1. Print numbers from 1 to 10

```
# solution
for i in range(1,11):
  print(i)
```

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

2. Print the odd numbers less than 100

```
for i in range(1,100,2):
  print(i)
```

```
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
```

```
35
37
39
41
43
45
47
49
51
53
55
57
59
61
63
65
67
69
71
73
75
77
79
81
83
85
87
89
91
93
95
97
99
```

3. Print the multiplication table with 7

```
num = int(input("Enter any number"))
for i in range(1,11):
  print(num,'*', i , '=', num * i)
```

```
Enter any number7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
```

```
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

4. Print all the multiplication tables with numbers from 1 to 10

```
for i in range(1,11):
  for j in range(1,11):
    print(i,'*', j , '=', i * j)
```

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
```

```
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
```

5. Calculate the sum of numbers from 1 to 10

```
## function definition
def is_sum(num):
  sum = 0
  for a in num:
    sum = sum + a
  print(sum)


## driving code

num = [i for i in range(1,11)]
is_sum(num)
```

```
55
```

6. Calculate 10!

```
## function definition

def factorial(num):
  if(num == 0 or num == 1):
    return 1
  else:
    return num * factorial(num-1)

## driving code

num = 10
result = factorial(num)
print("The factorial is ",result)
```

```
The factorial is  3628800
```

7. Calculate the sum of even numbers greaterthan 10 and less than 30

```
sum = 0
for i in range(12,30):
  if i%2==0:
    sum = sum + i

print(sum)
```

```
180
```

8. Create a function that will convertfrom Celsius to Fahrenheit

```
## function definition

def is_convert(celsius):
  far = celsius * (9/5) + 32
  return far
```

```
## driving code
far = 0
celsius = float(input("Enter celsius temp."))
result = is_convert(celsius)
print(result)
```

```
Enter celsius temp.180
356.0
```

9. Create a function that will convertfrom Fahrenheit to Celsius

```
## function definition

def is_convertor(fahrenheit):
  cel = 5/9*(fahrenheit - 32)
  return cel

## driving code
cel = 0
fahrenheit = float(input("Enter fahrenheit temp."))
result = is_convertor(fahrenheit)
print(result)
```

```
Enter fahrenheit temp.90
32.22222222222222
```

10. Calculate the sum of numbers in an array of numbers

```
arr = [1,2,3,4,5]
sum = 0
for i in range(len(arr)):
  sum = sum + arr[i]

print(sum)
```

```
15
```

11. Calculate the average of the numbers in an array of numbers

```
arr = [1,2,3,4,5]
sum = 0
for i in range(len(arr)):
  sum = sum + arr[i]
  average = sum/len(arr)

print(sum)
print(average)
```

```
15
3.0
```

12. Create a function thatreceives an array of numbers as argument and returns an array containing only the positive numbers

```
## function definition

def is_positive(arr):
  positive_num = []
  for i in range(len(arr)):
    if arr[i]>0:
      positive_num.append(arr[i])

  return positive_num

## Driving code

arr = [1,2,-3,4,88,-5,-22]
result = is_positive(arr)
print(result)
```

```
[1, 2, 4, 88]
```

13. Find the maximum number in an array of numbers

```
## function definition

def is_max(array):
  max = 0
  for i in range(len(array)):
    if array[i] >  max:
      max = array[i]
  return max

## driver code

array = [3,5,6,7,8,2,0,89,5]
# array = [1,2,3,4,5,6,7,8,9,10]
result = is_max(array)
print(result)
```

```
89
```

14. Print the first 10 Fibonacci numbers without recursion

```
## Function definition

def is_fib(num):
  arr = []
  n1 = 0
  n2 = 1
  arr.append(n1)
  arr.append(n2)

  for i in range(num):
    nextFib = n1 + n2
    arr.append(nextFib)
    n1 = n2
    n2 = nextFib

  return arr

## driving code
```

```
num = int(input("Enter the number"))
result = is_fib(num)
print(result)
```

```
Enter the number10
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

15.Create a function that will find the nth Fibonacci number using recursion

```
## function defintion

def fibRec(n):
  if(n<=0):
    return None
  if(n==1 or n==2):
    return n-1
  return fibRec(n-1) + fibRec(n-2)


## driving code
n = int(input("Enter any number"))
result = fibRec(n)
print(result)
```

```
Enter any number10
34
```

16. Create afunction that will return aBoolean specifying if anumber is prime

```
# -------------------- 1st Method ----------------------------

## function definition
def is_prime(number):
    # Check if the number is less than 2
    if number < 2:
        return False

