Self-Assessment: Please highlight where you think your report grade should be. Example below.

Criteria	Write simple algorithms using appropriate discrete data structures to solve computational problems (LO3)	Use appropriate methods to analyze the efficiency and correctness of algorithms (LO4)
Weight	**25%**	**25%**
0 – 29%	The algorithm does not solve an appropriate problem or has serious errors. There is little or no discussion of how the algorithm works. No discrete data structure has been used, or the choice of data structure is inappropriate.	The analysis is limited and seriously flawed.
30 – 39%	The algorithm solves part of an appropriate problem. There may be substantial aspects of the problem which are not attempted or explained, or errors in the solution. The explanation is unclear or missing important details about how the algorithm works.	An attempt has been made to analyze an algorithm, but appropriate methods of analysis were not used, and the results of the analysis may be incorrect or meaningless.
40 – 49%	A rudimentary algorithm solving a basic problem. There may be some errors which could be corrected with further work. There is a limited discussion of how the algorithm works. The choice of data structure is inappropriate, or unjustified.	An attempt has been made to measure the running time of the algorithm for some inputs, but the methodology is unclear, or the measurement may be inaccurate. There is a limited discussion of some other issues relating to efficiency. Analysis of the algorithm's correctness is vague, or not attempted.
50 – 59%	The algorithm solves an appropriate problem, though it may have minor errors or fail to account for special cases. There is an explanation of how the algorithm works. The choice of data structure may be inappropriate or poorly justified.	The running time of the algorithm has been measured accurately for an appropriate range of inputs, and the methodology has been explained. There is some discussion of other issues relating to efficiency. There is a basic or informal analysis of the algorithm's correctness.
60 – 69%	The algorithm correctly solves an appropriate problem. There is a clear explanation of how the algorithm works. At least one appropriate data structure has been used, and the choice has been adequately justified.	The efficiency of the algorithm has been accurately measured using an appropriate methodology, which has been explained. The measurements may include more than one metric. There is an analysis of the algorithm's correctness, which may specify pre- and post-conditions for part of the algorithm.
70 – 79%	The algorithm correctly solves a challenging problem. There is a clear explanation of how the algorithm works, and the explanation makes clear references to the relevant parts of the source code. Appropriate data structures have been used, and justification is given for each with reference to the specific problem.	The efficiency of the algorithm has been accurately measured using an appropriate methodology, with multiple metrics and a clear explanation. The asymptotic complexity of the algorithm is given. The efficiency may be compared with appropriate alternative algorithm(s). There is a formal analysis of the correctness of at least part of the algorithm.
80 - 90%	A well-designed algorithm which correctly solves a challenging problem. There is a clear, detailed explanation of how the algorithm works, with clear references to the relevant parts of the source code. Appropriate data structures have been used, and justification is given for each with reference to the specific problem.	The efficiency of the algorithm has been accurately measured using an appropriate methodology, with multiple metrics and a clear, detailed explanation. The asymptotic complexity of the algorithm is given. The efficiency has been compared with appropriate alternative algorithm(s). There is a detailed formal analysis of the correctness of the algorithm.
90 – 100%	An excellent algorithm written, explained and evaluated to the highest standards.	An excellent analysis of the efficiency, complexity and correctness of an algorithm, conducted and explained to the highest standards.

Technical Report: World Happiness Report 2019: Analyzing Top N Countries Within Specific Happiness Score Ranges and Correlating Socio-economic Factors

Author:	**Sanket Shrestha (BCU)**
Date:	**June 19, 2024**
Wordcount:	**2896**
Page count:	**25**
Confidential:	**Yes – Department Only**

Table of Contents:

Acknowledgement:

I am deeply thankful for the incredible opportunity to present my technical report on "Student Study Performance" as part of the "Data Structure and Algorithm" module. Throughout the module, I encountered numerous challenges. However, with the guidance and support of my teacher, I was able to navigate through them successfully. His advice was invaluable, and I am profoundly grateful for his assistance. Additionally, the internet tools I utilized during my research were immensely helpful in completing this project.

Executive Summary:

This report focuses on the World Happiness Report dataset to rank countries according to various macro-level factors influencing happiness. It uses proper data structures and algorithms for data sorting and selection of the data. Data structures like lists of lists and lists of objects are used which make it easy to handle the data.

