1. What is a function in Python and why is it used?

In [ ]:
```
function is a reusable block of code designed to perform a specific task.
it's help in organizing code, improving readability, and promoting reusability
A function in Python is defined using the def keyword
functions help you write code that is cleaner, easier to understand, and simpler to manage.
```

2. What is a function in Python and why is it used?

In [ ]:
```
function is a reusable block of code designed to perform a specific task.
it's help in organizing code, improving readability, and promoting reusability
A function in Python is defined using the def keyword
functions help you write code that is cleaner, easier to understand, and simpler to manage.
```

3. Explain the difference between a function definition and a function call.

In [ ]:
```
Function Definition : This is where you define the function using the def keyword followed
                      by the function name, parameters (optional), and a block of code that specifies what the
                      function does.

Function Call : This is where you actually invoke or execute the function using its name
                      and passing arguments (if required).
```

4. What are *args and **kwargs in Python functions and how are they used?

In [ ]:
```
*args: it used to pass a variable number of nonkeyword argumnts to function.
    This allow us to pass any number of arguments to function
    Used to handle a variable number of positional arguments. Collects them into a tuple.

**kwargs: The **kwargs parameter allows you to pass any number of keyword arguments as a
    dictionary to the function
    Used to handle a variable number of keyword arguments. Collects them into a dictionary
```

In [1]:
```
def print_numbers(*args):
    for number in args:
        print(number)

print_numbers(10,20,30,40,50,60)
```

```
10
20
30
40
50
60
```

In [2]:
```
def print_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_info(name="Jackson", age=27, city="Monaco")
```

```
name: Jackson
age: 27
city: Monaco
```

5. What is a higher-order function in Python? Give an example.

In [ ]:
```
Higher-order functions in Python are those that either take other functions as arguments or return functions as
They key concept in functional programming and are useful for creating more abstract and reusable code.
```

6. What is the purpose of the return statement in a function?

In [ ]:
```
The return statement used to terminates the execution of the function.
Once a return statement is executed, the function stops running, and control is returned to the caller.
No code after the return statement in the function will be executed
```

7. What are default arguments in Python functions?

In [ ]:
```
default argument means If a caller does not provide an argument for a parameter with a default value,
the function uses the default
```

8. Python Program to find the factorial of a number.

```
In [4]: def factorial(n):
            if n < 0:
                raise ValueError("Factorial is not defined for negative numbers")

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


        factorial(9)
```

```
Out[4]: 362880
```

```
In [5]: factorial(5)
```

```
Out[5]: 120
```

9. Python Program to replace whitespaces with an underscore and vice versa.

```
In [9]: def replace(text):
            x = text.replace(' ', '-')
            y = x.replace('_', ' ')
            z = y.replace('-','_')

            return z

        text = " Python and Data Science 9_9_9"
        replace(text)
```

```
Out[9]: '_Python_and_Data_Science_9 9 9'
```

10. Python Program to convert a date in yyyy-mm-dd format to dd-mm-yyyy.

```
In [15]: def convert(date_str):
             year, month, day = date_str.split('-')
             new_date_str = f"{day}-{month}-{year}"

             return new_date_str

         date = "2024-10-6"
         convert_date = convert(date)
         print(f"Original Date: {date}")
         print(f"Converted Date: {converted_date}")
```

```
Original Date: 2024-10-6
Converted Date: 6-10-2024
```

11. Python Program to check if the count of divisors is even or odd.

```
In [17]: def count_divisors(n):
             if n <= 0:
                 raise ValueError("Number must be greater than 0")

             count = 0
             for i in range(1, n + 1):
                 if n % i == 0:
                     count += 1
             return count

         def is_divisor_count_even_or_odd(n):
             div_count = count_divisors(n)
             if div_count % 2 == 0:
                 return 'Even'
             else:
                 return 'Odd'

         number = 36
         result = is_divisor_count_even_or_odd(number)
         print(f"The count of divisors of {number} is {result}.")
```

```
The count of divisors of 36 is Odd.
```