    # Check for factors from 2 to the square root of the number
    for i in range(2, int(number ** 0.5) + 1):
```

```
            if number % i == 0:
                return False

        return True

    # Get input from the user
    num = int(input("Enter a number: "))
    result = is_prime(num)
    print(result)

    # ----------------------------- 2nd Method ---------------------------------

    # num = 30
    # flag = False
    # for i in range(2,num):
    #   if num == 0:
    #     print("Not a prime")
    #   elif num > 1:
    #     if(num%i)==0:
    #       flag = True
    #       break



    # if flag:
    #   print("Not Prime")
    # else:
    #   print("Prime")
```

```
Enter a number: 7
True
```

```
Not Prime
```

17. Calculate the sum of digits of a positive integer number

```
def sumOfDigits(num):
  sum = 0
  while(num>0):
    digit = num %10
    sum += digit
    num = (num - digit)/10
  return sum
```

```
num = 12345
sum = sumOfDigits(num)
print(sum)
```

```
15.0
```

18. Print the first 100 prime numbers

```
## function definition
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

# Iterate through numbers from 1 to 100
for num in range(1, 101):
    if is_prime(num):
        print(num)
```

```
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
```

19. Create a function that will return in an array the first“p” prime number greater than“n”

```
def get_primes(p, n):
    primes = []
    num = n + 1

    while len(primes) < p:
        if is_prime(num):
            primes.append(num)
        num += 1

    return primes

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

# Get input from the user
p = int(input("Enter the number of prime numbers you want: "))
n = int(input("Enter the value of 'n': "))

# Call the function to get the prime numbers
res = get_primes(p, n)

# Print the result
```

```
print("The first", p, "prime numbers greater than", n, "are:", res)
```

```
Enter the number of prime numbers you want: 10
Enter the value of 'n': 2
The first 10 prime numbers greater than 2 are: [3, 5, 7, 11, 13, 17, 19, 23, 29, 31]
```

20. Rotate an array to the left 1 position

```
def rotate_left(arr):
    if len(arr) <= 1:
        return arr  # No rotation needed for empty or single-element array

    first_element = arr[0]  # Store the first element

    # Shift each element one position to the left
    for i in range(len(arr) - 1):
        arr[i] = arr[i + 1]

    arr[-1] = first_element  # Move the first element to the end

    return arr

# Example usage
original_array = [1, 2, 3, 4, 5]
print("Original Array:", original_array)
rotated_array = rotate_left(original_array)

print("Rotated Array:", rotated_array)
```

```
Original Array: [1, 2, 3, 4, 5]
Rotated Array: [2, 3, 4, 5, 1]
```

21. Rotate an array to the right 1 position

```
def rot_right(arr):
    if len(arr) <= 1:
        return arr

    last_element = arr[-1]
    for i in range(len(arr) - 1, 0, -1):
```

```
        arr[i] = arr[i - 1]

    arr[0] = last_element
    return arr

## driving code

arr = [1,2,3,4,5]
print("Original_array",arr)
rot_arr = rot_right(arr)
print("Rotated_array",arr)
```

```
Original_array [1, 2, 3, 4, 5]
Rotated_array [5, 1, 2, 3, 4]
```

22. Reverse an array

```
## function definition

arr = [1, 2, 3, 4, 5];
print("Original array: ");
for i in range(0, len(arr)):
    print(arr[i]),
print("Array in reverse order: ");
#Loop through the array in reverse order
for i in range(len(arr)-1, -1, -1):
    print(arr[i])
```

```
Original array:
1
2
3
4
5
Array in reverse order:
5
4
3
2
1
```

23. Reverse a string

```
## Method - 1

# original_string = "Hello, World!"
# reversed_string = original_string[::-1]
# print(reversed_string)


## Method - 2
original_string = "Hello, World!"
reversed_string = ''
for char in reversed(original_string):
    reversed_string += char
print(reversed_string)
```

```
!dlroW ,olleH
```

24. Create a function that will merge two arrays and return the result as a new

array

```
### 1st Method -----------------
# arr1 = [1,2,3,4,5]
# arr2 = [6,7,8,9,10]
# arr1.extend(arr2)
# print(arr1)

## 2nd Method------------------

## function defintion
def is_merged(arr1,arr2):
  arr1 +=arr2
  return arr1

## driving code

arr1 = [i for i in range(1,6)]
arr2 = [j for j in range(6,11)]
```

```
merged_arr = is_merged(arr1,arr2)
print(merged_arr)
```

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

25. Create a function that will receive two arrays of numbers as arguments and return an array composed of allthe numbers that are either in the first array or second array but not in both

