forloop assignment

February 3, 2024

[1]:
```
# for loop Assignment
# Basic Level:
```

[2]:
```
# 1. Write a Python program to print the numbers from 1 to 10 using a `for`␣
 ↪loop.
a=0
for i in range(10):
    i += 1
    print(i)
```

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

[3]:
```
# 2. Create a program that calculates the sum of all numbers in a list using a␣
 ↪`for` loop.
v=[1,2,3,4,5]
sum=0
for i in v:
    sum  +=i


print(sum)
```

```
15
```

[4]:
```
# 3. Write a program to print the characters of a string in reverse order using␣
 ↪a `for` loop
a=('hello')
```

```
b=a[::-1]
for num in a:
    reverse=b

print(reverse)
```

```
olleh
```

[5]:
```
# 4. Develop a program that finds the factorial of a given number using a `for` loop.
n=3
fact=1
for i in range(1,n+1):
    fact=fact*i

print(fact)
```

```
6
```

[6]:
```
# 5. Create a program to print the multiplication table of a given number using a `for` loop.
num= int(input("enter no"))
for i in range(1,11):
    print(f"{num} x {i}={num*i}" )
```

```
enter no2
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
```

[7]:
```
# 6.Write a program that counts the number of even and odd numbers in a list using a `for` loop.
a=[1,2,3,4,5,6,7,8,9,10]
evencount = 0
oddcount=0
for i in a:
    if i %2 ==0:
        evencount += 1
```

```
    else :
        oddcount +=1

print(f"evencount is {evencount}")
print(f"oddcount is {oddcount}")
```

```
evencount is 5
oddcount is 5
```

[8]:
```
# 7. Develop a program that prints the squares of numbers from 1 to 5 using a `for` loop.

for i in range(1,6):
    square = i*i

    print(f"square of {i} is {square}")
```

```
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
```

[9]:
```
# 8. Create a program to find the length of a string without using the `len()` function.
string=("hello")
count=0
for i in string :
    count += 1
    print( count)
```

```
1
2
3
4
5
```

[10]:
```
# 9. Write a program that calculates the average of a list of numbers using a `for` loop.
numbers=[10,20,30,40]
total=0
count=0
for i in numbers:
    total += i
    count+=1
average = total/count
```

```
print(f"the average of the list is:{average}")
```

```
the average of the list is:25.0
```

[11]:
```
# 10. Develop a program that prints the first `n` Fibonacci numbers using a␣
 ↪`for` loop.
def fibonacci(n):
    if n<=0:
        raise ValueError("n must be postive integer")
    a,b=0,1
    for i in range(n):
        yield(a)
        a,b=b,a+b
n=10

for i in fibonacci(n):
    print(i)
```

```
0
1
1
2
3
5
8
13
21
34
```

[12]:
```
# 11.Write a program to check if a given list contains any duplicates using a␣
 ↪`for` loop

def has_duplicates(list1):
    seen = set()
    for item in list1:
        if item in seen:
            return True
        seen.add(item)

    return False

list1=[1,2,3,2]
list2=[3,4,5]
print(has_duplicates(list1))
print(has_duplicates(list2))
```

```
True
```

```
False
```

[13]:
```
# 12. Create a program that prints the prime numbers in a given range using a␣
 ↪`for` loop
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 print_primes_in_range(start, end):
    print(f"Prime numbers in the range ({start}, {end}):")
    for num in range(start, end + 1):
        if is_prime(num):
            print(num, end=" ")

# Example usage:
start_range = int(input("Enter the start of the range: "))
end_range = int(input("Enter the end of the range: "))

print_primes_in_range(start_range, end_range)
```

```
Enter the start of the range: 2
Enter the end of the range: 10
Prime numbers in the range (2, 10):
2 3 5 7
```

[14]:
```
# 13. Develop a program that counts the number of vowels in a string using a␣
 ↪`for` loop
string=("hello")
vowels=('a','e','i','o','u','A','E','I','O','U')
count =0
for i in string:
    if i in vowels:
        count+=1
print(count)
```

```
2
```