The analysis reveals significant insights into the correlation between happiness and factors like GDP per capita, social support, life expectancy, freedom to make life choices, generosity, and perceptions of corruption. By ranking countries on these factors, the present study offers a clear and detailed picture of the global state of happiness. Thus, such findings emphasize the need to resolve these issues to enhance overall happiness.

Finally, this study ranks the happiness scores of the countries and identifies the key factors affecting these scores, which would be helpful for policymakers and researchers to enhance the quality of life globally.

1. Introduction:

1.1. Understanding the World Happiness Index and Its Key Factors

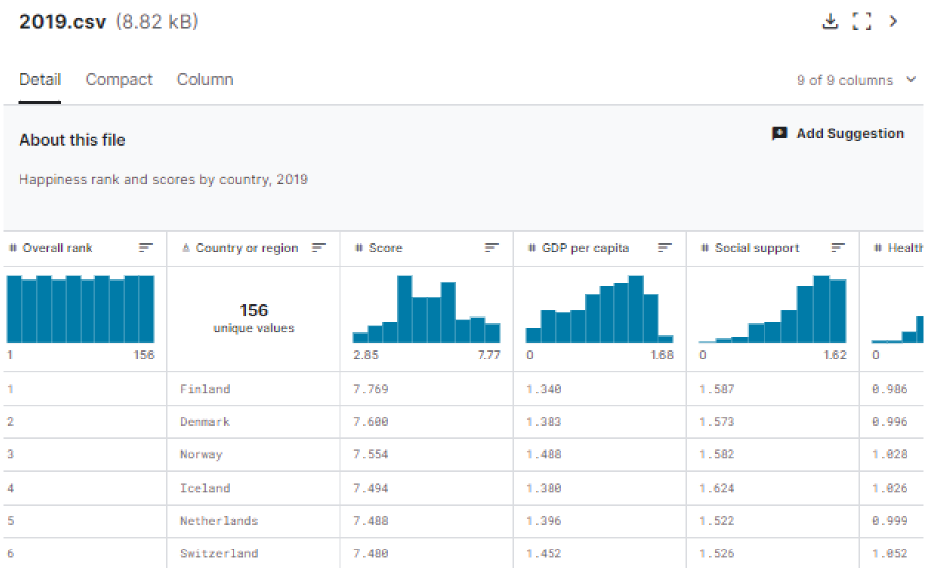
The World Happiness Report is an in-depth analysis that ranks the happiness of all nations based on the general wellbeing of their citizens. In this modern era, policymakers, scholars, and social researchers need this data to evaluate the population's well-being. From the rankings we can compare each country's performance in these categories to a fictional 'Dystopia' with the lowest scores (Carlsen, 2017).

To make concreate policy guidance, UN sustainable goals, create advocacy and awareness it is essential to rank the country based on the happiness scale and find the correlation of the seven socio-economic factors.

1.2. Objective of the report

Establishing a hierarchy of happiness-based country helps us to determine allows for comparisons between countries and regions, showcasing best practices and effective strategies for enhancing happiness and well-being along with the insights obtained from role of socio-economic and its effect on the region/country (Handayani et al., 2022).

In this report, we will work on the 2019 World Happiness Report dataset, which provides complete data for comparing and analyzing the happiness levels of countries globally (Network, 2019). We will rank 'N' highest countries in that range and then identify correlation of socio-economic factors within those countries. The objective of this report is to achieve an acceptable level of computational complexity and the quality of ranking results. Finally, we will confirm that the proposed approach is correct by applying an assertion table.



Overall rank	Country or region	Score	GDP per capita	Social support	Healt
1 – 156	156 unique values	2.85 – 7.77	0 – 1.68	0 – 1.62	0
1	Finland	7.769	1.340	1.587	0.986
2	Denmark	7.600	1.383	1.573	0.996
3	Norway	7.554	1.488	1.582	1.028
4	Iceland	7.494	1.380	1.624	1.026
5	Netherlands	7.488	1.396	1.522	0.999
6	Switzerland	7.480	1.452	1.526	1.052

Figure.1. World happiness Report file in kaggle

2. Theory:

2.1. Idea of Selecting Happiness Scores Index and it's Factors

We need to understand that Happiness score index is a vast field of study, to properly use it to make policy decisions we need to know how a certain index is obtained. Instead of working on one index at a time, skimming a range of index and finding the correlation of its factor is an effective way to understand the general idea of happiness in the region/country.

