**Task 1**

lista = [5,76,2,5]

exponent = int(input())  # User input for exponent

def funcpow(lista, exponent):

    result\_list = []  # Initialize an empty list to store results

    for num in lista:

        result = pow(num, exponent)  # Raise num to the power of exponent

        result\_list.append(result)  # Append the result to the result list

    return result\_list  # Return the list of powered numbers

powered\_list = funcpow(lista, exponent)

print(powered\_list)  # Print the list of powered numbers

**/ (Adi bölmə):** Həmişə **float (ondalıq ədəd) qaytarır**, hətta nəticə tam ədəd olsa belə.// **(Tam bölmə):** Həmişə **tam ədəd (integer) qaytarır**, qalıq hissəsini atır.

**Task 2**

def count\_character\_occurrences(input\_string, character):

    """

    Count how many times a given character appears in a user-provided string.

    Parameters:

    input\_string (str): The string in which to count occurrences.

    character (str): The character whose occurrences are to be counted.

    Returns:

    int: The count of the specified character in the input string.

    """

    # Ensure that the character is a single character

    if len(character) != 1:

        raise ValueError("Character must be a single character.")

    # Initialize the count

    count = 0

    # Loop through the input string to count occurrences

    for char in input\_string:

        if char == character:

            count += 1

    return count

# Example usage

input\_str = "hello world"

char\_to\_count = 'o'

result = count\_character\_occurrences(input\_str, char\_to\_count)

print(f"The character '{char\_to\_count}' appears {result} times in the string.")

Python-da startswith() metodu bir sətrin müəyyən bir prefiks (başlanğıc hissə) ilə başlayıb-başlamadığını yoxlayır və nəticə olaraq True (doğru) və ya False (yanlış) qaytarır. Bu metod həmçinin başlanğıc (start) və son (end) indeksləri qəbul edərək, axtarışı müəyyən bir hissədə məhdudlaşdıra bilər.

**Task 3**

def temizle\_bosluq(istifadeci\_girisi):

    """

    Bu funksiya, verilmiş sətirin əvvəlindəki və sonundakı boşluqları silir.

    Parametrlər:

    istifadeci\_girisi (str): İstifadəçinin daxil etdiyi sətir.

    Qaytarır:

    str: Boşluqları silinmiş sətir.

    """

    temizlenmis\_metn = istifadeci\_girisi.strip()  # strip() metodu ilə əvvəl və sondakı boşluqları sil

    print(f"Orijinal Mətn: '{istifadeci\_girisi}'")  # Orijinal mətn

    print(f"Təmizlənmiş Mətn: '{temizlenmis\_metn}'")  # Təmizlənmiş mətn

    return temizlenmis\_metn

# İstifadəçidən giriş al və funksiyanı işə sal

istifadeci\_girisi = input("Bir mətn daxil edin: ")

temizlenmis\_netice = temizle\_bosluq(istifadeci\_girisi)

strip() metodu bir sətirin (string) **başındakı və sonundakı boşluqları (whitespace) silir**, amma mətnin içərisindəki boşluqlara toxunmur. split() metodu bir sətiri **boşluqlara və ya verilən ayrıcılara görə hissələrə ayırır** və nəticəni bir siyahı (list) şəklində qaytarır.

**Task 4**

def replace\_substring(original, to\_replace, new\_substring):

    """

    Bu funksiya orijinal sətrdə olan bütün 'to\_replace' alt sətrini

    'new\_substring' ilə əvəz edir.

    """

    if to\_replace not in original:

        return "Əvəz ediləcək alt sətr mövcud deyil."

    return original.replace(to\_replace, new\_substring)

# Test üçün nümunə

original\_string = "Salam dünya! Dünya gözəldir."

substring\_to\_replace = "Dünya"

new\_substring = "Yer"

result = replace\_substring(original\_string, substring\_to\_replace, new\_substring)

print(result)

Əgər əvəz ediləcək alt sətr orijinal sətirdə mövcud deyilsə, heç bir dəyişiklik edilməz və proqram ya ilkin sətiri olduğu kimi qaytarar, ya da xüsusi bir xəbərdarlıq mesajı verə bilər.

**Task 5**

def calculate\_absolute\_value(number):

    """

    This function takes an integer as an input and returns its absolute value.

    Parameters:

    number (int): The integer whose absolute value is to be calculated.

    Returns:

    int: The absolute value of the input integer.

    """

    if number < 0:

        return -number

    else:

        return number

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    user\_input = int(input("Enter an integer: "))

    absolute\_value = calculate\_absolute\_value(user\_input)

    print(f"The absolute value of {user\_input} is {absolute\_value}.")

Python-da len() funksiyası bir sətrin uzunluğunu qaytarır.Əgər sətr boşdursa (""), len() 0 qaytarır, çünki heç bir simvol yoxdur.

**Task 6**

def calculate\_string\_length(user\_string):

    """

    This function calculates the length of a user-provided string.

    Parameters:

    user\_string (str): The string whose length is to be calculated.

    Returns:

    int: The length of the provided string.

    """

    # Check if the input is valid (it should be a string)

    if not isinstance(user\_string, str):

        raise ValueError("Input must be a string")

    # Calculate the length of the string using the built-in len() function

    length = len(user\_string)

    return length  # Returning the calculated length

# Testing the function with different cases

print(calculate\_string\_length("Hello, world!"))  # Expected output: 13

print(calculate\_string\_length(""))  # Expected output: 0

print(calculate\_string\_length("Python programming"))  # Expected output: 20

Python-da abs() funksiyası bir ədədin modulunu qaytarır, yəni həmişə müsbət dəyər verir.

**Task 7**

def check\_string\_start(string, prefix):

    """

    Checks if the provided string starts with the specified prefix.

    Parameters:

    string (str): The string to check.

    prefix (str): The prefix to look for.

    Returns:

    bool: True if the string starts with the prefix, False otherwise.

    """

    # Handle the case where the prefix is an empty string

    if prefix == "":

        return True

    return string.startswith(prefix)

# Example usage

result = check\_string\_start("Hello, World!", "Hello")

print(result)  # Output: True

endswith() metodu bir mətnin müəyyən bir sonluqla bitib-bitmədiyini yoxlamağa kömək edir. Məsələn, fayl adlarını .txt və ya .jpg kimi uzantılara görə filtrələmək üçün istifadə oluna bilər. Bu metod bir fayl adının verilmiş uzantı ilə bitib-bitmədiyini yoxlayır və uyğun gələnləri seçməyə imkan verir. Əgər bir neçə uzantı yoxlanmalıdırsa, onları siyahı və ya tuple şəklində verərək bir neçə fərqli fayl növünü eyni anda filtrələmək mümkündür. 🚀

**Task 8**

def check\_string\_end(string, suffix):

    """

    Check if the provided string ends with the specified suffix.

    Parameters:

    string (str): The main string to check.

    suffix (str): The suffix to check against.

    Returns:

    bool: True if string ends with suffix, otherwise False.

    """

    return string.endswith(suffix)

# Example usage

result = check\_string\_end("example.txt", ".txt")

print(result)  # Output should be True

Əgər startswith() metoduna boş string ("") ötürsəniz, hər bir mətn üçün nəticə True olacaq. Çünki hər bir string boş string ilə başlayır. 🚀

**Task 9**

def split\_sentence(sentence):

    """

    Splits the given sentence into words and returns a list of words.

    Args:

    sentence (str): The sentence to be split.

    Returns:

    list: A list of words in the sentence.

    """

    # Using the split method to handle multiple spaces

    words = sentence.split()

    return words

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    input\_sentence = "This is a sample sentence with  multiple spaces."

    result = split\_sentence(input\_sentence)

    print(result)

Separator verilmezse [] bos siyahi qaytarir.Separator verilerse [“ ”] bir elementli siyahi qaytarir.(Separator setiri hisselere ayirmaq ucun istifade olunan simvol ve simvollar ardicilligidir)

**Task 10**

def get\_ascii\_value(character):

    """

    This function accepts a single character as input

    and returns its ASCII value. If the input is not

    a single character, it raises a ValueError.