[15]:
```
# 14. Write a program to find the maximum element in a 2D list using a nested␣
 ↪`for` loop.
def find_max_element(matrix):
    if not matrix or not matrix[0]:
        return None  # Return None for empty lists
```

```
    max_element = matrix[0][0]

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

    return max_element

# Example usage:
matrix = [
    [3, 5, 2],
    [9, 1, 8],
    [4, 7, 6]
]

max_element = find_max_element(matrix)
print("The maximum element in the 2D list is:", max_element)
```

```
The maximum element in the 2D list is: 9
```

[16]:
```
# 15. Create a program that removes all occurrences of a specific element from␣
 ↪a list using a `for` loop
def remove_element(list,rem_ele):
    res=[]
    for ele in list:
        if ele != rem_ele:
            res.append(ele)
    return(res)


list=[1,2,3,4,5,6]
rem_ele=2

filtered_list = remove_element(list,rem_ele)
print(filtered_list)
```

```
[1, 3, 4, 5, 6]
```

[17]:
```
#16. Develop a program that generates a multiplication table for numbers from 1␣
 ↪to 5 using a nested `for` loop
start=1
end =5
for i in range(start,end+1):
    for j in range(start,end+1):
        result = i*j
```

```
        print("{:2} x {:2} = {:2}".format ( i,j,result))
    print()
```

```
1 x  1 =  1
1 x  2 =  2
1 x  3 =  3
1 x  4 =  4
1 x  5 =  5

2 x  1 =  2
2 x  2 =  4
2 x  3 =  6
2 x  4 =  8
2 x  5 = 10

3 x  1 =  3
3 x  2 =  6
3 x  3 =  9
3 x  4 = 12
3 x  5 = 15

4 x  1 =  4
4 x  2 =  8
4 x  3 = 12
4 x  4 = 16
4 x  5 = 20

5 x  1 =  5
5 x  2 = 10
5 x  3 = 15
5 x  4 = 20
5 x  5 = 25
```

[18]:
```
# 17. Write a program that converts a list of Fahrenheit temperatures to␣
 ↪Celsius using a `for` loop
fahrenheit_temps=[32,50,77,80,104]
celsius_temps=[]
for fahrenheit_temp in fahrenheit_temps:
    celsius_temp=(fahrenheit_temp-32)*5/9
    celsius_temps.append(celsius_temp)
print(f"fahrenheit temparatures {fahrenheit_temps}")
print(f"celsius temparatures {celsius_temps}")
```

```
fahrenheit temparatures [32, 50, 77, 80, 104]
celsius temparatures [0.0, 10.0, 25.0, 26.666666666666668, 40.0]
```

[19]:
```
# 18. Create a program to print the common elements from two lists using a␣
 ↪`for` loop
l1=[1,2,3,4]
l2=[2,1,5,6,7]
commonelem=[]
for i in l1:
    if i in l2 :
        commonelem.append(i)
        print(f" common elements from list1 and list2 are {commonelem}")
```

```
 common elements from list1 and list2 are [1]
 common elements from list1 and list2 are [1, 2]
```

[20]:
```
# 19. Develop a program that prints the pattern of right-angled triangles using␣
 ↪a `for` loop. Use '*' to draw the pattern
num_rows = 5
for row in range(1,num_rows+1):
    print("*" * row )
```

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

[21]:
```
# 20. Write a program to find the greatest common divisor (GCD) of two numbers␣
 ↪using a `for` loop.
def find_gcd(a,b):
    if a<b:
        a,b=b,a
    for i in range(b,0,-1):
        if a % i ==0 and b %i==0:
            gcd = i
            break
    return gcd

num1 = 48
num2 = 18
result_gcd = find_gcd(num1,num2)
print(f"the gcd of {num1} and {num2} is:{result_gcd}")
```

```
the gcd of 48 and 18 is:6
```

