446: 14
447: 15
448: 16
449: 17
450: 9
451: 10
452: 11
453: 12
454: 13
455: 14
456: 15
457: 16
458: 17
459: 18
460: 10
461: 11
462: 12
463: 13
464: 14
465: 15
466: 16
467: 17
468: 18
469: 19
470: 11
471: 12
472: 13
473: 14
474: 15
475: 16
476: 17
477: 18
478: 19
479: 20
480: 12
481: 13
482: 14
```

```
483: 15
484: 16
485: 17
486: 18
487: 19
488: 20
489: 21
490: 13
491: 14
492: 15
493: 16
494: 17
495: 18
496: 19
497: 20
498: 21
499: 22
500: 5
501: 6
502: 7
503: 8
504: 9
505: 10
506: 11
507: 12
508: 13
509: 14
510: 6
511: 7
512: 8
513: 9
514: 10
515: 11
516: 12
517: 13
518: 14
519: 15
520: 7
521: 8
522: 9
523: 10
524: 11
525: 12
526: 13
527: 14
528: 15
529: 16
530: 8
```

```
531: 9
532: 10
533: 11
534: 12
535: 13
536: 14
537: 15
538: 16
539: 17
540: 9
541: 10
542: 11
543: 12
544: 13
545: 14
546: 15
547: 16
548: 17
549: 18
550: 10
551: 11
552: 12
553: 13
554: 14
555: 15
556: 16
557: 17
558: 18
559: 19
560: 11
561: 12
562: 13
563: 14
564: 15
565: 16
566: 17
567: 18
568: 19
569: 20
570: 12
571: 13
572: 14
573: 15
574: 16
575: 17
576: 18
577: 19
578: 20
```

```
579: 21
580: 13
581: 14
582: 15
583: 16
584: 17
585: 18
586: 19
587: 20
588: 21
589: 22
590: 14
591: 15
592: 16
593: 17
594: 18
595: 19
596: 20
597: 21
598: 22
599: 23
600: 6
601: 7
602: 8
603: 9
604: 10
605: 11
606: 12
607: 13
608: 14
609: 15
610: 7
611: 8
612: 9
613: 10
614: 11
615: 12
616: 13
617: 14
618: 15
619: 16
620: 8
621: 9
622: 10
623: 11
624: 12
625: 13
626: 14
```

```
627: 15
628: 16
629: 17
630: 9
631: 10
632: 11
633: 12
634: 13
635: 14
636: 15
637: 16
638: 17
639: 18
640: 10
641: 11
642: 12
643: 13
644: 14
645: 15
646: 16
647: 17
648: 18
649: 19
650: 11
651: 12
652: 13
653: 14
654: 15
655: 16
656: 17
657: 18
658: 19
659: 20
660: 12
661: 13
662: 14
663: 15
664: 16
665: 17
666: 18
667: 19
668: 20
669: 21
670: 13
671: 14
672: 15
673: 16
674: 17
```

```
675: 18
676: 19
677: 20
678: 21
679: 22
680: 14
681: 15
682: 16
683: 17
684: 18
685: 19
686: 20
687: 21
688: 22
689: 23
690: 15
691: 16
692: 17
693: 18
694: 19
695: 20
696: 21
697: 22
698: 23
699: 24
700: 7
701: 8
702: 9
703: 10
704: 11
705: 12
706: 13
707: 14
708: 15
709: 16
710: 8
711: 9
712: 10
713: 11
714: 12
715: 13
716: 14
717: 15
718: 16
719: 17
720: 9
721: 10
722: 11
```

```
723: 12
724: 13
725: 14
726: 15
727: 16
728: 17
729: 18
730: 10
731: 11
732: 12
733: 13
734: 14
735: 15
736: 16
737: 17
738: 18
739: 19
740: 11
741: 12
742: 13
743: 14
744: 15
745: 16
746: 17
747: 18
748: 19
749: 20
750: 12
751: 13
752: 14
753: 15
754: 16
755: 17
756: 18
757: 19
758: 20
759: 21
760: 13
761: 14
762: 15
763: 16
764: 17
765: 18
766: 19
767: 20
768: 21
769: 22
770: 14
```

```
771: 15
772: 16
773: 17
774: 18
775: 19
776: 20
777: 21
778: 22
779: 23
780: 15
781: 16
782: 17
783: 18
784: 19
785: 20
786: 21
787: 22
788: 23
789: 24
790: 16
791: 17
792: 18
793: 19
794: 20
795: 21
796: 22
797: 23
798: 24
799: 25
800: 8
801: 9
802: 10
803: 11
804: 12
805: 13
806: 14
807: 15
808: 16
809: 17
810: 9
811: 10
812: 11
813: 12
814: 13
815: 14
816: 15
817: 16
818: 17
```

```
819: 18
820: 10
821: 11
822: 12
823: 13
824: 14
825: 15
826: 16
827: 17
828: 18
829: 19
830: 11
831: 12
832: 13
833: 14
834: 15
835: 16
836: 17
837: 18
838: 19
839: 20
840: 12
841: 13
842: 14
843: 15
844: 16
845: 17
846: 18
847: 19
848: 20
849: 21
850: 13
851: 14
852: 15
853: 16
854: 17
855: 18
856: 19
857: 20
858: 21
859: 22
860: 14
861: 15
862: 16
863: 17
864: 18
865: 19
866: 20
```

```
867: 21
868: 22
869: 23
870: 15
871: 16
872: 17
873: 18
874: 19
875: 20
876: 21
877: 22
878: 23
879: 24
880: 16
881: 17
882: 18
883: 19
884: 20
885: 21
886: 22
887: 23
888: 24
889: 25
890: 17
891: 18
892: 19
893: 20
894: 21
895: 22
896: 23
897: 24
898: 25
899: 26
900: 9
901: 10
902: 11
903: 12
904: 13
905: 14
906: 15
907: 16
908: 17
909: 18
910: 10
911: 11
912: 12
913: 13
914: 14
```

```
915: 15
916: 16
917: 17
918: 18
919: 19
920: 11
921: 12
922: 13
923: 14
924: 15
925: 16
926: 17
927: 18
928: 19
929: 20
930: 12
931: 13
932: 14
933: 15
934: 16
935: 17
936: 18
937: 19
938: 20
939: 21
940: 13
941: 14
942: 15
943: 16
944: 17
945: 18
946: 19
947: 20
948: 21
949: 22
950: 14
951: 15
952: 16
953: 17
954: 18
955: 19
956: 20
957: 21
958: 22
959: 23
960: 15
961: 16
962: 17
```

```
963: 18
964: 19
965: 20
966: 21
967: 22
968: 23
969: 24
970: 16
971: 17
972: 18
973: 19
974: 20
975: 21
976: 22
977: 23
978: 24
979: 25
980: 17
981: 18
982: 19
983: 20
984: 21
985: 22
986: 23
987: 24
988: 25
989: 26
990: 18
991: 19
992: 20
993: 21
994: 22
995: 23
996: 24
997: 25
998: 26
999: 27
1000: 1
```

1.4.10 40. Write a program that generates a list of prime palindromic numbers using list comprehension.

[36]:
```
def is_prime(number):
    if number < 2:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True

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

# Generate a list of prime palindromic numbers using list comprehension
prime_palindromes = [num for num in range(1, 1000) if is_prime(num) and␣
 ↪is_palindrome(num)]

# Print the result
print("Prime Palindromic Numbers up to 1000:", 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]
```

```
[ ]:
```