    Parameters:

    character (str): A single character to get its ASCII value.

    Returns:

    int: ASCII value of the input character.

    Raises:

    ValueError: If the input is not a single character.

    """

    # Check if the input is a single character

    if len(character) != 1:

        raise ValueError("Input must be a single character.")

    # Return the ASCII value of the character

    return ord(character)

# Example usage

try:

    char = input("Enter a single character: ")

    ascii\_value = get\_ascii\_value(char)

    print(f"The ASCII value of '{char}' is {ascii\_value}.")

except ValueError as e:

    print(e)

// tam hisseni qaytarir.Kesr hisse nezere alinmir.% qaliq hisseni qaytarir.

**Task 11**

def calculate\_remainder(a, b):

    """

    This function takes two integers as input and returns the remainder

    when the first integer is divided by the second. It handles the case

    where the second integer is zero to avoid division by zero.

    Parameters:

    a (int): The first integer.

    b (int): The second integer.

    Returns:

    int: The remainder of a divided by b, or None if b is 0.

    """

    # Check if the second integer is zero

    if b == 0:

        print("Error: Division by zero is not allowed.")

        return None  # Return None to indicate an error

    else:

        return a % b  # Return the remainder of a divided by b

# Example usage

result = calculate\_remainder(10, 3)

if result is not None:

    print(f"The remainder of 10 divided by 3 is: {result}")

result = calculate\_remainder(10, 0)

if result is not None:

    print(f"The remainder of 10 divided by 0 is: {result}")

ord('a') 97 qaytarır, ord('A') isə 65 qaytarır. Fərq onların ASCII dəyərlərindədir; kiçik a böyük A-dan 32 vahid böyükdür.

**Task 12**

def print\_pyramid():

    # Loop through numbers 1 to 10

    for i in range(1, 11):

        # Print spaces for pyramid shape

        print(' ' \* (10 - i), end='')  # Adjusting spaces for alignment

        # Print the number followed by asterisks

        for j in range(1, i + 1):

            print(f"{j}\*", end=' ')

        # Move to the next line after each row

        print()

# Call the function to display the pyramid

print\_pyramid()

Read(): Bütün faylı bir string kimi oxuyur. Readline(): Fayldan bir sətri (sətir sonu ilə birlikdə) oxuyur.Readlines(): Bütün sətrləri siyahı (list) şəklində qaytarır.

**Task 13**

def read\_file\_line\_by\_line(file\_path):

    """

    Reads a text file and prints its contents line by line.

    Parameters:

    file\_path (str): The path to the text file to be read.

    Returns:

    None: This function prints the lines to the console.

    """

    try:

        # Open the file in read mode

        with open(file\_path, 'r') as file:

            # Read and print each line

            for line in file:

                print(line.strip())  # Strip removes leading/trailing whitespace

    except FileNotFoundError:

        print(f"Error: The file '{file\_path}' does not exist.")

    except IOError:

        print("Error: An I/O error occurred.")

# Example usage:

# read\_file\_line\_by\_line('example.txt')

range(5) 0-dan 4-ə qədər (0, 1, 2, 3, 4) ədədləri qaytarır. Addım (step) dəyərini daxil etmək üçün range(start, stop, step) formatından istifadə olunur. Məsələn, range(0, 10, 2) nəticəsində 0, 2, 4, 6, 8 qaytarılır

**Task 14**

def factorial(n):

    """

    Calculate the factorial of a positive integer n using recursion.

    Parameters:

    n (int): A positive integer for which to calculate the factorial.

    Returns:

    int: The factorial of n.

    The factorial of n is defined as:

        n! = n \* (n - 1) \* (n - 2) \* ... \* 1

    The base case occurs when n is equal to 1, at which point we return 1.

    For other values, we reduce n until we reach the base case.

    """

    # Check if the input is a positive integer

    if n < 0:

        raise ValueError("Input must be a positive integer.")

    # Base case: factorial of 1 is 1

    if n == 1 or n == 0:

        return 1

    # Recursive case: n! = n \* (n-1)!

    return n \* factorial(n - 1)

# Example usage

try:

    num = 5

    print(f"The factorial of {num} is {factorial(num)}")

except ValueError as e:

    print(e)

pass Python-da boş bloklar yaratmaq üçün istifadə olunur. Əmr kimi heç nə etmədən kodun işləməsinə imkan verir. Məsələn, funksiyalar və ya döngülərdə yer tutucu kimi istifadə edilə bilər

**Task 15**

def square\_numbers(n):

    """

    Generate a list of squares of numbers from 1 to n.

    Parameters:

    n (int): A positive integer.

    Returns:

    list: A list of squares of numbers from 1 to n.

    """

    return [i\*\*2 for i in range(1, n + 1)]

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    n = int(input("Enter a positive integer: "))

    if n > 0:

        squares = square\_numbers(n)

        print(f"Squares from 1 to {n}: {squares}")

    else:

        print("Please enter a positive integer.")

Python-da **format()** funksiyası, string içindəki **{}** yer tutucularını doldurmaq üçün istifadə olunur. Məsələn, **"Mənim adım {} və mən {} yaşındayam.".format("Aydin", 30)** ifadəsi, **ad** və **yas** dəyərlərini müvafiq olaraq yer tutucularla əvəz edir.

**Task 16**

def is\_prime(number):

    """

    Function to check if a number is prime.

    A prime number is only divisible by 1 and itself.

    This function accepts an integer as input, checks if it is

    less than or equal to 100, and returns True if it is prime,

    otherwise returns False.

    Parameters:

    number (int): The number to check (should be not more than 100).

    Returns:

    bool: True if number is prime, otherwise False.

    """

    if number <= 1 or number > 100:

        return False

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

        if number % i == 0:  # Check if number is divisible by any number in the loop

            return False

    return True  # Return True if no divisors were found

# Example usage:

num = 29

if is\_prime(num):

    print(f"{num} is a prime number.")

else:

    print(f"{num} is not a prime number.")

**string[::-1]** dilimləmə ifadəsi, stringi sonundan başlanğıcına doğru iterasiya edərək tərsinə çevirir.

**Task 17**

[::-1] ifadəsi bir **string-i tərsinə çevirmək üçün qısa və effektiv üsuldur**.

def generate\_fibonacci(n):

    """

    Generate the first n numbers in the Fibonacci sequence.

    Parameters:

    n (int): The count of Fibonacci numbers to generate.

    Returns:

    list: A list containing the first n Fibonacci numbers.

    """

    # Base cases

    if n <= 0:

        return []

    elif n == 1:

        return [0]

    elif n == 2:

        return [0, 1]

    # Initialize the Fibonacci sequence list

    fibonacci\_sequence = [0, 1]

    # Generate Fibonacci sequence using a loop

    for i in range(2, n):

        next\_number = fibonacci\_sequence[i - 1] + fibonacci\_sequence[i - 2]

        fibonacci\_sequence.append(next\_number)

    return fibonacci\_sequence

# Example usage

n = 10  # Number of Fibonacci numbers to generate

print(generate\_fibonacci(n))

**Task 18**

def count\_vowels(input\_string):

    """

    Count the number of vowels in the given string.

    Parameters:

    input\_string (str): The string in which to count vowels.

    Returns:

    int: The number of vowels found in the string.

    """

    # Define the vowels

    vowels = "aeiouAEIOU"

    count = 0

    # Use a loop to iterate through the string

    for char in input\_string:

        # Check if the character is a vowel

        if char in vowels:

            count += 1  # Increment count if a vowel is found

    return count

# Example usage

sample\_string = "Hello World!"

vowel\_count = count\_vowels(sample\_string)

print(f"The number of vowels in '{sample\_string}' is: {vowel\_count}")

Python-da mövqeyə əsaslanan arqumentlər, bir funksiyaya verilən dəyərlərin, funksiyanın parametr siyahısında olan sıralama ilə uyğun gəlməsidir. Yəni, arqumentlər funksiyaya çağırıldıqda, onların sırası, funksiyanın parametrlərinə uyğun olaraq təyin edilir.