We do this by taking the least and highest index for a range and sorting countries on it then forking its socio-economic factors to find the correlation between the attributes. This helps us to provide a clear understanding of the distribution of happiness scores.

2.2. Data Structure and Algorithm used for Ranking Countries

This section provides a brief overview of the algorithms and data structures required for the proposed algorithm.

2.2.1. Data Structure:

1. **Class**: Classes are also data structures in Python, but they are more advanced than basic dictionaries, tuples, and lists (Mailund, 2021). It consists of all the socio-economic attributes of the dataset. Using a class makes it easy to represent each country and is reusabile. Ultimately it provides a clear structure and increases the readability of the code.

2. **List**: To efficiently store and manipulate the data, a list data structure is used here as an instance of the "country" class. Operations such as sorting and filtering in performed on it. Lists are used for their simplicity and efficiency in storing and manipulating sequences of data. They allow random access while being easy to traverse, which makes it useful for reading.

3. **Dictionary**: To efficiently store and manipulate the data, a list data structure is used here as an instance of the "country" class. It is particularly useful for storing and accessing correlation coefficients, where each socio-economic factor in the data is a key, and its correlation with the happiness score is the value. Thus, allowing an easy and efficient retrieval of correlation data for analysis and visualization.

2.2.2. Algorithm:

1. **Iteration and Collection:** It iterates the CSV file of the dataset and stores them in class. This algorithm reads the CSV file line by line and converts each row into a Country object. It ensures efficient data reading and conversion to appropriate types. The same is used to print multiple data in the collab file.

2. **List-Comprehension**: here list comprehension is used just like for loop to filer countries based on their happiness score. It is used as it provides better performance than "For" loops due to their optimized internal implementation in python, leading to readability and simplicity of code (Wadler, 1992). They have linear time complexity (O(n)), making them efficient for small to medium datasets.

3. **Quick Sort:** it is chosen for its average-case time complexity of O(n log n) and in-place sorting principle. It is efficient for large datasets. Although it has a worst-case time complexity of O(n^2), the implementation mitigates this by choosing the middle element as the pivot. Quicksort is a divide-and-conquer algorithm, making it highly efficient for sorting operations in this context.

3. Implementation:

3.1. Proposed Algorithm to Solve the Problem

This section introduces an advanced analysis and visualization tool for global happiness data based on the 2019 World Happiness Report (dataset '2019.csv'). This offers an effective approach to understanding factors influencing happiness scores across different countries.

The code utilizes object-oriented programming (OOP) with "country" class to encapsulate data attributes such as rank, country name, happiness score, GDP, social support, life expectancy, freedom, generosity, and corruption perception. These attributes are vital for conducting detailed analyses and visualizations.