```
## function defintion
# def re_dup(arr1,arr2):
#   arr3 = []
#   arr3 = set(arr1 + arr2)
#   return arr3


# ## driving code

# arr1 = [1,2,3,4,5]
# arr2 = [6,7,1,2,8]
# result = re_dup(arr1,arr2)
# print(result)

## ---------------------- Method - 2 ----------------------

## function defintion
def get_unique(arr1,arr2):
  unique_array = []
  for num in arr1:
    if num not in arr2:
      unique_array.append(num)
  for num in arr2:
    if num not in arr1:
      unique_array.append(num)

  return unique_array
## driving code
```

```
arr1 = [i for i in range(1,6)]
arr2 = [1,5,6,7,3]
result = get_unique(arr1,arr2)
print(result)
```

```
[2, 4, 6, 7]
```

26. Create a function that will receive two arrays and will return an array with elements that are in the first array but not in the second

```
## Method - 1

def get_diff(arr1,arr2):
  diffrence = [x for x in arr1 if x not in arr2]
  return diffrence


## driving code

arr1 = [2,34,6,4,3,6]
arr2 = [2,5,7,3,44,6]
res = get_diff(arr1,arr2)
print(res)

## Method - 2----------------------------------------------------

## function defintion


# def is_diff(arr1,arr2):
#   set_A = set(arr1)
#   set_B = set(arr2)
#   diff = set_A - set_B
#   return diff

# ## driving code
# arr1 = [2,34,6,4,3,6]
# arr2 = [2,5,7,3,44,6]
# result = is_diff(arr1,arr2)
# print(result)
```

```
[34, 4]
```

27. Create a function that will receive an array of numbers as argument and will return a new array with distinct elements

```
## function definition

def is_distinct(arr):
  unique_list = []

  for num in arr:
    if num not in unique_list:
      unique_list.append(num)

  return unique_list


## driving code

arr = [2,3,4,5,6,2,34,5,6,4,3,5]
result = is_distinct(arr)
print(result)
```

```
[2, 3, 4, 5, 6, 34]
```

28. Calculate the sum of first 100 prime numbers and return them in an array

```
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 get_first_100_primes_sum():
    primes = []
```

```
    num = 2
    while len(primes) < 100:
        if is_prime(num):
            primes.append(num)
        num += 1
    primes_sum = sum(primes)
    return primes, primes_sum

prime_numbers, sum_of_primes = get_first_100_primes_sum()
print("Prime Numbers:", prime_numbers)
print("Sum of Prime Numbers:", sum_of_primes)
```

```
Prime Numbers: [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, 101, 103, 107, 109, 113, 127, 131, 137,
Sum of Prime Numbers: 24133
```

29. Print the distance between the first 100 prime numbers

```
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_prime_distances():
    previous_prime = 2
    prime_count = 1
    num = 3
    while prime_count < 100:
        if is_prime(num):
            distance = num - previous_prime
            print(f"Distance between {previous_prime} and {num}: {distance}")
            previous_prime = num
            prime_count += 1
        num += 1

print_prime_distances()
```

```
Distance between 2 and 3: 1
Distance between 3 and 5: 2
Distance between 5 and 7: 2
Distance between 7 and 11: 4
Distance between 11 and 13: 2
Distance between 13 and 17: 4
Distance between 17 and 19: 2
Distance between 19 and 23: 4
Distance between 23 and 29: 6
Distance between 29 and 31: 2
Distance between 31 and 37: 6
Distance between 37 and 41: 4
Distance between 41 and 43: 2
Distance between 43 and 47: 4
Distance between 47 and 53: 6
Distance between 53 and 59: 6
Distance between 59 and 61: 2
Distance between 61 and 67: 6
Distance between 67 and 71: 4
Distance between 71 and 73: 2
Distance between 73 and 79: 6
Distance between 79 and 83: 4
Distance between 83 and 89: 6
Distance between 89 and 97: 8
Distance between 97 and 101: 4
Distance between 101 and 103: 2
Distance between 103 and 107: 4
Distance between 107 and 109: 2
Distance between 109 and 113: 4
Distance between 113 and 127: 14
Distance between 127 and 131: 4
Distance between 131 and 137: 6
Distance between 137 and 139: 2
Distance between 139 and 149: 10
Distance between 149 and 151: 2
Distance between 151 and 157: 6
Distance between 157 and 163: 6
Distance between 163 and 167: 4
Distance between 167 and 173: 6
Distance between 173 and 179: 6
Distance between 179 and 181: 2
Distance between 181 and 191: 10
Distance between 191 and 193: 2
Distance between 193 and 197: 4
Distance between 197 and 199: 2
Distance between 199 and 211: 12
Distance between 211 and 223: 12
Distance between 223 and 227: 4
Distance between 227 and 229: 2
Distance between 229 and 233: 4
Distance between 233 and 239: 6
Distance between 239 and 241: 2
```

```
Distance between 241 and 251: 10
Distance between 251 and 257: 6
Distance between 257 and 263: 6
Distance between 263 and 269: 6
Distance between 269 and 271: 2
```

30. Create a function that will add two positive numbers of indefinite size.The numbers are received as strings and the result should be also provided as string.

```
def add_strings(num1, num2):
    # Make sure num1 is the longer number
    if len(num1) < len(num2):
        num1, num2 = num2, num1

    # Pad the shorter number with leading zeros
    num2 = num2.zfill(len(num1))

    carry = 0
    result = []
    for i in range(len(num1)-1, -1, -1):
        digit_sum = int(num1[i]) + int(num2[i]) + carry
        digit = digit_sum % 10
        carry = digit_sum // 10
        result.append(str(digit))

    if carry:
        result.append(str(carry))

    result.reverse()
    return ''.join(result)

number1 = "12345678901234567890"
number2 = "98765432109876543210"
result = add_strings(number1, number2)
print(result)  # Output: "111111111011111111100"
```

```
111111111011111111100
```

31. Create a function that willreturn the number of words in a text

```
## function definition
def is_count(text):
  count = text.split()
  return len(count)

## driving code
text = input("Enter your text here : ")
output = is_count(text)
print(output)
```

```
Enter your text here : Hey
1
```

32. Create a function that will capitalize the first letter of each word in a text

```
## function definition

def is_capitalize(text):
  capital = text.title()
  return capital
## driving code
text = input("Enter your text here : ")
result = is_capitalize(text)
print(result)
```

```
Enter your text here : hello, this is me
Hello, This Is Me
```

33. Calculate the sum of numbers received in a comma delimited string

```
def calculate_sum(numbers_string):
    numbers = numbers_string.split(",")
    numbers = [int(num) for num in numbers]
    numbers_sum = sum(numbers)
    return numbers_sum
```

```
string = "1,2,3,4,5"
res = calculate_sum(string)
print(res)
```

```
15
```

34. Create a function thatreturns an array with words inside a text

```
## function definitions

def is_words(text):
  words = text.split()
  return words


## driving codes

text = input("Enter your text here : ")
result = is_words(text)
print(result)
```

```
Enter your text here : hello, this is python code
['hello,', 'this', 'is', 'python', 'code']
```

35. Create a function to convert a CSV text to a“bi-dimensional” array

```
def csv_to_array(csv_text):
    rows = csv_text.split("\n")
    array = [row.split(",") for row in rows]
    return array
csv_text = "1,2,3\n4,5,6\n7,8,9"
array_result = csv_to_array(csv_text)
print(array_result)
```

```
[['1', '2', '3'], ['4', '5', '6'], ['7', '8', '9']]
```

. Create a function that converts a string to an array of characters

```
## function defintion
def is_char(text):
  letter = [x for x in text]
  return letter

## driving code
text = "hello"
out = is_char(text)
print(out)
```

```
['h', 'e', 'l', 'l', 'o']
```

. Create afunction that will convert astring in an array containing the ASCII codes of each character

```
def string_to_ascii_array(string):
    ascii_array = [ord(char) for char in string]
    return ascii_array

## driving code
string = "hello, world"
my_output = string_to_ascii_array(string)
print(my_output)
```

```
[104, 101, 108, 108, 111, 44, 32, 119, 111, 114, 108, 100]
```

. Create a function that will convert an array containingASCII codes in a string Implement the Caesar cypher

```
## Ceaser cypher
def encrypt(text,shift):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if(char.isupper()):
            result += chr((ord(char)+ shift-65)%26 + 65)
        else:
            result += chr((ord(char)+ shift-97)%26 + 97)
    return result

text = "Hello"
shift = 3
print(encrypt(text,shift))
```

```
 Khoor
```

Double-click (or enter) to edit

Double-click (or enter) to edit

. Implement the Caesar cypher