**Task 19**

Python-da modul, müəyyən funksiyaları, sinifləri və dəyişənləri ehtiva edən bir fayldır. Modulu kodda çağırmaq üçün **import** ifadəsini istifadə edirsiniz; məsələn, **import math** yazaraq **math** modulunu çağırıb, onun funksiyalarını istifadə edə bilərsiniz.

**Task 20**

def find\_maximum(numbers):

    """

    This function returns the maximum element from a given list of numbers.

    Parameters:

    numbers (list): A list of numbers

    Returns:

    int or float or None: The maximum number in the list, or None if the list is empty.

    """

    # Check if the list is empty

    if not numbers:  # This checks if the list is empty

        return None  # Return None for an empty list

    # Use the built-in max() function to find the maximum

    maximum = max(numbers)

    return maximum

Python-da **max()** funksiyası boş bir listdə çağırıldıqda **ValueError** yaranır.

Xəta mesajı adətən "max() arg is an empty sequence" (max() arg boş ardıcıllıqdır) olur.

**Task 21**

def sum\_elements(lst):

    """

    This function takes a list of numbers as input and returns the sum of all elements.

    If the list is empty, it returns 0.

    Parameters:

    lst (list): A list of numeric values.

    Returns:

    int/float: The sum of all elements in the list.

    """

    return sum(lst) if lst else 0

# Example usage:

numbers = [1, 2, 3, 4, 5]

print(sum\_elements(numbers))  # Output: 15

empty\_list = []

print(sum\_elements(empty\_list))  # Output: 0

**sorted()**Yeni bir **sıralanmış siyahı qaytarır** və orijinal siyahını dəyişdirmir

**list.sort()**Siyahını **yerində (in-place) sıralayır** və heç bir şey qaytarmır (None qaytarır).

**Task 22**

def sum\_of\_list(numbers):

    """

    Bu fonksiyon, verilen bir listedeki sayıların toplamını döndürür.

    Parametre:

    numbers (list): Toplanacak sayıları içeren liste.

    Dönüş:

    int/float: Listenin elemanlarının toplamı.

    """

    return sum(numbers)

# Örnek kullanım

numbers = [5, 2, 9, 1, 5, 6]

total = sum\_of\_list(numbers)

print("Listenin Toplamı:", total)

Bir siyahıdakı ədədlərin cəmini tapmaq üçün həmin siyahının bütün elementlərini toplayırıq. Python-da bunu **sum()** funksiyası ilə asanlıqla hesablamaq mümkündür.

**Task 23**

Palindrom yoxlamaq üçün funksiyanı elə dəyişdirmək lazımdır ki, **boşluqları və durğu işarələrini nəzərə almasın**. Bunun üçün:

1. **lower() istifadə edərək bütün hərfləri kiçik hərfə çeviririk**.
2. **isalnum() ilə yalnız hərf və rəqəmləri saxlayırıq**, qalan simvolları silirik.
3. **Təmizlənmiş mətnin tərsini ([::-1]) yoxlayırıq**.

def find\_even\_numbers(input\_list):

    """

    This function returns a new list containing only the even numbers

    from the provided input list.

    Parameters:

    input\_list (list): A list of integers.

    Returns:

    list: A list containing the even integers.

    """

    # Using list comprehension to filter even numbers

    even\_numbers = [num for num in input\_list if num % 2 == 0]

    return even\_numbers

# Example usage:

if \_\_name\_\_ == "\_\_main\_\_":

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

    print("Even numbers:", find\_even\_numbers(numbers))

**Task 24**

def is\_palindrome(s):

    """

    Check if the input string s is a palindrome.

    Args:

    s (str): The string to check.

    Returns:

    bool: True if s is a palindrome, otherwise False.

    """

    # Convert string to lowercase to make the check case-insensitive

    s = s.lower()

    # Remove spaces (optional: modify according to requirements)

    s = s.replace(" ", "")

    # Check if string is equal to its reverse

    return s == s[::-1]

* **x % 2 == 0** ifadəsi **x** dəyişənin cütün olub olmadığını yoxlayır. Əgər **x** cütdürsə, nəticə True olacaq, əks halda False.

**Task 25**

def generate\_multiplication\_table(n):

    """

    This function generates and prints the multiplication table for a given integer n.

    :param n: The integer for which the multiplication table will be generated.

    """

    # Iterate through numbers from 1 to 10

    for i in range(1, 11):

        result = n \* i  # Calculate the multiplication result

        # Print the formatted output

        print(f"{n} x {i} = {result}")

# Example usage

number = int(input("Enter an integer: "))

generate\_multiplication\_table(number)

**Parameters Required:**

1. **old**: The substring you want to replace.
2. **new**: The substring you want to replace it with.
3. **count** (optional): The number of occurrences to replace. If not specified, all occurrences will be replaced.

**Task 26**

**// (tam bölmə - floor division)** → Qismətin yalnız **tam hissəsini qaytarır** (ondalıq hissəni atır).

**% (qalıq tapma - modulus)** → Bölmə əməliyyatının **qalıq hissəsini qaytarır**.

def count\_word\_frequency(input\_string):

    # Convert the string to lowercase to ensure uniformity

    input\_string = input\_string.lower()

    # Remove punctuation using str.replace

    for char in '.,!?;':

        input\_string = input\_string.replace(char, '')

    # Split the string into words

    words = input\_string.split()

    # Create a dictionary to store word counts

    word\_count = {}

    # Count the frequency of each word

    for word in words:

        if word in word\_count:

            word\_count[word] += 1

        else:

            word\_count[word] = 1

    return word\_count

# Example usage

text = "Hello world! Hello everyone."

result = count\_word\_frequency(text)

print(result)  # Output: {'hello': 2, 'world': 1, 'everyone': 1}

**Task 27**

**// (tam bölmə - floor division)** → Qismətin yalnız **tam hissəsini qaytarır** (ondalıq hissəni atır).

**% (qalıq tapma - modulus)** → Bölmə əməliyyatının **qalıq hissəsini qaytarır**.

def sum\_of\_digits(n):

    """

    Calculate the sum of the digits of a given positive integer.

    Parameters:

    n (int): A positive integer input.

    Returns:

    int: The sum of the digits of the input integer.

    """

    # Initialize the sum to 0

    total\_sum = 0

    # Use a loop to process each digit

    while n > 0:

        # Extract the last digit using the modulo operator

        digit = n % 10

        # Add the digit to the total sum

        total\_sum += digit

        # Remove the last digit using integer division

        n //= 10

    return total\_sum

# Example usage

positive\_integer = 12345

result = sum\_of\_digits(positive\_integer)

print(f"The sum of the digits of {positive\_integer} is: {result}")

**Task 28**

def find\_unique\_elements(input\_list):

    """

    This function returns a list of unique elements from the given list.

    Parameters:

    input\_list (list): The input list from which to find unique elements.

    Returns:

    list: A list containing only unique elements, preserving the original order.

    """

    unique\_elements = []  # List to store unique elements

    seen = set()  # Set to keep track of seen elements

    for element in input\_list:

        # Only add the element if it hasn't been seen before

        if element not in seen:

            unique\_elements.append(element)  # Append to the unique elements list

            seen.add(element)  # Mark the element as seen

    return unique\_elements

# Example usage

example\_list = [1, 2, 2, 3, 4, 4, 5]

print(find\_unique\_elements(example\_list))  # Output: [1, 2, 3, 4, 5]

pop listdeki son elementi silir del indekse gore silir

**Task 29**

def merge\_and\_sort\_lists(list1, list2):

    """

    This function takes two lists of numbers as input,

    merges them into a single list, and returns the sorted list in ascending order.

    Parameters:

    list1 (list): The first list of numbers.

    list2 (list): The second list of numbers.

    Returns:

    list: A sorted list containing all elements from both lists.