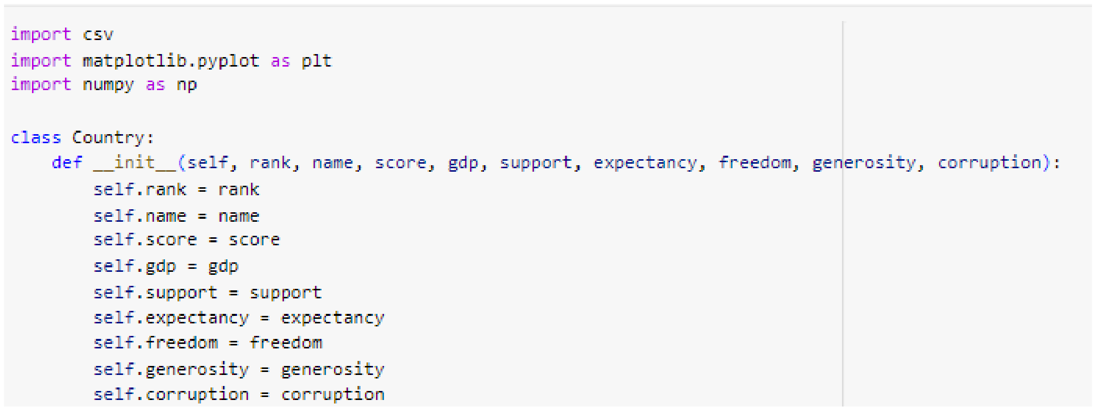
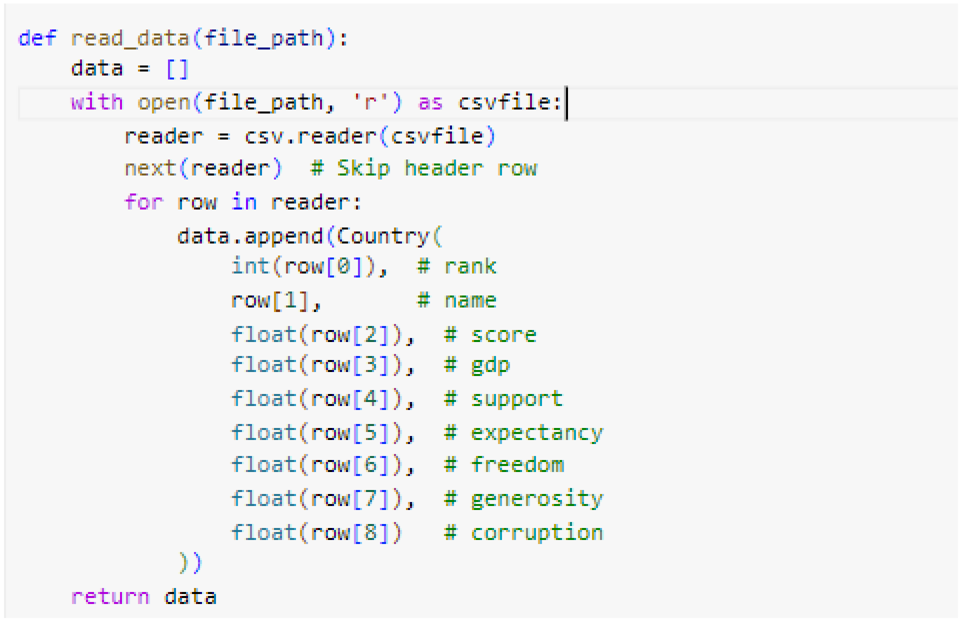

```python
import csv
import matplotlib.pyplot as plt
import numpy as np

class Country:
    def __init__(self, rank, name, score, gdp, support, expectancy, freedom, generosity, corruption):
        self.rank = rank
        self.name = name
        self.score = score
        self.gdp = gdp
        self.support = support
        self.expectancy = expectancy
        self.freedom = freedom
        self.generosity = generosity
        self.corruption = corruption
```

Figure.2. Code snippet of Country Class

3.1.1. Reading Data from CSV file:

Here for this portion, read_data(file_path) is designed to read data from a CSV file and convert it into a list of Country. It iterates through each row to create Country objects based on the data. Then initializes an empty list where objects will be stored after being read.

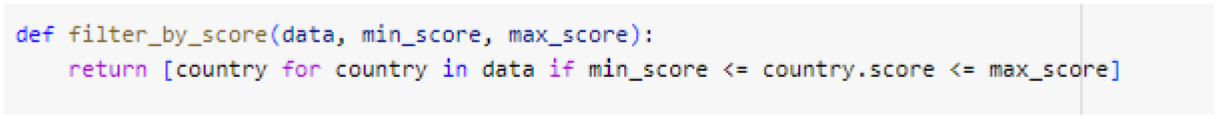


```python
def read_data(file_path):
    data = []
    with open(file_path, 'r') as csvfile:
        reader = csv.reader(csvfile)
        next(reader)  # Skip header row
        for row in reader:
            data.append(Country(
                int(row[0]),  # rank
                row[1],       # name
                float(row[2]),  # score
                float(row[3]),  # gdp
                float(row[4]),  # support
                float(row[5]),  # expectancy
                float(row[6]),  # freedom
                float(row[7]),  # generosity
                float(row[8])   # corruption
            ))
    return data
```

Figure.3. Code Snippet of Reading CSV file code

3.1.2. Filtering the countries within a certain Happiness Index:

Here, filter_by_score is used to pick out specific countries from a list based on their scores. It uses a list comprehension in Python that checks each country's score against a range with min_score and max_score. If a country's score falls within that range, it gets added to a new list.

