**BCA203P- Python Programming**

**Part A Programs:**

**1. Write a python Program to find “Anagram” and “Palindrome”.**

|  |
| --- |
| **Anagram:**A word or phrase made by transposing the letters of another word or phrase.  The word "secure" is an anagram of "rescue."  **Palindrome**: A word, phrase, or sequence that reads the same [backwards](https://www.google.com/search?sca_esv=f063ace4ba00ef26&q=backwards&si=ACC90nytWkp8tIhRuqKAL6XWXX-NJytwPzXFJzxvYyUSZicWZNRhIHxBEceODfF2S8UI6NuFp4ftMioOXCLtepxMZGXlYcgFpC8qOffjMe16hFtcWAD1PE0%3D&expnd=1&sa=X&ved=2ahUKEwj57oDGnICIAxVkumMGHQg7Hm8QyecJegQIIBAZ) as [forwards](https://www.google.com/search?sca_esv=f063ace4ba00ef26&q=forwards&si=ACC90nwZKElgOcNXBU934ENhMNgqQIrbhHkFZIckLeos0SYJMimjItGK0HsR9pZmIaiYDepz3rNbXqNTbun2AYI7Z_ABPpKKBfysST6viPExZltgG7tePBk%3D&expnd=1&sa=X&ved=2ahUKEwj57oDGnICIAxVkumMGHQg7Hm8QyecJegQIIBAa), e.g. [madam](https://www.google.com/search?sca_esv=f063ace4ba00ef26&q=madam&si=ACC90nwXlEU2j3qee_ajN1FbIPWBJq6CrJJYhj_P1GLEWlBujBbNIEvLzrb3xKlCOu8DFxZtz9Eq4iyDrR5v63_I7B2KJJn7ow%3D%3D&expnd=1&sa=X&ved=2ahUKEwj57oDGnICIAxVkumMGHQg7Hm8QyecJegQIIBAb), malayalam. |

**Code:**

def anagram\_check(s1, s2):

if(sorted(s1)== sorted(s2)):

print("The strings are anagrams.")

else:

print("The strings aren't anagrams.")

def palind\_check(p1):

rev\_p=p1[::-1]

if p1==rev\_p:

print("It's a palindrome")

else:

print("It's not a palindrome")

# driver code

print("\*\*\*\*Anagram\*\*\*\*")

s1 =input("Enter string1 : ")

s2 =input("Enter string2 :")

anagram\_check(s1, s2)

print(" ")

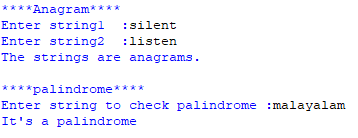
#driver code- palindrome

print("\*\*\*\*palindrome\*\*\*\*")

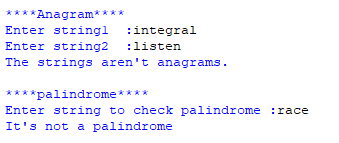
p1=input("Enter string to check palindrome : ")

palind\_check(p1)

**OUTPUT 1:**



**OUTPUT 2:**



**2. Write a python program to Demonstrate linear search.**

|  |
| --- |
| **Linear search is a sequential searching algorithm where we start from one end and check every element of the list until the desired element is found. It is the simplest searching algorithm.** |

**Code:**

def linear\_Search(list1, n, key):

for i in range(0, n):

if list1[i] == key:

return i

return -1

print("\_\_\_\_\_\_LINEAR SEARCH\_\_\_\_")

# Input the number of elements

n = int(input("Enter number of elements: "))

# Initialize the list

list1 = []

# Input the elements

print("Enter the elements:")

for i in range(n):

ele = int(input()) # Ensure the elements are integers

list1.append(ele)

print("List elements are:", list1)

# Input the key to be searched

key = int(input("Enter key to be searched: "))

# Get the length of the list

n = len(list1)

# Perform the linear search

res = linear\_Search(list1, n, key)

# Output the result

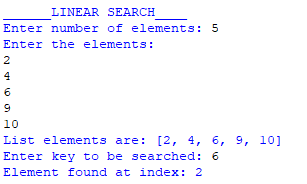
if res == -1:

print("Element not found")

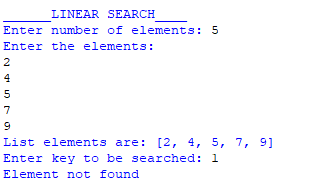
else:

print("Element found at index:", res)

**OUTPUT 1:**



**OUTPUT 2:**



**3. Write a Menu Driven Program to add and Delete Stationary items. Use a Dictionary for Adding Items and Brands.**

* + 1. **1.Add an item to the stationaries**
    2. **2.Remove an item from stationaries**
    3. **3.Display stationery**
    4. **4.exit**

|  |
| --- |
| Dictionaries are used to store data values in key:value pairs.  A dictionary is a collection which is ordered, changeable and do not allow duplicates. |

**Code:**

stationaries={}

while True:

print("Choose the Option from the Menu")

print("--------------------------------------------")

print("1.Add an item to the stationaries ")

print("2.Remove an item from stationaries ")

print("3.Display stationary ")

print("4.exit")

#key - brand

#value - item

ch=int(input("Enter your Choice: "))

if ch == 1:

brand = input("Enter the brand [key] to be inserted: ")

item = input("Enter the item to be inserted: ")

stationaries[brand] = item

print("Added successfully.")

elif ch == 2:

brand = input("Enter the brand [key] to be deleted: ")

if brand in stationaries:

del stationaries[brand]

print("Deleted successfully.")

else:

print("Invalid key.")

elif ch == 3:

print("Current Stationaries:", stationaries)

elif ch == 4:

print("Exiting!")

break

else:

print("Invalid choice! Please choose an option between 1 to 4.")

**Output:**

|  |
| --- |
| **Choose the Option from the Menu**  **--------------------------------------------**  **1.Add an item to the stationaries**  **2.Remove an item from stationaries**  **3.Display stationery**  **4.exit**  **Enter your Choice: 1**  **Enter the brand [key] to be inserted : doms**  **Enter the item to be inserted : pen**  **Added successfully**  **Enter your choice: 1**  **Enter the brand [key] to be inserted : Apsara**  **Enter the item to be inserted : pencil**  **Added successfully**  **Enter your choice: 3**  **{'doms': 'pen', 'Apsara': 'pencil'}**  **Enter your choice: 2**  **Enter the brand [key] to be deleted: doms**  **Deleted successfully**  **Enter your choice: 3**  **{'Apsara': 'pencil'}**  **Enter your choice: 2**  **Enter the brand [key] to be deleted: doms**  **invalid key**  **Enter your choice: 4**  **Exiting!!!!!** |

**4. Write a Python Program to Find the smallest element greater than K using list comprehension.**

|  |
| --- |
| A Python list comprehension consists of brackets containing the expression, which is executed for each element along with the for loop to iterate over each element in the Python list.  Syntax:  *newList* ***=******[*** *expression(element)* ***for*** *element* ***in*** *oldList* ***if*** *condition* ***]*** |

lst = []

#adding item into list

n = int(input("Enter number of elements : "))

print ("The", n ,"element in ascending order")

for i in range(0, n):

ele = int(input())

# adding the element

lst.append(ele)

print("Given list : ",lst)

#finding the smallest of value after k

k = int(input("enter the K value : "))

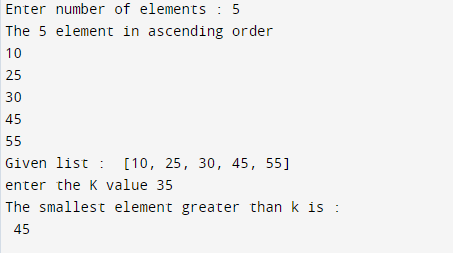
# using min

res = min(i for i in lst if i > k)

# Result

print("The smallest element greater than k is : \n" ,res)

**OUTPUT:**

****

**5. Write a Python program,**

* 1. **To remove empty tuples from the list.**

|  |
| --- |
| **Tuples- It is a collection of objects separated by commas. In some ways, a tuple is similar to a Python list in terms of indexing, nested objects, and repetition but the main difference between both is that the Python tuple is immutable(cannot change), unlike the Python list which is mutable(can change)** |

**Code**

def Remove(tuples):

for i in tuples:

if(len(i) == 0):

tuples.remove(i)

return tuples

# Driver Code

tuples = [(), ('alex', '15', '8'), (), ('jhon', 'roy'),('tom', 'joy', '45'), ('', ''), ()]

print(Remove(tuples))

**OUTPUT:**

|  |
| --- |
| **[('alex', '15', '8'), ('jhon', 'roy'), ('tom', 'joy', '45'), ('', '')]** |

* 1. **Find the largest and smallest element from the given list.**

lst = []

num = int(input('How many numbers: '))

for n in range(num):

numbers = int(input('Enter number '))

lst.append(numbers)

print("Maximum element in the list is :", max(lst), "\nMinimum element in the list is :", min(lst))

**OUTPUT:**

|  |
| --- |
| **How many numbers: 4**  **Enter number 10**  **Enter number 45**  **Enter number 7**  **Enter number 58**  **Maximum element in the list is: 58**  **Minimum element in the list is: 7** |