    """

    # Merge the two lists

    merged\_list = list1 + list2  # Alternatively, you can use list1.extend(list2)

    # Sort the merged list in ascending order

    sorted\_list = sorted(merged\_list)

    return sorted\_list

# Example usage

list1 = [5, 1, 9]

list2 = [8, 3, 2]

result = merge\_and\_sort\_lists(list1, list2)

print(result)  # Output: [1, 2, 3, 5, 8, 9]

**intersection ortaq elementleri verir**

**Task 30**

import math

def calculate\_area\_of\_circle(radius):

    """

    This function calculates the area of a circle given its radius.

    Parameters:

    radius (float): The radius of the circle. Must be a non-negative number.

    Returns:

    float: The area of the circle if the radius is valid.

    str: An error message if the radius is invalid.

    """

    # Handle invalid input

    if radius < 0:

        return "Error: Radius cannot be negative."

    # Calculate area using the formula: area = π \* r^2

    area = math.pi \* (radius \*\* 2)

    return area

# Example usage

radius = 5

result = calculate\_area\_of\_circle(radius)

if isinstance(result, str):

    print(result)  # Print error message if there is one

else:

    print(f"The area of the circle with radius {radius} is: {result:.2f}")

* **math.pi** daima daha dəqiq bir pi dəyəri təqdim edir (3.141592653589793).
* 3.14, pi-nin yuvarlanmış versiyasıdır və bəzi hesablamalarda dəqiqlik itirə bilər.

**Task31**

def find\_common\_elements(list1, list2):

    """

    This function takes two lists as input and returns a list of common elements between them without duplicates.

    Parameters:

    list1 (list): The first list.

    list2 (list): The second list.

    Returns:

    list: A list containing the common elements from list1 and list2.

    """

    # Converting lists to sets to eliminate duplicates and using intersection to find common elements.

    common\_elements = set(list1) & set(list2)

    # Converting the set back to a list before returning.

    return list(common\_elements)

# Example usage:

list\_a = [1, 2, 3, 4, 5]

list\_b = [4, 5, 6, 7, 8]

result = find\_common\_elements(list\_a, list\_b)

print("Common elements:", result)

**[::-1]** is a concise and effective way to reverse sequences in Python.

**Task32**

def reverse\_string(input\_string):

    """

    Function to reverse a given string and return the reversed result.

    Features:

    - Accepts a string as input.

    - Returns the string in reverse order.

    Args:

    input\_string (str): The string to be reversed.

    Returns:

    str: The reversed string or an error message if the input is empty.

    """

    # Validate input

    if not input\_string:

        return "Input cannot be empty."

    # Return the reversed string using slicing

    return input\_string[::-1]

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    test\_string = "Hello, World!"

    print(reverse\_string(test\_string))  # Output: !dlroW ,olleH

**[::-1]** is a concise and effective way to reverse sequences in Python.

**Task33**

* Exceptions, proqramın icrası zamanı qarşılaşdığı gözlənilməz vəziyyətlərdir.
* Python-da exceptions **try** və **except** blokları istifadə edərək idarə olunur.
* **try bloku:** Burada bölmə əməliyyatı baş verir.
* **except bloku:** Hər hansı bir istisna baş verdikdə (məsələn, **ZeroDivisionError** və **TypeError**), müvafiq mesajı qaytarır.
* **else bloku:** Heç bir istisna baş vermədiyi zaman, bölmə nəticəsini qaytarır.

def factorial\_iterative(n):

    """

    Calculate the factorial of a non-negative integer using an iterative approach.

    Args:

    n (int): A non-negative integer for which to calculate the factorial.

    Returns:

    int: Factorial of the input integer n.

    Raises:

    ValueError: If n is a negative integer.

    """

    if n < 0:

        raise ValueError("Input must be a non-negative integer.")

    result = 1

    for i in range(2, n + 1):

        result \*= i

    return result

def factorial\_recursive(n):

    """

    Calculate the factorial of a non-negative integer using a recursive approach.

    Args:

    n (int): A non-negative integer for which to calculate the factorial.

    Returns:

    int: Factorial of the input integer n.

    Raises:

    ValueError: If n is a negative integer.

    """

    if n < 0:

        raise ValueError("Input must be a non-negative integer.")

    if n == 0 or n == 1:

        return 1

    return n \* factorial\_recursive(n - 1)

# Example usage

try:

    num = 5

    print(f"Factorial of {num} (Iterative): {factorial\_iterative(num)}")

    print(f"Factorial of {num} (Recursive): {factorial\_recursive(num)}")

except ValueError as e:

    print(e)

def factorial\_iterative(n):

    """

    Calculate the factorial of a non-negative integer using an iterative approach.

    Args:

    n (int): A non-negative integer for which to calculate the factorial.

    Returns:

    int: Factorial of the input integer n.

    Raises:

    ValueError: If n is a negative integer.

    """

    if n < 0:

        raise ValueError("Input must be a non-negative integer.")

    result = 1

    for i in range(2, n + 1):

        result \*= i

    return result

def factorial\_recursive(n):

    """

    Calculate the factorial of a non-negative integer using a recursive approach.

    Args:

    n (int): A non-negative integer for which to calculate the factorial.

    Returns:

    int: Factorial of the input integer n.

    Raises:

    ValueError: If n is a negative integer.

    """

    if n < 0:

        raise ValueError("Input must be a non-negative integer.")

    if n == 0 or n == 1:

        return 1

    return n \* factorial\_recursive(n - 1)

# Example usage

try:

    num = 5

    print(f"Factorial of {num} (Iterative): {factorial\_iterative(num)}")

    print(f"Factorial of {num} (Recursive): {factorial\_recursive(num)}")

except ValueError as e:

    print(e)

**Task34**

def celsius\_to\_fahrenheit(celsius):

    """

    Convert Celsius to Fahrenheit.

    Parameters:

        celsius (float): The temperature in Celsius.

    Returns:

        float: The equivalent temperature in Fahrenheit.

    """

    # Validate input

    if not isinstance(celsius, (int, float)):

        raise ValueError("The input must be a number.")

    # Calculate Fahrenheit using the formula F = (C \* 9/5) + 32

    fahrenheit = (celsius \* 9 / 5) + 32

    return fahrenheit

# Example usage

try:

    celsius\_temp = float(input("Enter the temperature in Celsius: "))

    fahrenheit\_temp = celsius\_to\_fahrenheit(celsius\_temp)

    print(f"{celsius\_temp} degrees Celsius is equal to {fahrenheit\_temp} degrees Fahrenheit.")

except ValueError as e:

    print(f"Error: {e}")

**c yerine – 40qoyduqda ele -40 alinacaq.Celsius fahrenheite beraber olacaq**

**Task35**

def remove\_duplicates(input\_list):

    """

    Remove duplicates from a list and return a new list of unique elements.

    Parameters:

    input\_list (list): The list from which duplicates need to be removed.

    Returns:

    list: A new list containing unique elements from the input list.

    """

    # Set to track seen elements

    seen = set()

    # List to store the result

    unique\_list = []

    for item in input\_list:

        # Only add elements if they haven't been added before

        if item not in seen:

            seen.add(item)  # Mark this item as seen

            unique\_list.append(item)  # Add to the result list

    return unique\_list

# Example usage

example\_list = [1, 2, 3, 2, 1, 4, 5, 3]

result = remove\_duplicates(example\_list)

print(result)  # Output: [1, 2, 3, 4, 5]

 **r+" (oxuma və yazma rejimi)** – Mövcud faylı açır və həm oxumağa, həm də yazmağa imkan verir. Əgər fayl mövcud deyilsə, **səhv baş verir**. Faylın əvvəlində yazmağa başlasa da, əvvəlki məzmunu **silmədən** üzərinə yaza bilər.

 **"w" (yazma rejimi)** – Mövcud faylı açır və **tamamilə silir**, sonra yeni məzmun yazmağa başlayır. Əgər fayl mövcud deyilsə, **yeni fayl yaradır**.

**Task 36**

def longest\_word\_length(sentence):

    """

    This function takes a sentence as input and returns the length of the longest word.

    Parameters:

    sentence (str): A string containing multiple words.

    Returns:

    int: The length of the longest word in the sentence.