12. Python Program to convert a float decimal to an octal number.

In [18]:
```python
def float_decimal_to_octal(num):
    int_part = int(num)
    frac_part = num - int_part

    int_octal = oct(int_part)[2:]

    frac_octal = ''
    while frac_part != 0:
        frac_part *= 8
        frac_digit = int(frac_part)
        frac_octal += str(frac_digit)
        frac_part -= frac_digit

    if frac_octal:
        octal_num = int_octal + '.' + frac_octal
    else:
        octal_num = int_octal

    return octal_num

decimal_num = 99.9999
octal_num = float_decimal_to_octal(decimal_num)
print(f"The octal representation of {decimal_num} is:", octal_num)
```
```
The octal representation of 99.9999 is: 143.777745622164247
```

13. Python Program to copy odd lines of one file to another.

In [ ]:
```python

```

14. Python Program to find the largest prime factor of a number.

In [23]:
```python
def largest_prime_factor(n):
    factor = 2
    while factor * factor <= n:
        if n % factor == 0:
            n //= factor
        else:
            factor += 1
            return n

number = 90
largest_prime = largest_prime_factor(number)
print(f"The largest prime factor of {number} is:", largest_prime)
```
```
The largest prime factor of 90 is: 45
```

15. Python Program to find the product of unique prime factors of a number.

In [24]:
```python
def unique_prime_factors_product(n):
    if n <= 1:
        raise ValueError("The number must be greater than 1")

    product = 1
    factor = 2
    while factor * factor <= n:
        if n % factor == 0:
            product *= factor
            while n % factor == 0:
                n //= factor
        factor += 1

    if n > 1:
        product *= n

    return product


number = 90
print(f"The product of unique prime factors of {number} is {unique_prime_factors_product(number)}.")
```
```
The product of unique prime factors of 90 is 30.
```

16. Python Program to find the sum of odd factors of a number.

In [26]:
```python
def sum_of_odd_factors(n):
    if n <= 1:
        raise ValueError("The number must be greater than 1")

    total_sum = 0
    for i in range(1, n + 1, 2):
        if n % i == 0:
            total_sum += i

    return total_sum

number = 30
print(f"The sum of odd factors of {number} is {sum_of_odd_factors(number)}.")
```

```
The sum of odd factors of 30 is 24.
```

17. Python Program to find the common divisors of two numbers.

In [28]:
```
def find_common_divisors(x, y):
    common_divisors = []
    for i in range(1, min(x, y) + 1):
        if x % i == 0 and y % i == 0:
            common_divisors.append(i)
    return common_divisors

num1 = 30
num2 = 45
common_divisors = find_common_divisors(num1, num2)
print(f"The common divisors of {num1} and {num2} are:", common_divisors)
```

```
The common divisors of 30 and 45 are: [1, 3, 5, 15]
```

18. Python Program to find the minimum sum of factors of a number.

In [30]:
```
def sum_of_factors(n):
    if n <= 0:
        raise ValueError("The number must be greater than 0")

    minimum_sum = 0
    for i in range(1, int(n**0.5) + 1):
        if n % i == 0:
            minimum_sum += i
            if i != n // i:
                minimum_sum += n // i

    return minimum_sum

number = 18
print(f"The sum of factors of {number} is {sum_of_factors(number)}.")
```

```
The sum of factors of 18 is 39.
```

19. Python Program to find the difference between sums of odd and even digits.

In [41]:
```
def difference_odd_even_sum(number):
    odd_sum = 0
    even_sum = 0

    for digit in range(1,number+1):
        if digit % 2 == 0:
            even_sum += digit
        else:
            odd_sum += digit

    return odd_sum - even_sum

number = 321
print(f"The difference between the sums of odd and even digits is {difference_odd_even_sum(number)}.")
```

```
The difference between the sums of odd and even digits is 161.
```

20. Python Program to find the sum of even factors of a number.

In [42]:
```
def sum_of_even_factors(n):
    total_sum = 0
    for i in range(1, n + 1):
        if n % i == 0 and i % 2 == 0:
            total_sum += i
    return total_sum
```

```
number = 18
print(f"The sum of even factors of {number} is {sum_of_even_factors(number)}.")
```

```
The sum of even factors of 18 is 26.
```

21. Python Program to check if all digits of a number divide it.

In [46]:
```
def divide_number(n):
    for digit in str(n):
        if digit == '0' or n % int(digit) != 0:
            return False
    return True

number = 11
print(divide_number(number))
```

```
True
```