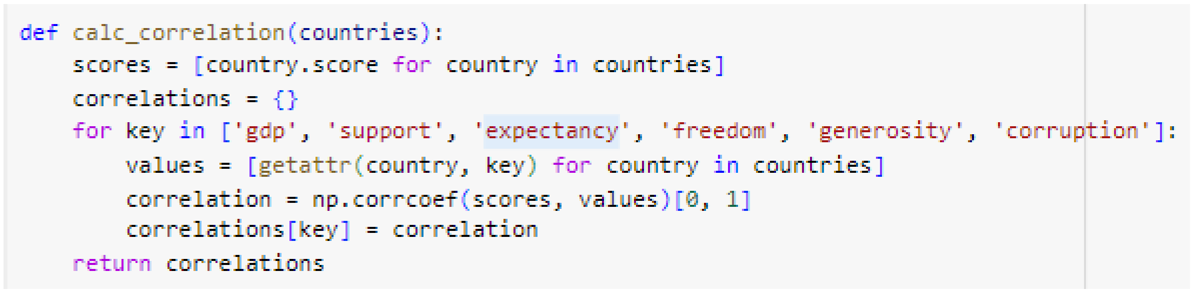


```python
def filter_by_score(data, min_score, max_score):
    return [country for country in data if min_score <= country.score <= max_score]
```

Figure.4. Code Snippet of Filtering Code

3.1.3. Calculating Correlation:

Here, calc_correlation helps us find out how different socio-economic aspects relate to a country's overall score. It gathers all the scores of the countries into a list called scores then calculates the correlation between the scores and these aspect-specific values using a mathematical method. The results are stored in a dictionary , where each aspect is paired with its correlation value.

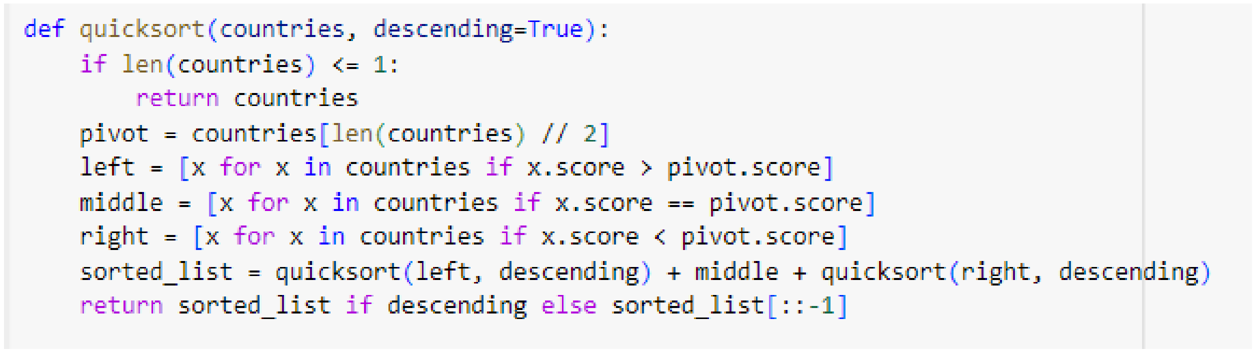


```
def calc_correlation(countries):
    scores = [country.score for country in countries]
    correlations = {}
    for key in ['gdp', 'support', 'expectancy', 'freedom', 'generosity', 'corruption']:
        values = [getattr(country, key) for country in countries]
        correlation = np.corrcoef(scores, values)[0, 1]
        correlations[key] = correlation
    return correlations
```

Figure.5. Code Snippet of Calculating Correlation

3.1.4. Use of Quicksort for Ranking:

Here, quicksort algorithm is used to rank the countries, it picks a middle point(pivot) in the list of countries based on their scores that helps in dividing the countries into 3 groups (higher, lower and equal to pivot). It then repeats this process for the groups with higher and lower scores, sorting them within each group. This is called recursion. Finally, it combines all the sorted groups back together and reverses the order of sorting.



```
def quicksort(countries, descending=True):
    if len(countries) <= 1:
        return countries
    pivot = countries[len(countries) // 2]
    left = [x for x in countries if x.score > pivot.score]
    middle = [x for x in countries if x.score == pivot.score]
    right = [x for x in countries if x.score < pivot.score]
    sorted_list = quicksort(left, descending) + middle + quicksort(right, descending)
    return sorted_list if descending else sorted_list[::-1]
```

