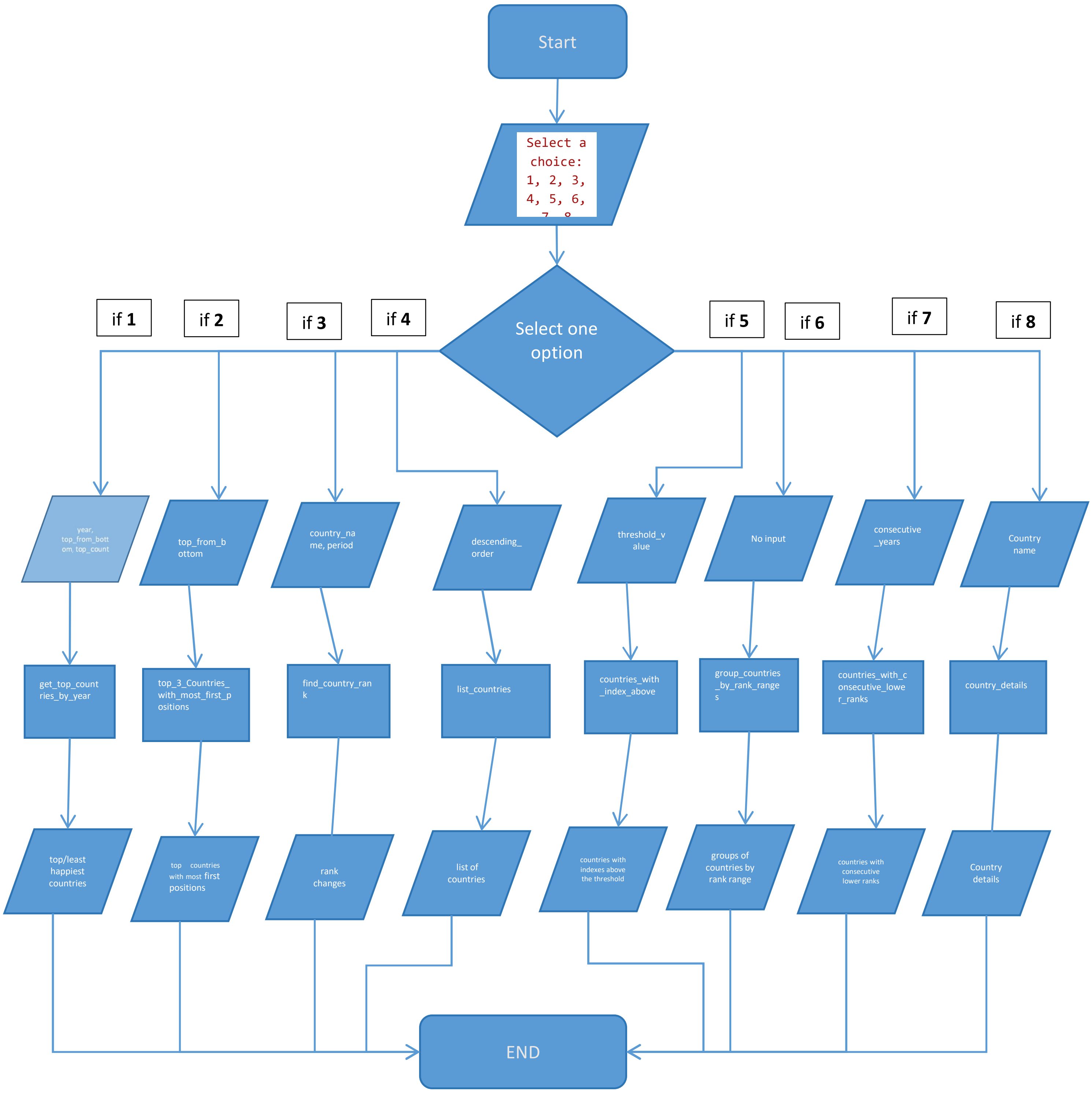
UML Activity Diagram


Start
Select a choice: 1, 2, 3, 4, 5, 6, 7, 8
Select one option
if 1
if 2
if 3
if 4
if 5
if 6
if 7
if 8
year, top_from_bottom, top_count
top_from_bottom
country_name, period
descending_order
threshold_value
No input
consecutive_years
Country name
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
top/least happiest countries
top countries with most first positions
rank changes
list of countries
countries with indexes above the threshold
groups of countries by rank range
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)
```