In [47]:
```
def divide_number(n):
    for digit in str(n):
        if digit == '0' or n % int(digit) != 0:
            return False
    return True

number = 27
print(divide_number(number))
```

```
False
```

22. Python Program to find all words starting with 'a' or 'e' in a given string.

In [50]:
```
def words_starting_with_a_or_e(text):
    words = text.split()
    result = []
    for word in words:
        if word.lower().startswith('a') or word.lower().startswith('e'):
            result.append(word)

    return result

text = "Python And Data Science and elephant."
print(words_starting_with_a_or_e(text))
```

```
['And', 'and', 'elephant.']
```

23. Python Program to abbreviate 'Road' as 'Rd.' in a given string.

In [51]:
```
def abbreviate(y):
    return y.replace("Road", "Rd.")


text = "Take a right turn on SM Road and go straight."
print(abbreviate(text))
```

```
Take a right turn on SM Rd. and go straight.
```

24. Python Program to check if the binary representation is a palindrome.

In [52]:
```
def is_binary_palindrome(n):
    binary_str = bin(n)[2:]
    return binary_str == binary_str[::-1]

number = 9
print(is_binary_palindrome(number))
```

```
True
```

In [53]:
```
def is_binary_palindrome(n):
    binary_str = bin(n)[2:]
    return binary_str == binary_str[::-1]

number = 11
print(is_binary_palindrome(number))
```

```
False
```

25. Python Program to find the number of elements with odd factors in a given range.

In [ ]:

26. Python Program to find the largest and smallest K-digit number divisible by X.

In [ ]:

27. Python Program to find the perimeter of a cylinder.

In [54]:
```
import math

def Cylinder(r):
    perimeter_cy = 2 * math.pi * r
    print(perimeter_cy)

Cylinder(9)
```

```
56.548667764616276
```

In [55]:
```
import math

def Cylinder(r):
    perimeter_cy = 2 * 3.14 * r
    print(perimeter_cy)

Cylinder(9)
```

```
56.52
```

28. Python Program to find the most occurring character and its count.

In [56]:
```
def most_occurring_char(string):
    char_count = {}
    for char in string:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1

    most_common_char = max(char_count, key=char_count.get)
    return most_common_char, char_count[most_common_char]

string = input("Enter a string: ")
char, count = most_occurring_char(string)
print(f"The most occurring character is '{char}' which appears {count} times.")
```

```
The most occurring character is 'n' which appears 3 times.
```

29. Python Program to check if a number is a prime number using a function.

In [57]:
```
def is_prime(number):
    if number <= 1:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True

number = int(input("Enter a number: "))
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

```
9 is not a prime number.
```

30. Python Program to merge two sorted lists using a function.

In [58]:
```
def sorting(l1, l2):
    a = sorted(l1)
    b = sorted(l2)
    merged_list = sorted(a + b)
    print(merged_list)

# Example usage
l1 = [10,20,30]
l2 = [9,18,27]
```

```
sorting(l1, l2)
```

```
[9, 10, 18, 20, 27, 30]
```

31. What is a recursive function in Python?

In [ ]:
```
A recursive function in Python is a function that calls itself in order to solve a problem.
```

32. What are the advantages of a recursive function?

In [ ]:
```
Recursion can make code easier to read and understand by breaking problems into smaller, manageable pieces.
Recursion allows for breaking down complex problems into simpler sub-problems.
This can simplify the problem-solving process, making it easier to implement
```

33. What are the disadvantages of a recursive function?

In [ ]:
```
Difficulty in Understanding: Recursive solutions can sometimes be harder to understand
for people not familiar with recursion or for very complex recursive patterns.
```

34. Python Program to find the factorial of a number using recursion

In [59]:
```python
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

number = int(input("Enter a number: "))
print(f"The factorial of {number} is {factorial(number)}.")
```

```
The factorial of 9 is 362880.
```

35. Python Program to find the nth Fibonacci number using recursion

In [60]:
```python
def fibonacci(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

n = int(input("Enter the position of Fibonacci number: "))
print(f"The {n}th Fibonacci number is {fibonacci(n)}.")
```

```
The 9th Fibonacci number is 34.
```

In [ ]: