**LAB4: PYTHON BASICS**

Disha Jain - 220905554 – 61

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

def reverse\_file\_content(input\_file, output\_file):

try:

# Open the input file in read mode

with open(input\_file, 'r') as file:

content = file.read()

# Reverse the content

reversed\_content = content[::-1]

# Open the output file in write mode and write the reversed content

with open(output\_file, 'w') as file:

file.write(reversed\_content)

print(f"Content from '{input\_file}' has been reversed and written to '{output\_file}'.")

except FileNotFoundError:

print(f"Error: The file '{input\_file}' was not found.")

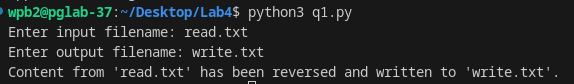
except Exception as e:

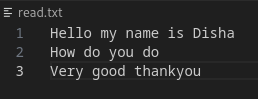
print(f"An error occurred: {e}")

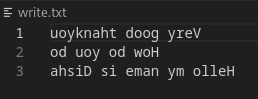
input\_file = input("Enter input filename: ")

output\_file = input("Enter output filename: ")

reverse\_file\_content(input\_file, output\_file)







1. Write a python program to implement binary search with recursion.

def Binary(ele,low,high,arr):

if high>=low:

mid=(high+low)//2

if arr[mid]==ele:

return mid

elif arr[mid]>ele:

return Binary(ele,low,mid-1,arr)

else:

return Binary(ele,mid+1,high,arr)

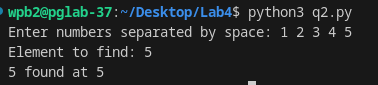
arr = list(map(int, input("Enter numbers separated by space: ").split()))

high = len(arr) - 1

ele = int(input("Element to find: "))

pos = Binary(ele,0,high,arr)

print(f"{ele} found at {pos+1}")



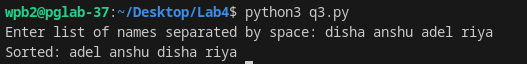
1. Write a python program to sort words in alphabetical order.

arr=(input("Enter list of names separated by space: ")).split()

arr.sort(key=lambda x: x.lower()) #to make sure it doesnt differentiate b/w capital and small

print("Sorted:",end=" ")

print(" ".join(arr))



1. 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]]

class Subsets:

def \_\_init\_\_(self, nums):

self.nums = nums

def generate\_subsets(self):

result = [[]]

for num in self.nums:

result+=[item + [num] for item in result]

return result

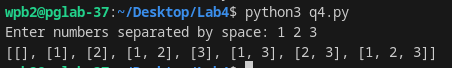
nums = list(map(int, input("Enter numbers separated by space: ").split()))

subsets\_generator = Subsets(nums)

subsets = subsets\_generator.generate\_subsets()

# Print all unique subsets

print(subsets)



1. 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.

class PairSum:

def \_\_init\_\_(self, numbers, target):

self.numbers = numbers

self.target = target

def find\_pair(self):

seen = {} # Dictionary to store the number and its index

for i, num in enumerate(self.numbers):

complement = self.target - num

if complement in seen:

# If complement is found, return the indices

return seen[complement]+1, i+1

seen[num] = i

return None # If no pair is found

# Example usage

numbers = list(map(int, input("Enter numbers separated by space: ").split()))

target = int(input("Target sum: "))

pair\_finder = PairSum(numbers, target)

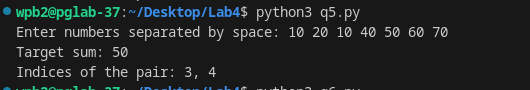
result = pair\_finder.find\_pair()

if result:

print(f"Indices of the pair: {result[0]}, {result[1]}")

else:

print("No pair found")



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

class Power:

def \_\_init\_\_(self, x, n):

self.x = x

self.n = n

def my\_pow(self):

return self.\_power(self.x, self.n)

def \_power(self, x, n):

if n == 0:

return 1 # Base case: x^0 = 1

elif n < 0:

return 1 / self.\_power(x, -n) # Handle negative powers

elif n % 2 == 0:

half = self.\_power(x, n // 2)

return half \* half # If n is even, use (x^(n/2))^2

else:

return x \* self.\_power(x, n - 1) # If n is odd, use x \* x^(n-1)

# Example usage

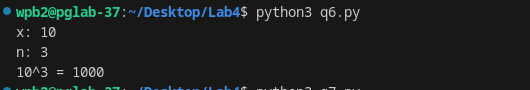
x = int(input("x: "))

n = int(input("n: "))

power\_calculator = Power(x, n)

result = power\_calculator.my\_pow()

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



1. 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 upper case.

class StringManipulator:

def \_\_init\_\_(self):

self.user\_string = ""

def get\_String(self):

self.user\_string = input("Enter a string: ")

def print\_String(self):

print("Uppercase string: {}".format(self.user\_string.upper()))

# Example usage:

string\_manipulator = StringManipulator()

string\_manipulator.get\_String()

string\_manipulator.print\_String()