```
def encrypt(text,shift):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if(char.isupper()):
            result += chr((ord(char)+ shift-65)%26 + 65)
        else:
            result += chr((ord(char)+ shift-97)%26 + 97)
    return result

text = "Tarun"
shift = 3
print(encrypt(text,shift))
```

```
 Wduxq
```

. Implement the bubble sort algorithm for an array of numbers

```
def bubble_sort(numbers):
    n = len(numbers)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if numbers[j] > numbers[j + 1]:
                numbers[j], numbers[j + 1] = numbers[j + 1], numbers[j]
    return numbers

numbers = [7, 3, 9, 2, 1, 5]
sorted_numbers = bubble_sort(numbers)
print(sorted_numbers)
```

```
[1, 2, 3, 5, 7, 9]
```

Create a function to calculate the distance between two points defined by their x,y coordinates

```
import math
## function definitions
def is_dist(dis1,dis2):
  distance = math.dist(dis1,dis2)
  return distance


## driving code

dis1 = [4,0]
dis2 = [6,6]
distance = is_dist(dis1,dis2)
print(distance)
```

```
6.324555320336759
```

. Create afunction that will return aBoolean value indicating iftwo circles defined by center coordinates and radius are intersecting

```
import math

def are_circles_intersecting(circle1, circle2):
    x1, y1, r1 = circle1
    x2, y2, r2 = circle2

    distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
    if distance <= r1 + r2:
        return True
    else:
        return False

circle1 = (0, 0, 5)
circle2 = (3, 3, 4)
result = are_circles_intersecting(circle1, circle2)
print(result)
```

```
True
```

. Create a function that will receive a bi-dimensional array as argument and a number and will extract as a unidimensional array the column specified by the number

```
def extract_column(arr, column_number):
    column = [row[column_number] for row in arr]
    return column

array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
column_number = 1
result = extract_column(array, column_number)
print(result)
```

```
[2, 5, 8]
```

Create afunction that will convert astring containing abinary number into a number

```
def binary_to_number(binary_string):
    decimal_number = int(binary_string, 2)
    return decimal_number

binary_string = "101010"
decimal_number = binary_to_number(binary_string)
print(decimal_number)  # Output: 42
```

```
42
```

Create a function to calculate the sum of all the numbers in a jagged array (contains numbers or other arrays of numbers on an unlimited number of levels)

```
def sum_jagged_array(arr):
    total_sum = 0
    for item in arr:
        if isinstance(item, list):
            total_sum += sum_jagged_array(item)
        else:
            total_sum += item
    return total_sum
jagged_array = [1, [2, 3], [4, [5, 6, [7, 8]]]]
array_sum = sum_jagged_array(jagged_array)
print(array_sum)
```

```
36
```

Shuffle an array of strings

```
import random
```

```
def shuffle_array(arr):
    random.shuffle(arr)

# Example usage
my_array = ['apple', 'banana', 'cherry', 'date']
shuffle_array(my_array)
print(my_array)
```

```
['banana', 'cherry', 'date', 'apple']
```

Find the frequency of letters inside a string.Return the result as an array of arrays.Each subarray has 2 elements:letter and number of occurrences.

```
def letter_frequency(string):
    frequency = {}

    # Count the frequency of each letter in the string
    for letter in string:
        if letter.isalpha():
            if letter.lower() in frequency:
                frequency[letter.lower()] += 1
            else:
                frequency[letter.lower()] = 1

    # Convert the frequency dictionary to an array of arrays
    result = [[letter, count] for letter, count in frequency.items()]

    return result

# Example usage
my_string = "Hello, world!"
frequency_array = letter_frequency(my_string)
print(frequency_array)
```

```
[['h', 1], ['e', 1], ['l', 3], ['o', 2], ['w', 1], ['r', 1], ['d', 1]]
```

Calculate Fibonacci(500) with high precision (all digits)

```
from decimal import Decimal, getcontext

def fibonacci(n):
    getcontext().prec = n + 2  # Set precision to accommodate the desired number of digits
    sqrt5 = Decimal(5).sqrt()
    phi = (1 + sqrt5) / 2

    fib = (phi ** n - (-phi) ** (-n)) / sqrt5
    return int(fib)

fib_500 = fibonacci(500)
print(fib_500)
```

```
139423224561697880139724382870407283950070256587697307264108962948325571622863290691557658876222521294125
```

Calculate 70! with high precision (all digits)

```
from decimal import Decimal, getcontext

def factorial(n):
    getcontext().prec = n + 2  # Set precision to accommodate the desired number of digits

    result = Decimal(1)
    for i in range(2, n + 1):
        result *= Decimal(i)

    return result

factorial_70 = factorial(70)
print(factorial_70)
```

```
1.197857166996989179607278372168909873645893814254642585755536286462800 95E+100
```