[1]:
```
# 21.Create a program that calculates the sum of the digits of numbers in a␣
 ↪list using a list comprehension.
numbers = [123, 456, 789]
```

```
# Calculate the sum of the digits of each number in the list using a list␣
 ↪comprehension
sum_of_digits = [sum(int(digit) for digit in str(num)) for num in numbers]

print(sum_of_digits)
```

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

[28]:
```
# 22. Write a program to find the prime factors of a given number using a `for`␣
 ↪loop and list comprehension.
def prime_factors(number):
    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))]
    return factors

# Example usage:
input_number = 84

prime_factors_result = prime_factors(input_number)

print(f"The prime factors of {input_number} are:", prime_factors_result)
```

```
The prime factors of 84 are: [2, 3, 7]
```

[66]:
```
#23. Develop a program that extracts unique elements from a list and stores␣
 ↪them in a new list using a list comprehension
def unique_ele(list1):
    return(set(list1))
original_list = [1,2,3,4,1,1,1,2,2,2,5,5,6,6]
unq_ele_result = unique_ele(original_list)
print(f"original list:",original_list)
print(f"unq ele result:",unq_ele_result)

a= unq_ele_result
print(a)
```

```
original list: [1, 2, 3, 4, 1, 1, 1, 2, 2, 2, 5, 5, 6, 6]
unq ele result: {1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6}
```

[ ]:
```

```

[37]:
```
# 24. Create a program that generates a list of all palindromic numbers up to a␣
 ↪specified limit using a list comprehension
def palindromes(limit):
    palindrome=[
        [i for i in range(1,limit+1)
        if str(i)==str(i)[::-1]]
```

```
    ]
    return palindrome
limit = 50
palindrome = palindromes(limit)
print(f"palindromes up to {limit}:{palindrome}")
```

```
palindromes up to 50:[[1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44]]
```

[38]:
```
# 25. Write a program to flatten a nested list using list comprehension.
nest_list=[[1,2,3],[4,5],[6,7]]
flat_list=[i for sublist in nest_list  for i in sublist  ]
print(f"nested list: {nest_list}")
print(f"flat list: {flat_list}")
```

```
nested list: [[1, 2, 3], [4, 5], [6, 7]]
flat list: [1, 2, 3, 4, 5, 6, 7]
```

[1]:
```
# 26. Develop a program that computes the sum of even and odd numbers in a list␣
 ↪separately using list comprehension

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

# Calculate the sum of even and odd numbers using list comprehension
sum_of_even = sum(num for num in numbers if num % 2 == 0)
sum_of_odd = sum(num for num in numbers if num % 2 != 0)

print("Sum of even numbers:", sum_of_even)
print("Sum of odd numbers:", sum_of_odd)
```

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

[ ]:
```

```

[41]:
```
#27. Create a program that generates a list of squares of odd numbers between 1␣
 ↪and 10 using list comprehension.
odd_no = range(1,11,2)
squares = [n**2 for  n in odd_no]
print(f"squares of odd numbers between 1 and 10 ",squares)
```

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

[42]:
```
# 28.Write a program that combines two lists into a dictionary using list␣
 ↪comprehension.
names =["a","b","c","d"]
age =[10,20,30,40]
combined_dict = {name: age for name, age in zip(names,age) }
print(combined_dict)
```

```
{'a': 10, 'b': 20, 'c': 30, 'd': 40}
```

[43]:
```
#29.Develop a program that extracts the vowels from a string and stores them in␣
 ↪a list using list comprehension
text = "this is a program"
vowels=("a","e","i","o","u")
extracted_txt=[i for i in text.lower() if i in vowels]
print(extracted_txt)
```

```
['i', 'i', 'a', 'o', 'a']
```

[44]:
```
# 30. Create a program that removes all non-numeric characters from a list of␣
 ↪strings using list comprehension
strings = ['123abc' , '456def' , '789ghi']
allowed_char =set("0123456789")
numeric_strings=["".join(char for char in string if char in allowed_char) for␣
 ↪string in strings]
print(numeric_strings)
```

```
['123', '456', '789']
```

[45]:
```
#31. Write a program to generate a list of prime numbers using the Sieve of␣
 ↪Eratosthenes algorithm and list comprehension.
def sieve_of_eratosthenes(n):
    primes = [True] * (n+1)
    primes[0]=primes[1]=False
    for i in range(2,int(n**0.5),+1):

        if primes[i]:


            for j in range(i*i,n+1,i):

                primes[j]= False
                return [i for i,is_prime in  enumerate(primes) if is_prime]

primes=sieve_of_eratosthenes(100)
print(primes)
```

```
[2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
```

[46]:
```
# 32. Create a program that generates a list of all Pythagorean triplets up to␣
 ↪a specified limit using list comprehension
def pythagorean_triplets(limit):
```

```
            return [(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 a**2 + b**2 == c**2]

# Generate Pythagorean triplets up to 100
triplets = pythagorean_triplets(100)

# Print the list of triplets
for a, b, c in triplets:
 print(f"({a}, {b}, {c})")
```

```
(3, 4, 5)
(5, 12, 13)
(6, 8, 10)
(7, 24, 25)
(8, 15, 17)
(9, 12, 15)
(9, 40, 41)
(10, 24, 26)
(11, 60, 61)
(12, 16, 20)
(12, 35, 37)
(13, 84, 85)
(14, 48, 50)
(15, 20, 25)
(15, 36, 39)
(16, 30, 34)
(16, 63, 65)
(18, 24, 30)
(18, 80, 82)
(20, 21, 29)
(20, 48, 52)
(21, 28, 35)
(21, 72, 75)
(24, 32, 40)
(24, 45, 51)
(24, 70, 74)
(25, 60, 65)
(27, 36, 45)
(28, 45, 53)
(28, 96, 100)
(30, 40, 50)
(30, 72, 78)
(32, 60, 68)
(33, 44, 55)
(33, 56, 65)
```

```
(35, 84, 91)
(36, 48, 60)
(36, 77, 85)
(39, 52, 65)
(39, 80, 89)
(40, 42, 58)
(40, 75, 85)
(42, 56, 70)
(45, 60, 75)
(48, 55, 73)
(48, 64, 80)
(51, 68, 85)
(54, 72, 90)
(57, 76, 95)
(60, 63, 87)
(60, 80, 100)
(65, 72, 97)
```

[49]:
```python
# 33. Develop a program that generates a list of all possible combinations of two lists using list comprehension
def list_combination(list1,list2):
    return [(a,b) for a in list1 for b in list2]


list1=["a","b","c"]
list2=[1,2,3]
combination = list_combination(list1,list2)
print(f"combinations of {list1} and {list2}")
for a,b in combination:

    print(f"({a},{b})")
```

```
combinations of ['a', 'b', 'c'] and [1, 2, 3]
(a,1)
(a,2)
(a,3)
(b,1)
(b,2)
(b,3)
(c,1)
(c,2)
(c,3)
```

[2]:
```python
#34. Write a program that calculates the mean, median, and mode of a list of numbers using list comprehension.
def statistics(data):
    mean = sum(data)/len(data)
    data.sort()
```

```
    median=data[len(data) // 2] if len(data) % 2 else (data[len(data) // 2 - 1]␣
 ↪+ data[len(data) // 2]) / 2
    from collections import Counter
    mode = Counter(data).most_common(1)[0][0] if Counter(data).most_common(1)␣
 ↪else None

    return {"mean": mean, "median": median, "mode": mode}

# Example usage
data = [1, 2, 3, 4, 5, 5, 6, 7, 8]

statistics = statistics(data)

print(f"Mean: {statistics['mean']}")
print(f"Median: {statistics['median']}")
print(f"Mode: {statistics['mode']}")
```

```
Mean: 4.555555555555555
Median: 5
Mode: 5
```

