ASSIGMENT – 2

1. Write a program that counts the number of tabs, spaces, and newline characters in a file.

ANS:

def count\_tabs\_spaces\_newlines(filename):

try:

with open(filename, 'r') as file:

content = file.read()

tabs\_count = content.count('\t')

spaces\_count = content.count(' ')

newlines\_count = content.count('\n')

return tabs\_count, spaces\_count, newlines\_count

except FileNotFoundError:

print(f"File '{filename}' not found.")

return 0, 0, 0

except Exception as e:

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

return 0, 0, 0

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

filename = input("Enter the name of the file: ")

tabs, spaces, newlines = count\_tabs\_spaces\_newlines(filename)

print(f"Tabs: {tabs}")

print(f"Spaces: {spaces}")

print(f"Newlines: {newlines}")

1. Write a program that validates name and age as entered by the user to determine whether the person can cast vote or not. Raise an invalid name and age exception if person types wrong.

ANS:

class InvalidNameError(Exception):

pass

class InvalidAgeError(Exception):

pass

def can\_vote(name, age):

try:

if not name.isalpha():

raise InvalidNameError("Invalid name. Please enter a valid name with only alphabetic characters.")

age = int(age)

if age < 18:

raise InvalidAgeError("You are not eligible to vote. You must be 18 years or older.")

print(f"Hello, {name}! You are eligible to vote.")

except InvalidNameError as name\_error:

print(name\_error)

except InvalidAgeError as age\_error:

print(age\_error)

except ValueError:

print("Invalid age. Please enter a valid age as a number.")

except Exception as e:

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

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

name = input("Enter your name: ")

age = input("Enter your age: ")

can\_vote(name, age)

3. Write a python program to prompt the users for a list of integers, stores the even numbers in one list and odd numbers in another list and display the resultant lists. Also find the ODD maximum and Even minimum from the list.

Ex: List =[1,3,56,8,97,200,10,33,57]

O/P: Evenlist=[56,8,200,10]

Oddlist=[1,3,97, 33,57]

ODD Maximum : 57

ANS:

# Function to get a list of integers from the user

def get\_integer\_list():

input\_str = input("Enter a list of integers separated by spaces: ")

integer\_list = [int(x) for x in input\_str.split()]

return integer\_list

# Main program

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

# Get the list of integers from the user

input\_list = get\_integer\_list()

# Initialize separate lists for even and odd numbers

even\_list = []

odd\_list = []

for num in input\_list:

if num % 2 == 0:

even\_list.append(num)

else:

odd\_list.append(num)

# Find the odd maximum and even minimum

odd\_max = max(odd\_list, default=None)

even\_min = min(even\_list, default=None)

# Display the resultant lists and statistics

print("Evenlist:", even\_list)

print("Oddlist:", odd\_list)

print("ODD Maximum:", odd\_max)

print("Even Minimum:", even\_min)

4.Write a Python program to generate and print a list except for the first 5 elements, where the values are square of numbers between 1 and 30 (both included)

ANS:

# Generate a list of squares of numbers from 1 to 30

squares\_list = [x\*\*2 for x in range(1, 31)]

# Print the list, excluding the first 5 elements

print("List of squares (excluding the first 5 elements):")

print(squares\_list[5:])

5. Write a program that has a dictionary of states and their codes. Add another state in the predefined dictionary, print all the items in the dictionary, and try to print code for a state that does not exist. Set a default value prior to printing.

ANS:

# Predefined dictionary of states and their codes

states\_dict = {

"California": "CA",

"New York": "NY",

"Texas": "TX",

"Florida": "FL",

}

# Add a new state to the dictionary

states\_dict["Washington"] = "WA"

# Print all items in the dictionary

print("All items in the dictionary:")

for state, code in states\_dict.items():

print(f"{state}: {code}")

# Print the code for a state that may not exist, setting a default value

state\_to\_lookup = "Oregon"

default\_value = "N/A"

state\_code = states\_dict.get(state\_to\_lookup, default\_value)

print(f"The code for {state\_to\_lookup} is {state\_code}")