6. **Write a User Input Program to Check if a given number is a Prime Number or not.**

|  |
| --- |
| Any whole number which is greater than 1 and has only one factor that is 1 and the number itself, is called a prime number. A number is said to be prime if it is only divisible by 1 and itself. For example, 13 is a prime number because it is only divisible by 1 and 13, on the other hand 12 is not a prime number because it is divisible by 2, 4, 6 and number itself. |

**Code:**

#Taking input from user

num = int(input("Enter a number"))

if num ==1:

print("it is not prime")

#prime numbers is always greater than1

if num>1:

for i in range(2,num):

if num%i==0:

print("The number ",num,"is not a prime")

break

else:

print("The number ",num,"is a prime")

**Output1:**

|  |
| --- |
| Enter a number12  The number 12 is not a prime |

**Output2:**

|  |
| --- |
| Enter a number 5  The number 5 is a prime |

**7. Write a python program to,**

**a. To find sum of digit of a given number**

**b. To Find the Sum of n Natural numbers.**

**CODE**:

1. **To find sum of digit of a given number**

num = input("Enter Number: ")

sum = 0

for i in num:

sum = sum + int(i)

print(sum)

**OUTPUT**:

|  |
| --- |
|  |

**b. Find the Sum of n Natural numbers**

**Natural numbers are all positive integers from 1 to infinity.**

**CODE**:

n=int(input("Enter a number: "))

sum1 = 0

while(n > 0):

sum1=sum1+n

n=n-1

print("The sum of first n natural numbers is",sum1)

**OUTPUT**:

|  |
| --- |
|  |

**8. Write a Python program to add two matrices.**

|  |
| --- |
| Matrix, a set of numbers arranged in rows and columns so as to form a rectangular array |

**Code:**

X = [[1,2,3],

[4,5,6],

[7,8,9]]

Y = [[10,11,12],

[13,14,15],

[16,17,18]]

result = [[0,0,0],

[0,0,0],

[0,0,0]]

# iterate through rows

for i in range(len(X)):

# iterate through columns

for j in range(len(X[0])):

result[i][j] = X[i][j] + Y[i][j]

for r in result:

print(r)

**OUTPUT:**

|  |
| --- |
|  |

**9. Write a multithreaded program to create 2 threads where one thread calculates the factorial and second thread calculates square of a number.**

A program or process's smallest unit is called a thread, and it can run on its own or as part of a schedule set by the Operating System. Multitasking in a computer system is achieved by dividing a process into threads by an operating system. A string is a lightweight cycle that guarantees the execution of the interaction independently on the framework. When multiple processors are running on a program in Python 3, each processor runs simultaneously to carry out its own tasks.

**CODE:**

import threading

# Function to calculate factorial

def calculate\_factorial(num):

    factorial = 1

    for i in range(1, num + 1):

        factorial \*= i

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

# Function to calculate square

def calculate\_square(num):

    square = num \* num

    print(f"Square of {num} is {square}")

# Main function to create threads

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

    number = int(input("Enter a number ") )

    # Creating two threads

    thread1 = threading.Thread(target=calculate\_factorial, args=(number,))

    thread2 = threading.Thread(target=calculate\_square, args=(number,))

    # Starting the threads

    thread1.start()

    thread2.start()

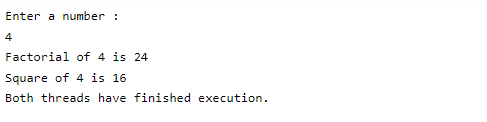
    # Waiting for both threads to finish

    thread1.join()

    thread2.join()

    print("Both threads have finished execution.")

**OUTPUT:**

****

**10. Write a Python program that opens a file and handles a FileNotFoundError exception if the file does not exist.**

**Exception Handling:**

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The else block lets you execute code when there is no error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

**CODE:**

def open\_file(filename):

    try:

        # Attempt to open the specified file in read mode ('r').

        file = open(filename, 'r')

        # Read the contents of the file and store them in the 'contents' variable.

        contents = file.read()

        # Print a message to indicate that the file contents will be displayed.

        print("File contents:")

        # Print the contents of the file.

        print(contents)

        # Close the file to release system resources.

        file.close()

    except FileNotFoundError:

        # Handle the exception if the specified file is not found.

        print("Error: File not found.")

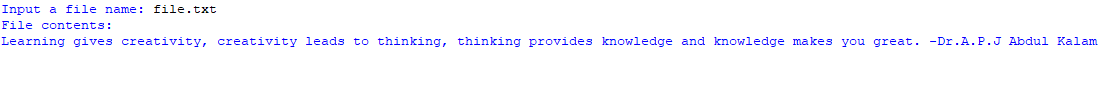
# Prompt the user to input a filename and store it in the 'file\_name' variable.

filename = input("Input a file name: ")

# Call the open\_file function with the provided file name.

open\_file(filename)

**OUTPUT1:**



**OUTPUT2:**

****