    """

    # Split the sentence into words

    words = sentence.split()

    # Initialize the maximum length to 0

    max\_length = 0

    # Loop through each word and check its length

    for word in words:

        # Update max\_length if the current word is longer

        max\_length = max(max\_length, len(word))

    return max\_length

# Example usage:

sentence = "Finding the Length of the Longest Word"

print("The length of the longest word is:", longest\_word\_length(sentence))

**ASCII cədvəli** (American Standard Code for Information Interchange) – **hərfləri, rəqəmləri və simvolları** kompüterin başa düşə biləcəyi **rəqəmlərə (0-127 arası) çevirən kod sistemidir**.

### ****Üstünlükləri:****

✅ **Standartlaşdırılmışdır** – Bütün sistemlərdə eyni qaydada işləyir.  
✅ **Yüngüldür və sadədir** – Cəmi **128 simvol** ehtiva etdiyi üçün yaddaş və emal baxımından sürətlidir.  
✅ **Müxtəlif proqramlaşdırma dillərində istifadə edilə bilər** – Hərfi və simvolu rəqəmə çevirmək üçün

**Task 37**

**upper() və lower() funksiyaları** Python-da mətnləri **böyük və ya kiçik hərflərə çevirmək üçün istifadə olunur**.

### ****İstifadə məqsədləri:****✅ **Mətnləri standartlaşdırmaq** (Məsələn, istifadəçi girişlərini müqayisə edərkən).✅ **Hərf ölçüsündən asılı olmayan axtarış aparmaq**. ✅ **Verilənləri formatlaşdırmaq** (Məsələn, başlıqları avtomatik böyük hərflə yazmaq).

def count\_vowels(input\_string):

    """

    This function takes a string as input and returns the count of vowels (a, e, i, o, u).

    Parameters:

    input\_string (str): The string in which to count vowels.

    Returns:

    int: The number of vowels in the input string.

    """

    # Define a set of vowels

    vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}

    # Initialize a counter for vowels

    vowel\_count = 0

    # Loop through each character in the input string

    for char in input\_string:

        # Check if the character is a vowel

        if char in vowels:

            vowel\_count += 1

    return vowel\_count

# Example usage

test\_string = "Hello World"

print(f"The number of vowels in '{test\_string}' is: {count\_vowels(test\_string)}")

### Task 38

def find\_largest\_element(numbers):

    """

    This function accepts a list of numbers as input and returns the largest element.

    Parameters:

    numbers (list): A list of numerical values.

    Returns:

    int/float: The largest number in the provided list.

    """

    # Check if the input list is empty

    if not numbers:

        return None  # Return None if the list is empty

    # Initialize maximum with the first element of the list

    max\_element = numbers[0]

    # Loop through all the numbers in the list

    for number in numbers:

        # Update max\_element if the current number is greater

        if number > max\_element:

            max\_element = number

    return max\_element  # Return the largest element found

# Example usage

example\_list = [3, 5, 1, 8, 2]

largest = find\_largest\_element(example\_list)

print(f"The largest element in the list is: {largest}")

Python-da **flush() funksiyası** fayl üzərində əməliyyatlardan sonra **yaddaşdakı məlumatları dərhal fayla yazmaq üçün istifadə olunur**.

### ****İstifadə məqsədi:****✅ **Məlumat itkisini qarşısını almaq** – Əgər proqram qəfil dayanarsa, flush() çağırılmamış məlumatlar itə bilər.✅ **Dərhal fayla yazmaq** – Standart olaraq, Python bəzi məlumatları **buffer-də (müvəqqəti yaddaşda) saxlayır**, flush() isə bunu dərhal fayla ötürür.

### Task 39

def sum\_of\_digits(num):

    """

    Function to calculate and return the sum of the digits of a given positive integer.

    Parameters:

    num (int): A positive integer input. If a negative integer is passed, it raises a ValueError.

    Returns:

    int: The sum of the digits of the input integer.

    Raises:

    ValueError: If the input number is negative.

    """

    # Check if the number is negative

    if num < 0:

        raise ValueError("Please enter a positive integer.")

    total\_sum = 0

    # Loop to calculate the sum of digits

    while num > 0:

        digit = num % 10  # Extract last digit

        total\_sum += digit  # Add digit to total sum

        num //= 10  # Remove last digit from number

    return total\_sum

# Example usage

try:

    result = sum\_of\_digits(12345)

    print(f"The sum of the digits is: {result}")  # Output: The sum of the digits is: 15

except ValueError as e:

    print(e)

Python-da **seek(5) funksiyası**, faylda **kursoru 5-ci bayta (simvola) köçürmək üçün istifadə olunur**.

### ****Necə işləyir?****✅ **Faylda müəyyən mövqeyə keçməyə imkan verir**. ✅ **Yazı və ya oxuma əməliyyatlarında faydalıdır** (məsələn, faylın ortasından məlumat əlavə etmək üçün).

### Task 40

def is\_palindrome(text):

    # Böyük-küçük hərf fərqini nəzərə almamaq üçün kiçik hərflərə çeviririk

    text = text.lower()

    # Mətnin içindəki boşluqları və işarələri silmək üçün yalnız hərfləri və rəqəmləri saxlayırıq

    cleaned\_text = ""

    for char in text:

        if char.isalnum():  # Yalnız hərf və rəqəm olan simvolları saxlayırıq

            cleaned\_text += char

    # Mətnin tərsini alıb müqayisə edirik

    return cleaned\_text == cleaned\_text[::-1]

# Test nümunələri

print(is\_palindrome("A man, a plan, a canal, Panama"))  # True (Palindromdur)

print(is\_palindrome("racecar"))  # True (Palindromdur)

print(is\_palindrome("Hello"))  # False (Palindrom deyil)

Python-da faylı açmağın ən təhlükəsiz yolu with ifadəsini open() funksiyası ilə birlikdə istifadə etməkdir. Bu, istisna baş versə belə, faylın düzgün şəkildə açılmasını, istifadə edilməsini və bitirdikdən sonra avtomatik bağlanmasını təmin edir. Bu yanaşma kontekst idarəçiliyi adlanır və o, istəmədən açıq qalan fayl tutacaqları kimi resurs sızmasının qarşısını almağa kömək edir.

**Task 41**

def word\_lengths(words):

    """

    This function takes a list of words and returns a new list containing

    the length of each word in the original list.

    Parameters:

    words (list): A list of strings (words).

    Returns:

    list: A list of integers representing the lengths of the words.

    """

    # Base case: If the list is empty, return an empty list

    if not words:

        return []

    # Recursive case: Get the length of the first word and call the function

    # recursively for the rest of the list

    first\_length = len(words[0])

    rest\_lengths = word\_lengths(words[1:])

    # Append the length of the first word to the results from the rest of the list

    return [first\_length] + rest\_lengths

# Example usage

word\_list = ["apple", "banana", "cherry"]

lengths = word\_lengths(word\_list)

print(lengths)  # Output: [5, 6, 6]

Set-mutable,non-ordering,non-duplicate

Tuple-immutable,ordering,duplicate

**Task 42**

def count\_words(sentence):

    """

    Bu funksiya verilmiş cümlədəki sözlərin sayını hesablayır və qaytarır.

    Parametr:

        sentence (str): Sözlərin sayını hesablayacağımız cümlə.

    Return:

        int: Cümlədəki sözlərin sayı.

    """

    # Boş sətir yoxlanışı

    if not sentence:

        return 0

    # Sözləri boşluğa görə ayırırıq

    words = sentence.split()

    # Sözlərin sayını qaytarırıq

    return len(words)

# İstifadəçi daxil etməsi

user\_input = input("Cümləni daxil edin: ")

print("Sözlərin sayı:", count\_words(user\_input))

Dictionary-mutable,ordering,non-duplicate

Set-mutable,non-ordering,non-duplicate

**Task 43**

def replace\_substring(original: str, target: str, replacement: str) -> str:

    if not target:  # Boş target üçün istisna

        return original

    return original.replace(target, replacement)

# Test nümunələri

print(replace\_substring("Hello world", "world", "Python"))  # "Hello Python"

print(replace\_substring("AaBbCc", "b", "x"))  # "AaBxCc"

print(replace\_substring("Test Test", "test", "check"))  # "Test Test" (case sensitive)

print(replace\_substring("abc", "", "x"))  # "abc" (empty target, no change)

**Task 44**

def sort\_words(sentence):

    words = sentence.split()  # Cümləni boşluqlara görə ayırırıq

    sorted\_words = sorted(words)  # Sözləri əlifba sırası ilə sıralayırıq

    return sorted\_words

# Test nümunəsi

sentence = "Python is a powerful programming language"

print(sort\_words(sentence))

A global variable is a variable that is declared outside of any function or class.It can be accessed and modified by any function or code block in the same file or module.A local variable is a variable that is declared inside a function or a block of code.It is only accessible within the function or block where it is defined, and once the function finishes executing, the variable is destroyed**.**

**Task 45**

def second\_largest(nums):

    """

    Siyahıdakı ikinci ən böyük ədədi qaytarır.

    Əgər mümkün deyilsə, None qaytarır.

    """

    unique\_nums = list(set(nums))  # Təkrarlanan ədədləri silirik

    if len(unique\_nums) < 2:

        return None  # Siyahıda 2-dən az unikal element varsa, ikinci ən böyük yoxdur

    unique\_nums.sort(reverse=True)  # Ədədləri azalan sıra ilə sıralayırıq

    return unique\_nums[1]  # İkinci ən böyüyü qaytarırıq

# Nümunə istifadələr:

print(second\_largest([10, 20, 4, 45, 99]))  # 45 qaytarmalıdır

print(second\_largest([1, 1, 1, 1]))  # None qaytarmalıdır

print(second\_largest([5]))  # None qaytarmalıdır

print(second\_largest([2, 2, 3, 1]))  # 2 qaytarmalıdır

Break ve return:return funksiyani dayandirib neticeni qaytarir,break dovru(loop) dayandirir

**Task 46**

def calculate\_average(numbers):

    """

    This function calculates the average of a list of numbers.

    Args:

        numbers (list): A list of numerical values.

    Returns:

        float: The calculated average of the numbers, or None if the list is empty.

    Raises:

        TypeError: If the input is not a list.

    """

    if not isinstance(numbers, list):

        raise TypeError("Input must be a list.")

    if len(numbers) == 0:

        # Return None for empty list

        return None

    total\_sum = sum(numbers)  # Calculate the sum of the list

    count = len(numbers)      # Get the count of numbers in the list

    average = total\_sum / count  # Calculate average

    return average  # Return the average value

# Example usage:

numbers\_list = [10, 20, 30, 40]

average\_value = calculate\_average(numbers\_list)

print(f"The average is: {average\_value}")

**In Python, pass is a placeholder statement that does nothing when executed. It is used in situations where a statement is syntactically required but no action is needed.**

**Task 47**

def is\_armstrong(number):

    # Ədədin rəqəmlərini string kimi götürüb uzunluğunu tapırıq

    num\_str = str(number)

    num\_digits = len(num\_str)

    # Rəqəmləri ayrı-ayrılıqda n-ci dərəcəyə yüksəldib cəmini tapırıq

    total = sum(int(digit) \*\* num\_digits for digit in num\_str)

    # Əgər cəm ədədin özünə bərabərdirsə, Armstrong ədədidir

    return total == number

# Test nümunələri

print(is\_armstrong(153))  # True

print(is\_armstrong(9474)) # True

print(is\_armstrong(9475)) # False

print(is\_armstrong(370))  # True

**The index() function in Python is used to find the position of a specified element in a sequence (such as a list, tuple, or string). It returns the first occurrence of the specified value.**

**Task 48**

def remove\_duplicates(input\_list):

    """

    This function removes duplicate elements from a given list.

    Parameters:

    input\_list (list): A list of numbers containing duplicates.

    Returns:

    list: A new list containing only unique elements.

    """

    # Using a set to remove duplicates

    unique\_elements = set(input\_list)

    # Converting the set back to a list

    return list(unique\_elements)

# Alternative method using a loop

def remove\_duplicates\_alt(input\_list):

    """

    This alternative function removes duplicates by looping through the list.

    Parameters:

    input\_list (list): A list of numbers containing duplicates.

    Returns:

    list: A new list containing only unique elements.

    """

    unique\_list = []

    for item in input\_list:

        # Add item to unique\_list if it's not already present

        if item not in unique\_list:

            unique\_list.append(item)

    return unique\_list

# Example usage

numbers = [1, 2, 2, 3, 4, 4, 5]

print(remove\_duplicates(numbers))  # Output: [1, 2, 3, 4, 5]

print(remove\_duplicates\_alt(numbers))  # Output: [1, 2, 3, 4, 5]

In Python, the range() function returns an object of type range. The range type is an immutable sequence that generates numbers lazily, meaning it doesn't store all the numbers in memory but generates them as needed. It supports iteration and indexing, making it efficient for loops and sequences

**Task 49**

def is\_perfect\_number(n):

    if n <= 0:

        return False  # Müsbət tam ədəd olmalıdır

    divisors\_sum = sum(i for i in range(1, n) if n % i == 0)

    return divisors\_sum == n

# Test nümunələri

print(is\_perfect\_number(28))  # True

print(is\_perfect\_number(6))   # True

print(is\_perfect\_number(12))  # False

print(is\_perfect\_number(100)) # False

**Use remove() when you want an error if the element is missing (ensuring it's present before deletion).Use discard() when you want to avoid errors if the element is absent.**

**Task 50**

import string

def word\_frequency\_counter(filename):

    try:

        with open(filename, 'r', encoding='utf-8') as file:

            text = file.read().lower()  # Bütün mətni kiçik hərflərə çeviririk

        text = text.translate(str.maketrans("", "", string.punctuation))  # İmlâ işarələrini silirik

        words = text.split()  # Sözləri ayırırıq

        word\_count = {}

        for word in words:

            word\_count[word] = word\_count.get(word, 0) + 1

        sorted\_words = sorted(word\_count.items(), key=lambda x: x[1], reverse=True)  # Azalan sıraya görə sıralayırıq

        for word, count in sorted\_words:

            print(f"{word}: {count}")

    except FileNotFoundError:

        print("Fayl tapılmadı. Zəhmət olmasa düzgün fayl adı daxil edin.")

# Test üçün fayl adı

word\_frequency\_counter("sample.txt")  # Burada "sample.txt" sənin fayl adın olmalıdır

1. del – Açarı (key) silir, əgər açar yoxdursa, KeyError verir.

2. pop(key) – Açarı silir və onun dəyərini qaytarır. Açar yoxdursa, KeyError verə bilər.

3. popitem() – Son əlavə olunan elementi silir və (key, value) qaytarır.

4. clear() – Lüğətin bütün elementlərini silir.

**Task 51**

def find\_character\_indices(text, char):

    first\_index = text.find(char)

    if first\_index == -1:

        return -1, -1

    last\_index = text.rfind(char)

    return first\_index, last\_index

# Alternative implementation that raises an error

def find\_character\_indices\_strict(text, char):

    first\_index = text.find(char)

    if first\_index == -1:

        raise ValueError(f"Character '{char}' not found in string")

    last\_index = len(text) - 1 - text[::-1].find(char)  # Using slicing as required

    return first\_index, last\_index

**Task 52**

def count\_word\_occurrences(sentence, word):

    """

    Counts how many times a word appears in a sentence (case-insensitive).

    Args:

        sentence: The input string to search in

        word: The word to search for

    Returns:

        int: Number of occurrences of the word

    """

    words\_in\_sentence = sentence.lower().split()

    target\_word = word.lower()

    return words\_in\_sentence.count(target\_word)

**Task 53**

Python's len() function works similarly for both strings and lists, but there are some key differences:

**\***Common Behavior:\*

- Returns the number of elements/characters

- Works with any sequence type in Python

- Time complexity is O(1) for both

\*With Strings:\*

- Counts the number of Unicode characters

- Example: len("hello") returns 5

- Includes whitespace and special characters in count

\*With Lists:\*

- Counts the number of elements in the list

- Example: len([1, 2, 3]) returns 3

- Counts each element regardless of its type or size

def get\_string\_length(input\_string):

    """

    Returns the length of a string or a message for empty strings.

    Args:

        input\_string: The string to measure

    Returns:

        int or str: Length of string or "Empty string" message

    """

    if not input\_string:  # This handles None and empty string

        return "Empty string"

    return len(input\_string)

**Task 54**

These are both sequence types in Python, but with important differences:

\*Mutability:\*

- list() creates a mutable sequence (can be modified after creation)

- tuple() creates an immutable sequence (cannot be modified after creation)

def starts\_with\_ignore\_case(main\_string, prefix):

    """

    Checks if a string starts with a specified prefix (case-insensitive).

    Args:

        main\_string: The string to check

        prefix: The prefix to look for

    Returns:

        bool: True if the string starts with the prefix (case-insensitive), False otherwise

    """

    return main\_string.lower().startswith(prefix.lower())

**Task 55**

These are both string methods but serve different purposes:

\*index() Method:\*

- Returns the lowest index where the substring is found

- Raises ValueError if substring is not found

- Syntax: str.index(sub[, start[, end]])

- Useful when you need the position of a substring

\*count() Method:\*

- Returns the number of occurrences of a substring

- Returns 0 if substring is not found (no error)

- Syntax: str.count(sub[, start[, end]])

- Useful when you need occurrence statistics

def is\_txt\_file(filename):

    """

    Checks if a filename ends with .txt extension (case-insensitive).

    Args:

        filename: The filename to check

    Returns:

        bool: True if filename ends with .txt extension, False otherwise

    """

    return filename.lower().endswith('.txt')

**Task 56**

def clean\_whitespace(input\_string):

    """

    Removes leading and trailing whitespace from a string and prints the result.

    Handles edge cases like all-space strings.

    Args:

        input\_string: The string to clean

    Returns:

        None: Prints the cleaned string directly

    """

    cleaned = input\_string.strip()

    print(f"Cleaned string: '{cleaned}'")  # Quotes show exact boundaries

hese string methods serve completely different purposes:

\*strip() Method:\*

- Removes leading and trailing characters (whitespace by default)

- Returns a new string

- Syntax: str.strip([chars])

- Modifies string boundaries

- Useful for cleaning user input or file data

\*split() Method:\*

- Splits a string into a list of substrings

- Uses a delimiter (whitespace by default)

- Syntax: str.split(sep=None, maxsplit=-1)

- Divides string content

- Useful for parsing text or CSV data

**Task 58**

def replace\_word(sentence):

    # Replace 'Python' with 'JavaScript' in a case-insensitive manner

    return sentence.replace('Python', 'JavaScript').replace('python', 'JavaScript').replace('PYTHON', 'JavaScript')

# Example usage

input\_sentence = "Python is an awesome programming language. I love python!"

output\_sentence = replace\_word(input\_sentence)

print(output\_sentence)

Python-da bir mətnin **yalnız ilk uyğun gələn sözünü dəyişdirmək üçün** replace() və re.sub() metodlarından istifadə olunur.

✅ **replace() metodu** üçüncü parametr kimi **dəyişdiriləcək maksimum sayı təyin etməyə imkan verir**.  
✅ **re.sub() metodu** isə **regular ifadələrlə işləyir** və count=1 parametri ilə yalnız ilk uyğunluğu əvəzləyə bilər.

**Task 59**

def remove\_prefix(string, prefix):

    """

    Removes the specified prefix from the string if it exists.

    Parameters:

    string (str): The original string.

    prefix (str): The prefix to be removed.

    Returns:

    str: The modified string without the prefix.

    """

    # Check if the string starts with the prefix

    if string.startswith(prefix):

        return string[len(prefix):]  # Remove the prefix

    return string  # Return original string if prefix not found

# Example usage

result = remove\_prefix("55HelloWorld", "55")

print(result)  # Output: HelloWorld

Python-da **startswith('55') funksiyası** bir sətirin "55" ilə başlayıb-başlamadığını yoxlayır.

✅ Əgər başlayırsa, **True**, başlamırsa, **False** qaytarır.  
✅ **Alternativ çap üsulları:**

* if-else istifadə edərək nəticəni ayrıca çap etmək
* Ternary operator (print(True if ... else False)) ilə bir sətirdə yazmaq
* int() ilə True → 1, False → 0 kimi göstərmək.

**Task 60**

def remove\_suffix(filename):

    """

    Removes the .html suffix from a given filename if it exists.

    Parameters:

    filename (str): The filename from which to remove the suffix.

    Returns:

    str: The filename without the .html suffix.

    """

    # Check if the filename ends with .html

    if filename.endswith('.html'):

        # Remove the suffix

        return filename[:-5]

    return filename

# Example usage

file\_name\_1 = "index.html"

file\_name\_2 = "document.txt"

print(remove\_suffix(file\_name\_1))  # Output: index

print(remove\_suffix(file\_name\_2))  # Output: document.txt

Python-da **tuple dəyişməzdir (immutable)**, yəni birbaşa elementlərini **dəyişmək mümkün deyil**.

**Task61**

def absolute\_value\_message(number):

    """

    Bu funksiya bir tam ədəd qəbul edir və onun mütləq dəyərini

    "This is absolute value" ifadəsi ilə birlikdə qaytarır.

    """

    # Mütləq dəyəri tap

    abs\_value = abs(number)

    # Format funksiyası ilə düzgün qaydada çap et

    message = "This is absolute value: {}".format(abs\_value)

    return message

# Funksiyanı test edək

print(absolute\_value\_message(-10))  # This is absolute value: 10

print(absolute\_value\_message(5))    # This is absolute value: 5

print(absolute\_value\_message(-100)) # This is absolute value: 100

 round() funksiyası **ədədi yuvarlaqlaşdıraraq** 2 rəqəm saxlayır.

 format() və f-string metodları isə **ədədi sətir kimi formatlayır**, lakin dəyəri dəyişmir.

**Task62**

def calculate\_power():

    """

    Bu funksiya istifadəçidən iki ədəd (x və y) alır və x^y-ni hesablayır.

    Həm tam, həm də onluq ədədləri qəbul edir.

    """

    try:

        # İstifadəçidən ədədləri al

        x = float(input("Əsas (x) ədədini daxil edin: "))

        y = float(input("Üst (y) ədədini daxil edin: "))

        # Üstlü qüvvəti hesabla

        result = x \*\* y

        # Nəticəni qaytar və formatla

        print(f"{x} ^ {y} = {result:.2f}")

    except ValueError:

        # Əgər istifadəçi səhv məlumat daxil edərsə

        print("Xəta! Zəhmət olmasa, yalnız ədədlər daxil edin.")

# Funksiyanı işə sal

calculate\_power()

x\*\*y sadedir ve birbasa hesablayir.Sadece quvvet hesablayir.

Pow(x,y) daha funksionaldir

**Task 63**

def round\_number():

    num = float(input("Ondalıklı ədəd daxil edin: "))

    rounded\_value = round(num, 2)

    print(f"Yuvarlaqlaşdırılmış dəyər: {rounded\_value}")

round\_number()

str() funksiyası ilə

Nümunə:

age = 25

message = "Mənim yaşım " + str(age) + " -dir."

print(message)

**Task 64**

def integer\_division(a, b):

    if b == 0:

        return "Error: Division by zero is not allowed"

    quotient = a // b  # Tam bölmə

    remainder = a % b   # Qalıq

    return quotient, remainder

**Python-da / və // operatorları bölmə üçün istifadə olunur, lakin fərqli nəticələr qaytarırlar:**

**/ (Adi bölmə): Həmişə float tipli nəticə qaytarır.**

**// (Tam bölmə): Bölmənin yalnız tam hissəsini qaytarır**

**Task 65**

def format\_number(num):

    formatted\_num = "{:>10.2f}".format(num)

    print(formatted\_num)

Python-da **sətiri bayt formatında çap etmək üçün** encode() metodundan istifadə olunur.

