WEB PROGRAMMING LAB

ADITYA AGARWAL **2209015106** **ROLL NO. 14**

LAB4 – PYTHON BASICS

Q1 - Write a python program to reverse a content a file and store it in another file.

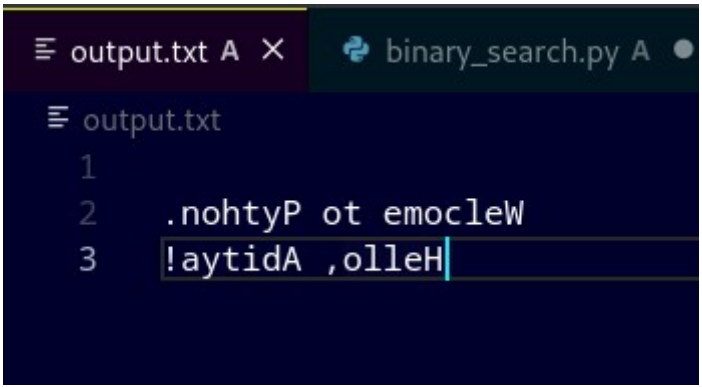
CODE:

```
def reverse_file_content(input_file, output_file):
    try:
        # Open the input file and read its content
        with open(input_file, 'r', encoding='utf-8') as file:
            content = file.read()

        # Reverse the content
        reversed_content = content[::-1]

        # Write the reversed content to the output file
        with open(output_file, 'w', encoding='utf-8') as file:
            file.write(reversed_content)

        print(f"Reversed content has been saved to {output_file}")

    except FileNotFoundError:
        print("Error: The input file does not exist.")
    except Exception as e:
        print(f"An error occurred: {e}")

# Example usage
input_filename = "input.txt"
output_filename = "output.txt"

reverse_file_content(input_filename, output_filename)
```

INPUT.txt:

```
Hello, Aditya!
Welcome to Python.
```

OUTPUT.txt:

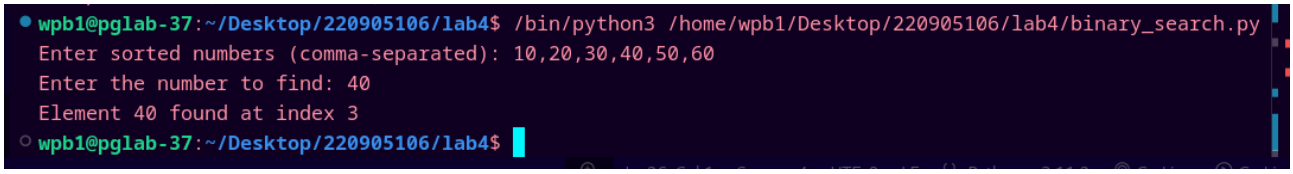


Q2 - Write a python program to implement binary search with recursion.

CODE:

```
def binary_search(arr, low, high, target):
    if low > high:
        return -1  # Base case: Element not found

    mid = (low + high) // 2

    if arr[mid] == target:
        return mid  # Found the target
    elif target < arr[mid]:
        return binary_search(arr, low, mid - 1, target)  # Search in left half
    else:
        return binary_search(arr, mid + 1, high, target)  # Search in right half

# Taking input from user
numbers = list(map(int, input("Enter sorted numbers (comma-separated): ").split(',')))
target = int(input("Enter the number to find: "))

# Performing binary search
result = binary_search(numbers, 0, len(numbers) - 1, target)

# Display result
if result != -1:
    print(f"Element {target} found at index {result}")
else:
    print(f"Element {target} not found")
```

OUTPUT:



Q3 - Write a python program to sort words in alphabetical order.

CODE:

```
sentence = input("Enter a sentence: ")
words = sentence.split()
words.sort()
print("Sorted words:", " ".join(words))
```

OUTPUT:

```
Element 40 found at index 3
wpb1@pglab-37:~/Desktop/220905106/lab4$ /bin/python3 /home/wpb1/Desktop/220905106/lab4/alpha_order.py
Enter a sentence: we are here solving python basics
Sorted words: are basics here python solving we
wpb1@pglab-37:~/Desktop/220905106/lab4$
```

Q4 - Write a Python class to get all possible unique subsets from a set of distinct integers.

