# UML Activity Diagram

if **1**

if **2**

if **3**

if **4**

if **5**

if **6**

if **7**

if **8**

Start

Select a choice:

1, 2, 3, 4, 5, 6, 7, 8

Select one option

get\_top\_countries\_by\_year

find\_country\_rank

top\_3\_Countries\_with\_most\_first\_positions

list\_countries

group\_countries\_by\_rank\_ranges

countries\_with\_index\_above

countries\_with\_consecutive\_lower\_ranks

country\_details

year, top\_from\_bottom, top\_count

country\_name, period

top\_from\_bottom

descending\_order

No input

threshold\_value

consecutive\_years

Country name

top/least happiest countries

rank changes

top countries with most first positions

list of countries

groups of countries by rank range

countries with indexes above the threshold

countries with consecutive lower ranks

Country details

END

# Full Code Print Out

**all\_task.py file**

####################### task 1 ##############################

# Function to separate country data by a specific year

def separate\_countries\_by\_year(data, year):

    year\_i = []

    country\_i = []

    index\_i = []

    rank\_i = []

    for row in data:

        cntry, yr, ind, rnk = row

        # Check if the row matches the specified year and index is not empty

        if yr == year and ind != '':

            country\_i.append(cntry)

            year\_i.append(yr)

            index\_i.append(float(ind))

            rank\_i.append(rnk)

    return country\_i, year\_i, index\_i, rank\_i

# Function to perform selection sort based on index values

def selection\_sort(index\_i, country\_i):

    # Create a list of tuples pairing elements from lists index\_i and country\_i

    paired\_lists = list(zip(index\_i, country\_i))

    # Perform a selection sort on list index\_i and synchronize the sorting on list country\_i in descending order

    for i in range(len(index\_i)):

        max\_index = i

        for j in range(i + 1, len(index\_i)):

            if paired\_lists[j][0] > paired\_lists[max\_index][0]:

                max\_index = j

        # Swap elements in the paired list

        paired\_lists[i], paired\_lists[max\_index] = paired\_lists[max\_index], paired\_lists[i]

    # Extract the sorted elements from list country\_i

    sorted\_countries = [pair[1] for pair in paired\_lists]

    return list(zip(sorted\_countries, sorted(index\_i, reverse=True)))

# Function to get top countries by year

def get\_top\_countries\_by\_year(data, specific\_year=None, top\_count=5, top\_from\_bottom=False, print\_all=True):

    year\_list = ['2013','2015','2016','2017','2018','2019','2020','2021','2022','2023']

    # Process for a specific year if provided and it exists in the year list

    if specific\_year:

        if specific\_year in year\_list:

            country\_i, year\_i, index\_i, rank\_i = separate\_countries\_by\_year(data, specific\_year)

            top\_countries = selection\_sort(index\_i, country\_i)

            if top\_from\_bottom:

                print(f"Top {top\_count} least happiest countries in year {specific\_year}\n", top\_countries[-top\_count:], '\n\n')

            else:

                print(f"Top {top\_count} most happiest countries in year {specific\_year}\n", top\_countries[:top\_count], '\n\n')

        else:

            print("Please select year that is in this list {year\_list}")

    # Process for all years if print\_all is True and no specific year is provided

    if print\_all and not specific\_year:

        for year in year\_list: # to print data for each year

            country\_i, year\_i, index\_i, rank\_i = separate\_countries\_by\_year(data, year)

            top\_countries = selection\_sort(index\_i, country\_i)

            if top\_from\_bottom:

                print(f"Top {top\_count} least happiest countries in year {year}\n", top\_countries[-top\_count:], '\n\n')

            else:

                print(f"Top {top\_count} most happiest countries in year {year}\n", top\_countries[:top\_count], '\n\n')

############################ task 2 ####################

# select top\_from\_bottom=True if you want to print most unhappiest countries

# Function to find top countries with the most first positions in the index across 10 years

def top\_3\_Countries\_with\_most\_first\_positions(data, top\_count=5, top\_from\_bottom=False):

    year\_list = ['2013','2015','2016','2017','2018','2019','2020','2021','2022','2023']

    result = []

    # Iterate through each year in the year\_list

    for year in year\_list:

        # Retrieve country, year, index, and rank data for each year

        country\_i, year\_i, index\_i, rank\_i = separate\_countries\_by\_year(data, year)

        # Perform selection sort based on the index for the current year

        top\_countries = selection\_sort(index\_i, country\_i)

        # Append top countries to result based on top\_from\_bottom condition

        if top\_from\_bottom:

            result.append((year, top\_countries[-top\_count:]))

        else:

            result.append((year, top\_countries[:top\_count]))

    top\_in\_10\_years = []