Figure.6. Code Snippet of Use of Quick Sort

3.2. Alternative Problem Solution:

Functionality	**Current Data Structure**	**Alternative Data Structure**	**Advantages of Current**	**Disadvantages of Current**	**Advantages of Alternative**	**Disadvantages of Alternative**
Storing Country Data	Class	Dictionary	Encapsulation, reusability, easy to extend.	Overhead of object creation and attribute access.	Simple key-value access, flexible structure.	Less strict structure, harder to maintain consistency.
Storing List of Countries	List	Linked List	Fast random access, simple to iterate and use.	Inefficient for frequent insertions and deletions.	Efficient insertions and deletions, dynamic size.	No direct random access, more memory overhead.
Filtering Data	List Comprehension	Generator Expression	fast iteration for small datasets.	Entire list needs to be created in memory, can be inefficient for large datasets.	Memory efficient, elements generated on-the-fly.	Only allows single-pass iteration, can be slower for small datasets.
Correlation Calculation	Dictionary	Pandas Data Frame	Simple key-value storage, easy to access and iterate.	Limited functionality for statistical operations and data manipulation.	Rich functionality for statistical operations, easy data manipulation and visualization.	More complex to set up and use, additional dependencies.
Sorting	Quicksort	Merge Sort	Average case performance (O(n log n)), in-place sorting (memory efficient).	Worst case performance ($O(n^2)$), not stable.	Stable sorting (preserves order of equal elements), guaranteed O(n log n) performance.	Requires additional memory (not in-place).

Table.1. Alternative Possible Algorithm that could be used

4. Performance Evaluation:

4.1. Time Complexity of the Problem:

1. **Reading the Data:**
   Time Complexity: O(n), where "n" is the number of rows in the CSV file. Each row is processed once to create a Country object.

2. **Filtering the Countries:**
   Time Complexity: O(n), where "n" is the number of countries in data. It iterates through the list of countries once to filter out countries based on score.

3. **Calculating Correlation:**
   Time Complexity: O(n·m), where "n" is the number of countries and "m" is the number of indicators. This is because it calculates correlation for each indicator with respect to the scores.

4. **Quick sort:**
   Time Complexity: O(nlogn) on average, where n is the number of countries. This is the complexity of the quicksort algorithm, partitioning the list based on scores.

We take all the obtained time complexity of our algorithm, among which the most dominating are:

- O(n.m) from calculating correlation
- O(nlogn) from quicksort

Since, O(n,m) dominates O(nlogn), therefore the total time complexity T(n,m) is,

$$T(n, m) = O(n \cdot m) + O(n\log n) = O(n \cdot m)$$

4.1.1. Time Complexity in Different Metric:

Function	Best Case Complexity	Worst Case Complexity	Average Case Complexity
read_data(file_path)	O(N)	O(N)	O(N)
filter_by_score(data, min_score, max_score)	O(1)	O(N)	O(N)
calc_correlation(countries)	O(N)	O(N.M)	O(N.M)
quicksort(countries, descending=True)	O(n \log n)	O(n^2)	O(n \log n)
print_countries(countries, num_to_print)	O(1)	O(num_to_print)	O(num_to_print)
print_correlations(correlations)	O(M)	O(M)	O(M)

Table.2. Time Complexity Comparison

4.1.2. Comparison of Quicksort with other algorithms:

Time Complexity	Algorithm	Disadvantages Compared to Quicksort
Merge Sort	O(nLogN)	• Typically requires additional space for recursive calls (stack space). • More complex implementation
Counting Sort	O(N)	• Limited to a specific range of integer inputs • Requires additional space proportional to the range of input values.
Bubble Sort, Selection Sort	O (N.N)	• Quadratic time complexity, slower for large dataset • Simple to implement but less efficient

Table.3. Comparison of other Algorithms with Quick Sort

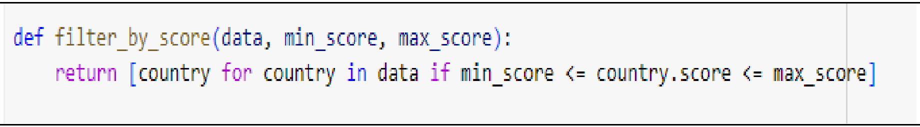
After looking at the above table, we find that the best alternative of quicksort is merge sort but, it is only useful for dataset with huge volume of data. Thus, quicksort is the best.