✅ **UTF-8 və ya başqa kodlaşdırma formatı seçilə bilər**.  
✅ **Bayt formatında çap edildikdə** nəticə b'' şəklində görünür.

**Task 66**

def factorial(n):

    if n < 0:

        print("Error: Factorial is not defined for negative numbers.")

        return None

    elif n == 0:

        return 1

    else:

        result = 1

        for i in range(1, n + 1):

            result \*= i

        return result

For döngüsü :döngünün neçə dəfə işləyəcəyini əvvəlcədən bilirik.

While döngüsü isə müəyyən bir şərt doğru olduğu müddətdə işləyir.

**Task 67**

String: Dəyişməzdir (immutable). Yəni, bir string yaradıldıqdan sonra onu dəyişdirmək olmaz. Yeni bir dəyər təyin etmək üçün yenidən string yaratmaq lazımdır.String: Yalnız simvollardan (character) ibarətdir.List: Dəyişkəndir (mutable). Siyahının (listin) elementlərini dəyişmək, silmək və ya yenisini əlavə etmək mümkündür.List: Fərqli tipdə (int, float, string, hətta başqa listlər) elementlər saxlayır**.**

# 1-ci üsul: Daxili (built-in) funksiyalarla

def min\_max\_builtin(numbers):

    """

    Bu funksiya daxili Python funksiyaları olan min() və max() ilə

    minimum və maksimum dəyərləri tapır.

    Əgər siyahı boşdursa, None qaytarır.

    """

    if not numbers:

        return None

    return min(numbers), max(numbers)

# 2-ci üsul: Dövr (loop) və müqayisə ilə

def min\_max\_manual(numbers):

    """

    Bu funksiya dövr (loop) istifadə edərək siyahıdakı

    minimum və maksimum dəyərləri əl ilə hesablayır.

    Əgər siyahı boşdursa, None qaytarır.

    """

    if not numbers:

        return None

    min\_val = numbers[0]

    max\_val = numbers[0]

    for num in numbers[1:]:

        if num < min\_val:

            min\_val = num

        if num > max\_val:

            max\_val = num

    return min\_val, max\_val

# Nümunə istifadə:

nums = [12, 5, 87, 3, 45, 99]

print("Built-in nəticə:", min\_max\_builtin(nums))

print("Manual nəticə:", min\_max\_manual(nums))

**Task 68**

def filter\_even\_numbers(lst):

    # Boş siyahı yaradılır ki, cüt ədədlər burada toplanacaq

    even\_numbers = []

    # Siyahıdakı hər bir ədəd üzərində dövr edilir

    for num in lst:

        # Əgər ədədin 2-yə bölümündən qalığı 0-dırsa (yəni ədəd cütdür)

        if num % 2 == 0:

            # O, cüt ədədlər siyahısına əlavə edilir

            even\_numbers.append(num)

    # Funksiya, yalnız cüt ədədlərin yerləşdiyi siyahını qaytarır

    return even\_numbers

# Funksiyanı test edirik

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

print(filter\_even\_numbers(numbers))  # Nəticə: [2, 4, 6, 8, 10]

tuple\_list = [("name", "Alice"), ("age", 25), ("city", "Baku")]

dictionary = dict(tuple\_list)

print(dictionary) # Nəticə: {'name': 'Alice', 'age': 25, 'city': 'Baku'}

**Task 69**

# Fibonacci ardıcıllığını döngü ilə yaradan funksiya

def fibonacci(n):

    """

    Bu funksiya istifadəçidən alınan 'n' sayda Fibonacci ədədlərini qaytarır.

    Fibonacci ardıcıllığı: 0, 1, 1, 2, 3, 5, 8, ...

    Əgər n < 1 olarsa, boş siyahı qaytarır.

    """

    if n <= 0:

        return []  # Mənfi və ya 0 üçün boş siyahı qaytarılır

    elif n == 1:

        return [0]  # Yalnız ilk element

    elif n == 2:

        return [0, 1]  # İlk iki element

    fib = [0, 1]  # Fibonacci başlanğıcı

    for i in range(2, n):

        next\_num = fib[-1] + fib[-2]  # Son iki ədədi toplayırıq

        fib.append(next\_num)  # Yeni ədəd siyahıya əlavə olunur

    return fib  # Tam Fibonacci siyahısı qaytarılır

# Nümunə istifadə:

print("Fibonacci ardıcıllığı:", fibonacci(10))

Pass - biz çoxlu funksiyalar yazacayıqsa amma içini helə yazmayacayıqsa error verilməsin deye pass yazırıq yəni sonradan yaza bilək deye

**Task 70**

def count\_vowels(s):

    vowels = "aeiou"

    return sum(1 for char in s.lower() if char in vowels)

# Example usage

text = "Hello, World!"

print(count\_vowels(text))  # Output: 3

In Python, you can convert an integer to a string using the str() function.

Example:

num = 123

str\_num = str(num)

print(str\_num) # Output: '123'

print(type(str\_num)) # Output: <class 'str'>

**Task 71**

def average(numbers):

    if not numbers:  # Check if the list is empty

        return 0

    return sum(numbers) / len(numbers)  # Calculate the average

# Example usage

print(average([1, 2, 3, 4, 5]))  # Output: 3.0

print(average([]))  # Output: 0

**1.**Using the union() method:

The union() method returns a new set with all the elements from both sets, removing duplicates.

2.Using the | (pipe) operator:

The | operator is a shorthand for union(). It combines both sets in the same way.

3. Using the update() method:

The update() method modifies the first set by adding elements from the second set.

Key Notes:

union() and | create a new set without modifying the original sets.

update() modifies the first set in place.

**Task 72**

def is\_palindrome(s):

    cleaned\_s = ''.join(s.lower().split())  # Convert to lowercase and remove spaces

    return cleaned\_s == cleaned\_s[::-1]  # Compare with its reverse

# Example usage

print(is\_palindrome("Racecar"))        # Output: True

print(is\_palindrome("A man a plan a canal Panama"))  # Output: True

print(is\_palindrome("Hello"))          # Output: False

The split() function in Python is used to divide a string into a list of substrings based on a specified delimiter (default is whitespace).

Syntax:

string.split(separator, maxsplit)

separator (optional): The delimiter where the string will be split. Default is any whitespace.

maxsplit (optional): The maximum number of splits to perform. Default is -1 (no limit).

**Task 73**

def multiplication\_table(n):

    for i in range(1, 11):

        print(f"{n} x {i} = {n \* i}")

# Example usage

multiplication\_table(5)

/ Division: Performs floating-point (decimal) division, always returns a float.

// Floor Division: Performs integer division, rounding down to the nearest whole number.

**Task 74**

def merge\_and\_sort(list1, list2):

    merged\_list = list1 + list2  # Merge the lists

    return sorted(merged\_list)   # Sort the merged list

# Example usage

list1 = [3, 1, 4]

list2 = [5, 2, 6]

print(merge\_and\_sort(list1, list2))  # Output: [1, 2, 3, 4, 5, 6]

**Feature sort() sorted()**

**Type Method of lists Built-in function**

**Modifies Original List? ✅ Yes (modifies in-place) ❌ No (returns a new sorted list)**

**Returns None (modifies the list directly) A new sorted list**

**Usable on Other Iterables? ❌ No, only for lists ✅ Yes, works with lists, tuples, sets, dictionaries, etc.**

**Task 75**

def count\_unique\_elements(lst):

    return len(set(lst))  # Convert list to a set to remove duplicates, then get the length

# Example usage

print(count\_unique\_elements([1, 2, 3, 2, 4, 5, 1]))  # Output: 5

print(count\_unique\_elements([1, 1, 1]))  # Output: 1

1. index()

Returns the first occurrence index of a specified value in a sequence (string, list, tuple, etc.).Raises a ValueError if the value is not found.2. count()Returns the total number of occurrences of a specified value in a sequence.Does not raise an error if the value is not found; it simply returns 0