    # Extract the top country in each year's top list and create a list of top countries over 10 years

    for first in result:

        top\_in\_10\_years.append(first[1][0][0])

    count\_dict = {}

    # Count occurrences of each item in the list

    for item in top\_in\_10\_years:

        if item in count\_dict:

            count\_dict[item] += 1

        else:

            count\_dict[item] = 1

    # Create a list of tuples with unique items and their counts

    result\_list = [(value, key) for key, value in count\_dict.items()]

    print("top 3 Countries with most first positions")

    return sorted(result\_list, reverse=True)[:3]

############################# task 3 ####################

# Function to find whether a country's rank has increased or decreased over a specific period

def find\_country\_rank(data, country, period):

    one\_country = []

    # Iterate through the data to collect rows pertaining to the specified country

    for i, row in enumerate(data):

        if row[0] == country:

            one\_country.append(row)

    # Check the change in rank over the specified period

    if float(one\_country[0][3]) > float(one\_country[period - 1][3]):

        print(f"{country} rank decreasing over period of {period} years")

    else:

        print(f"{country} rank increasing over period of {period} years")

################################# task 4 ###################################

# function to find list of countries

def list\_countries(data, dsc=False): # select dsc true if you want descending order

    countries = []

    for row in data:

        countries.append(row[0])

    return sorted(list(set(countries)), reverse=dsc)

################################# task 5 ###################################

# function to find countries with or above specific index value

def countries\_with\_index\_above(data, index\_threshold):

    countries\_above\_index = []

    for row in data:

        if row[2] and float(row[2]) >= index\_threshold:  # Ensure index is available and meets the threshold

            countries\_above\_index.append((row[0], float(row[2])))  # Store country and index as tuple

    # Sort countries in descending order based on their index

    sorted\_countries = sorted(countries\_above\_index, key=lambda x: x[1], reverse=True)

    return sorted\_countries

################################# task 6 ###################################

# group contries contries by rank

def group\_countries\_by\_rank\_ranges(data):

    # Filter data for the last 5 years

    last\_5\_years\_data = [row for row in data if int(row[1]) >= 2019]

    # Initialize a dictionary to store countries by rank ranges

    rank\_ranges = {f"{i}-{i+9}": [] for i in range(1, 151, 10)}

    # Group countries by rank ranges for the last 5 years

    for row in last\_5\_years\_data:

        rank = row[3]

        if rank and rank!="":  # Check if rank is available and numeric

            rank = int(float(rank))

            for start\_rank in range(1, 151, 10):

                end\_rank = start\_rank + 9

                if start\_rank <= rank <= end\_rank:

                    rank\_ranges[f"{start\_rank}-{end\_rank}"].append(row[0])

                    break  # Stop checking other ranges once added

    return rank\_ranges

################################# task 7 ###################################

# countries\_with\_consecutive\_lower\_ranks over specific period

def countries\_with\_consecutive\_lower\_ranks(data, consecutive\_years):

    countries = set()

    for i in range(len(data)):

        lower\_count = 0

        for j in range(consecutive\_years):

            # Check for missing or non-numeric rank values

            if i + j < len(data) - 1 and data[i + j][3] and data[i + j + 1][3]:

                if data[i + j][3]!="" and data[i + j + 1][3]!="":

                    if int(float(data[i + j][3])) > int(float(data[i + j + 1][3])):

                        lower\_count += 1

                    else:

                        break  # Reset count if ranks are not consecutive

                else:

                    break  # Reset count if rank values are not numeric

            else:

                break  # Reset count if rank values are missing

        if lower\_count == consecutive\_years - 1:  # Check if consecutive lower ranks occurred

            countries.add(data[i][0])

    return list(countries)

################################# task 8 ###################################

# Function to extract details of a specific country from the dataset

def country\_details(data, country\_name):

    # Filter rows related to the specified country and ensure valid data for index and rank

    country\_data = [row for row in data if row[0] == country\_name and row[2] and row[3]

                    and row[2] != '' and row[3] != '']

    # Check if country data is found or if data is missing/invalid

    if not country\_data:

        print(f"Country '{country\_name}' not found or missing data.")

        return None

    # Extract indexes and ranks from valid data rows

    indexes = [float(row[2]) for row in country\_data if row[2].replace('.', '', 1) != ""]

    ranks = [int(float(row[3])) for row in country\_data if row[3] != ""]

    # Check if extracted indexes or ranks are empty or invalid

    if not indexes or not ranks:

        print(f"Country '{country\_name}' has invalid data.")