[51]:
```
#35. Create a program that generates Pascal's triangle up to a specified number␣
 ↪of rows using list comprehension
def pascal_triangle(num_rows):
    triangle = [[1] for _ in range(num_rows)]
    for i in range(1, num_rows):

        for j in range(1, i + 1):

            triangle[i].append(triangle[i - 1][j - 1] + (triangle[i - 1][j] if␣
 ↪j < i else 0))
    return triangle

# Example usage
num_rows = 5

pascal_triangle = pascal_triangle(num_rows)

for row in pascal_triangle:
    print(" ".join(str(x) for x in row))
```

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

[3]:
```
# 36. Develop a program that calculates the sum of the digits of a factorial of␣
 ↪numbers from 1 to 5 using list comprehension.
from math import factorial
factorials = [factorial(i) for i in range(1,6)]

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

for i,factorial in enumerate(factorials):
    print(f"Factorial of {i+1}: {factorial}, Sum of digits : {digit_sums[i]}")
```

```
Factorial of 1: 1, Sum of digits : 1
Factorial of 2: 2, Sum of digits : 2
Factorial of 3: 6, Sum of digits : 6
Factorial of 4: 24, Sum of digits : 6
Factorial of 5: 120, Sum of digits : 3
```

[53]:
```
#37.Write a program that finds the longest word in a sentence using list␣
 ↪comprehension.
sentence = " the quick brown fox jumps over the lazy dog"
words = sentence.split()
long_word = max(words,key=len)
print(f"the longest word in sentence is :{long_word}")
```

```
the longest word in sentence is :quick
```

[4]:
```
# 38. Create a program that filters a list of strings to include only those␣
 ↪with more than three vowels using list comprehension
words = ["apple", "banana", "orange", "pineapple", "grapefruit", "kiwi",␣
 ↪"mango"]

vowels = "aeiou"

filtered_words = [word for word in words if sum(letter in vowels for letter in␣
 ↪word.lower()) > 3]

print(f"Words with more than 3 vowels: {', '.join(filtered_words)}")
```

```
Words with more than 3 vowels: pineapple, grapefruit
```

[5]:
```
# 39. Develop a program that calculates the sum of the digits of numbers from 1␣
 ↪to 1000 using list comprehension.
def sum_digits(n):
  return sum(int(digit) for digit in str(n))

# Generate a list of numbers from 1 to 1000
numbers = range(1, 1001)
```

```
# Calculate the sum of digits for each number using list comprehension
digit_sums = [sum_digits(number) for number in numbers]

# Calculate the total sum of digits
total_sum = sum(digit_sums)

# Print the total sum of digits
print(f"The sum of the digits of numbers from 1 to 1000 is: {total_sum}")
```

```
The sum of the digits of numbers from 1 to 1000 is: 13501
```

[56]:
```
#40. Write a program that generates a list of prime palindromic numbers using␣
 ↪list comprehension.

def is_prime(n):
 """
 Checks if a number is prime.

 Args:
   n: An integer.

 Returns:
   True if n is prime, False otherwise.
 """
 if n <= 1:
   return False
 for i in range(2, int(n**0.5) + 1):
   if n % i == 0:
     return False
 return True

def is_palindrome(n):
 """
 Checks if a number is a palindrome.

 Args:
   n: An integer.

 Returns:
   True if n is a palindrome, False otherwise.
 """
 original = n
 reversed_num = 0
 while n > 0:
   digit = n % 10
   reversed_num = reversed_num * 10 + digit
   n //= 10
```

```
 return original == reversed_num

# Generate a list of prime palindromic numbers
prime_palindromes = [n for n in range(1, 10000) if is_prime(n) and␣
 ↪is_palindrome(n)]

# Print the list of prime palindromic numbers
print(f"Prime palindromic numbers: {', '.join(str(n) for n in␣
 ↪prime_palindromes)}")
```

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

```
[ ]:
```