Input:[4,5,6]

Output : [[], [6], [5], [5, 6], [4], [4, 6], [4, 5], [4, 5, 6]]

CODE:

```
class SubsetGenerator:
    def __init__(self, nums):
        self.nums = nums
        self.result = []

    def generate_subsets(self, index=0, current_subset=[]):
        # Add the current subset to the result
        self.result.append(current_subset[:])

        # Iterate through the remaining elements
        for i in range(index, len(self.nums)):
            # Include nums[i] in the current subset
            current_subset.append(self.nums[i])
            # Recursively call for the next index
            self.generate_subsets(i + 1, current_subset)
            # Backtrack: Remove the last element to explore the next possibility
            current_subset.pop()

    def get_subsets(self):
        self.generate_subsets()
        return self.result

# Example Usage
nums = [4, 5, 6]  # Input set
subset_gen = SubsetGenerator(nums)
unique_subsets = subset_gen.get_subsets()

# Print the output
print(unique_subsets)
```

OUTPUT:

```
wpb1@pglab-37:~/Desktop/220905106/lab4$ /bin/python3 /home/wpb1/Desktop/220905106/lab4/subsets.py
[[], [4], [4, 5], [4, 5, 6], [4, 6], [5], [5, 6], [6]]
wpb1@pglab-37:~/Desktop/220905106/lab4$
```

Q5 - Write a Python class to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number.
Input: numbers= [10,20,10,40,50,60,70], target=50
Output: 3, 4.

CODE:

```
class PairFinder:
    def __init__(self, numbers, target):
        self.numbers = numbers
        self.target = target

    def find_pair(self):
        index_map = {}  # Dictionary to store numbers and their indices

        for i, num in enumerate(self.numbers):
            complement = self.target - num  # Find the required pair value

            if complement in index_map:
                return index_map[complement], i  # Return indices of the pair

            index_map[num] = i  # Store the index of the current number

        return None  # Return None if no pair is found

# Example Usage
numbers = [10, 20, 10, 40, 50, 60, 70]
target = 50

finder = PairFinder(numbers, target)
result = finder.find_pair()

# Print the output
if result:
    print(result[0] + 1, result[1] + 1)  # Convert 0-based index to 1-based index
else:
    print("No pair found")
```

OUTPUT:

```
wpb1@pglab-37:~/Desktop/220905106/lab4$ /bin/python3 /home/wpb1/Desktop/220905106/lab4/sum_indices.py
3 4
wpb1@pglab-37:~/Desktop/220905106/lab4$
```

Q6 - Write a Python class to implement pow(x, n).

CODE:

```
class Power:
    def __init__(self, base, exponent):
        self.base = base
        self.exponent = exponent

    def my_pow(self):
        result = 1
        # Handle negative exponent
        if self.exponent < 0:
            self.base = 1 / self.base
            self.exponent = -self.exponent

        # Loop to multiply the base 'exponent' times
        for _ in range(self.exponent):
            result *= self.base
        return result

# Taking input from the user
x = float(input("Enter the base (x): "))  # Base (x)
n = int(input("Enter the exponent (n): "))  # Exponent (n)

# Creating an instance of Power class
power_calculator = Power(x, n)

# Calculating the result
result = power_calculator.my_pow()

# Displaying the result
print(f"The result of {x}^{n} is: {result}")
```

OUTPUT:

```
wpb1@pglab-37:~/Desktop/220905106/lab4$ /bin/python3 /home/wpb1/Desktop/220905106/lab4/pwer.py
Enter the base (x): 2
Enter the exponent (n): 5
The result of 2.0^5 is: 32.0
wpb1@pglab-37:~/Desktop/220905106/lab4$
```

Q7 - Write a Python class which has two methods get_String and print_String. The get_String accept a string from the user and print_String print the string in uppercase.

CODE:

```
class StringManipulator:
    def __init__(self):
        self.input_string = ""

    # Method to accept string input from the user
    def get_String(self):
        self.input_string = input("Enter a string: ")

    # Method to print the string in uppercase
    def print_String(self):
        print(self.input_string.upper())

# Example Usage
string_manipulator = StringManipulator()
string_manipulator.get_String()  # Get string input from user
string_manipulator.print_String()  # Print the string in uppercase
```

OUTPUT:

```
wpb1@pglab-37:~/Desktop/220905106/lab4$ /bin/python3 /home/wpb1/Desktop/220905106/lab4/uppercase.py
Enter a string: aditya agarwal working
ADITYA AGARWAL WORKING
wpb1@pglab-37:~/Desktop/220905106/lab4$
```