        return None

    # Calculate various statistics for the country

    avg\_rank = sum(ranks) / len(ranks)

    rank\_range = (min(ranks), max(ranks))

    index\_range = (min(indexes), max(indexes))

    index\_std\_dev = (sum((index - avg\_rank) \*\* 2 for index in indexes) / len(indexes)) \*\* 0.5

    highest\_rank\_year = country\_data[ranks.index(max(ranks))][1]

    lowest\_rank\_year = country\_data[ranks.index(min(ranks))][1]

    # Construct and return a dictionary containing country details

    return {

        'Country': country\_name,

        'Average Rank': avg\_rank,

        'Rank Range': rank\_range,

        'Index Range': index\_range,

        'Standard Deviation of Indexes': index\_std\_dev,

        'Year of Highest Rank': highest\_rank\_year,

        'Year of Lowest Rank': lowest\_rank\_year

    }

if \_\_name\_\_=='\_\_main\_\_':

    pass

**main.py file**

import os

import sys

from all\_tasks import get\_top\_countries\_by\_year, top\_3\_Countries\_with\_most\_first\_positions, find\_country\_rank, list\_countries, countries\_with\_index\_above, group\_countries\_by\_rank\_ranges, countries\_with\_consecutive\_lower\_ranks, country\_details

def read\_data(file\_path):

    if not os.path.isfile(file\_path):  # check if the file is available or not

        print("File not found.")

        sys.exit()

    data = [] # we will store our data in list

    with open(file\_path, 'r') as file:   # open file in read mode

        for line in file: # iterate through each line

            row = line.strip().split(',')  # split the line at comma

            data.append(row) # append each row in the data list

    return data

    # in above code we opened file in read mode and store it as lists within list. Each list represent a row

def select\_task(file\_path):

    data = read\_data(file\_path)

    instructions = """

        Please Select from the menu to perform specific operation

        1. Top 10 happiest countries or least happiest

        2. top 3 countries that have most first positions from top and bottom

        3. Specific country with increasing its rank or decreasing its rank over specific period

        4. find list of countries

        5. countries with or above specific index value

        6. group contries contries by rank

        7. countries\_with\_consecutive\_lower\_ranks over specific period

        8. specific country details

    """

    print(instructions)

    choice = int(input("Please select 1 to 8 number to perform operations: "))

    header = data[0]

    data\_ = data[1:]

    if choice==1:

        specific\_year = input("Enter year: ")

        top\_from\_bottom = True if input("select top\_from\_bottom value(True/False)")=="True" else False

        top\_count=int(input("Enter top count: "))

        top\_countries\_data = get\_top\_countries\_by\_year(data\_, specific\_year=specific\_year, top\_from\_bottom=top\_from\_bottom, top\_count=top\_count, print\_all=False)

        top\_countries\_data

    elif choice==2:

        top\_from\_bottom = True if input("select top\_from\_bottom value(True/False)")=="True" else False

        # top\_count=int(input("Enter top count: "))

        top\_countries\_data = top\_3\_Countries\_with\_most\_first\_positions(data\_, top\_from\_bottom=top\_from\_bottom, top\_count=3)

        print(top\_countries\_data)

    elif choice==3:

        country = input("Enter country name(that are in list): ")

        period = int(input("Enter period value(int 1 to 10): "))

        find\_country\_rank(data\_, country, period)

    elif choice==4:

        dsc = True if input("descending order?(True/False): ")=="True" else False

        print(list\_countries(data\_, dsc=dsc))

    elif choice==5:

        index\_threshold = float(input("Enter threshold value(float): "))

        result\_countries = countries\_with\_index\_above(data\_, index\_threshold)

        for country, index in result\_countries:

            print(f"{country}: {index}")

    elif choice==6:

        rank\_groups = group\_countries\_by\_rank\_ranges(data\_)

        # Print the countries in each rank range for the last 5 years

        for rank\_range, countries in rank\_groups.items():

            if countries:  # Display only non-empty rank ranges

                print(f"Rank Range {rank\_range}: {countries}")

    elif choice==7:

        consecutive\_years = int(input("Enter the number of consective years: "))

        countries\_with\_consecutive\_lower = countries\_with\_consecutive\_lower\_ranks(data\_, consecutive\_years)

        print(f"Countries with at least {consecutive\_years} consecutive years of lower ranks: {countries\_with\_consecutive\_lower}")

    elif choice==8:

        country\_name = input("Enter Country name: ")

        details = country\_details(data\_, country\_name)

        if details:

            print("Country Details:")

            for key, value in details.items():

                print(f"{key}: {value}")

    else:

        print("Please select a valid choice!!")

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

    file\_path = 'world\_happiness\_index\_2013\_2023.csv'

    select\_task(file\_path)