Sapthami_Upadhya_230905090_L4

January 28, 2026

Name: Sapthami Upadhya
Section: CSE A
Roll No.: 15
Reg. No.: 230905090

Week 4

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

[1]:
```
with open("sample.txt", "r") as file:
    content = file.read()
    print("Original: \n", content)

rev = content[::-1]

with open("reversed.txt", "w") as file:
    file.write(rev)

with open("reversed.txt", "r") as file:
    content = file.read()
    print("Reversed: \n", content)
```

```
Original:
 File handling refers to the process of performing operations on a file, such as
creating, opening, reading, writing and closing it through a programming
interface.
It involves managing the data flow between the program and the file system on
the storage device, ensuring that data is handled safely and efficiently.
Why do we need File Handling
    - To store data permanently, even after the program ends.
    - To access external files like .txt, .csv, .json, etc.
    - To process large files efficiently without using much memory.
    - To automate tasks like reading configs or saving outputs.
Reversed:
 .stuptuo gnivas ro sgifnoc gnidaer ekil sksat etamotua oT -
.yromem hcum gnisu tuohtiw yltneiciffe selif egral ssecorp oT -
.cte ,nosj. ,vsc. ,txt. ekil selif lanretxe ssecca oT -
.sdne margorp eht retfa neve ,yltnenamrep atad erots oT -
gnildnaH eliF deen ew od yhW
 .yltneiciffe dna ylefas deldnah si atad taht gnirusne ,ecived egarots eht no
```

```
metsys elif eht dna margorp eht neewteb wolf atad eht gniganam sevlovni tI
 .ecafretni gnimmargorp a hguorht ti gnisolc dna gnitirw ,gnidaer ,gninepo
,gnitaerc sa hcus ,elif a no snoitarepo gnimrofrep fo ssecorp eht ot srefer
gnildnah eliF
```

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

[ ]:
```
def bin_search(arr, start, end, key):
    if start > end: return -1
    mid = (start+end)//2
    if key < arr[mid]:
        return bin_search(arr, start, mid-1, key)
    elif key > arr[mid]:
        return bin_search(arr, mid+1, end, key)
    else: return mid

n = int(input("Enter no. of elements: "))
print("Enter the array elements: ")
arr = []
for i in range(n):
    arr.append(int(input()))

arr.sort()
print(arr)

key = int(input("Enter the key: "))
pos =  bin_search(arr, 0, n-1, key)
print(key, "found in position", pos) if pos != -1 else print(key, " not found")

#to show output when key is absent

key = int(input("Enter the key: "))
pos =  bin_search(arr, 0, n-1, key)
print(key, "found in position", pos) if pos != -1 else print(key, "not found")
```

```
Enter the array elements:
[1, 2, 3, 4, 5, 6, 7]
7 found in position 6
9 not found
```

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

[5]:
```
li = ["strawberry", "banana", "apple"]
li.sort()
print(li)
```

```
['apple', 'banana', 'strawberry']
```

4. 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]]

[12]:
```
class Subsets:
    '''This is a class to find all unique subsets from a set of distinct␣
 ↪integers'''
    def __init__(self):
        self.arr = []

    def populate(self, n):
        print("Enter set elements:")
        for i in range(n):
            self.arr.append(int(input()))

    def get_subsets(self):
        powset = []
        n = len(self.arr)
        for i in range(1 << n): #traverse all subsets, bits from 0 to 2^n-1
            subset = []
            for j in range(n): #traverse arr
                if i & (1<<j): #if jth bit is set in i, add it to current subset
                    subset.append(self.arr[j])
            powset.append(subset)
        return powset

obj = Subsets()
n = int(input("Enter no. of elements in the set: "))
obj.populate(n)
powset = obj.get_subsets()
print("Set: ", obj.arr)
print("Powerset: ", powset)
```

```
Enter set elements:
Set:  [4, 5, 6]
Powerset:  [[], [4], [5], [4, 5], [6], [4, 6], [5, 6], [4, 5, 6]]
```

5. 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.

[17]:
```
class Sumpair:
    def __init__(self):
        self.arr = []

    def populate(self, n):
        print("Enter set elements:")
        for i in range(n):
            self.arr.append(int(input()))
```

```
    def twosum(self, target):
        n = len(self.arr)
        for i in range(n):
            for j in range(i+1, n):
                if self.arr[i] + self.arr[j] == target:
                    return [i+1, j+1]
        return [-1, -1]

obj = Sumpair()
n = int(input("Enter no. of elements: "))
obj.populate(n)
print(obj.arr)
target = int(input("Enter the target sum: "))
n1, n2 = obj.twosum(target)
if(n1 != -1): print("The two indices: ", n1, n2)
else: print("No such pairs")
```

```
Enter set elements:
[10, 20, 10, 40, 50, 60, 70]
The two indices:  1 4
```

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

[18]:
```
class Power:
    def pow(self, x, n):
        result = 1
        for i in range(n):
            result *= x
        return result

x = int(input("Base: "))
n = int(input("Power: "))
obj = Power()
print(x, "^", n, "=", obj.pow(x, n))
```

```
5 ^ 2 = 25
```

7. 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.

[19]:
```
class StringOp:
    def __init__(self):
        self.str = ""
    def get_String(self):
        self.str = input("Enter a string: ")
    def print_String(self):
        print(self.str.upper())
```

```
obj = StringOp()
obj.get_String()
obj.print_String()
```

```
HELLO WORLD
```