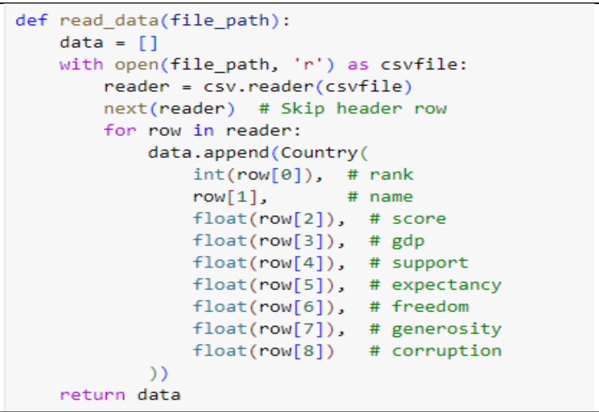
4.2. Space Complexity:

The space complexity analysis of the code shows how each function uses memory. Functions like reading data and quicksort use O(n) space, where n is the number of countries while filtering uses O(k) space, with k being the number of filtered countries.

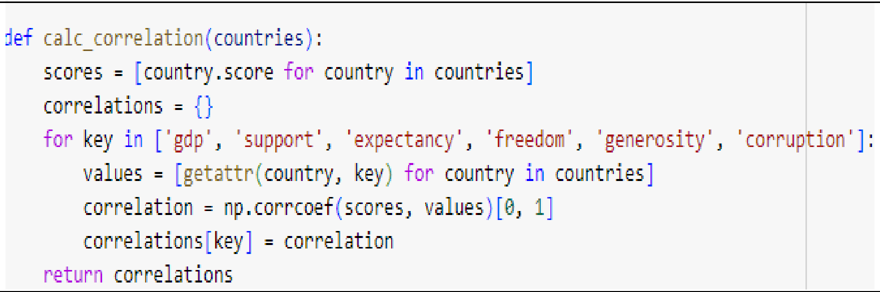
Thus, the Total Space Complexity = O(N)

4.3. Assertion Table:

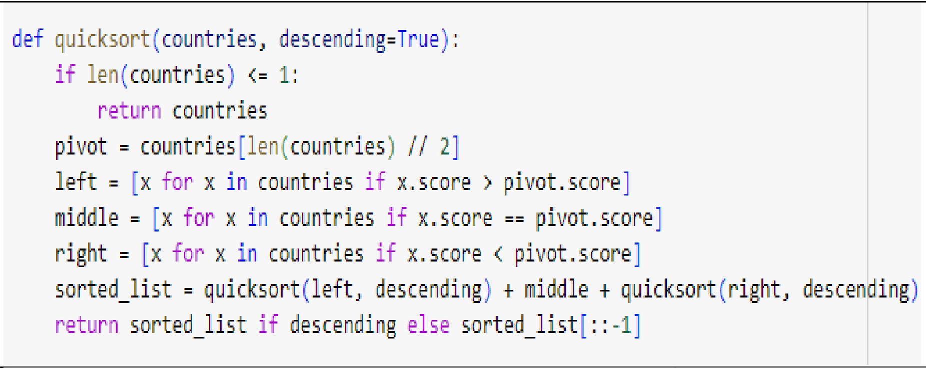
Assertion	Code Snippet	Functionality
Filtering Countries	 `def filter_by_score(data, min_score, max_score):` `    return [country for country in data if min_score <= country.score <= max_score]`	Working

Reading the Data and storing in list		Working

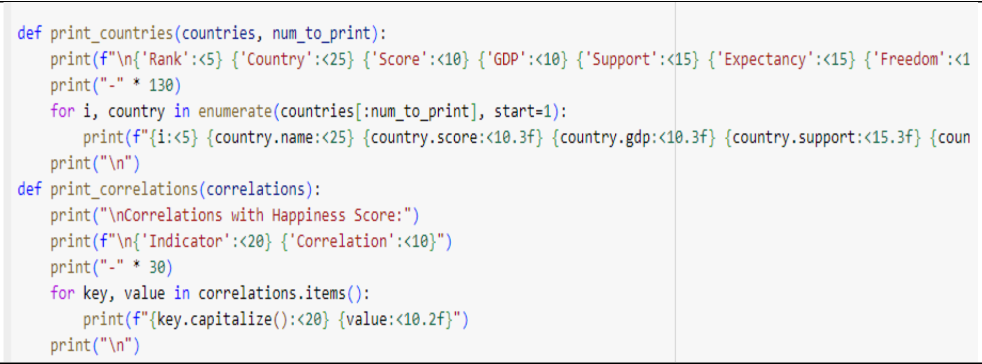
```python
def read_data(file_path):
    data = []
    with open(file_path, 'r') as csvfile:
        reader = csv.reader(csvfile)
        next(reader)  # Skip header row
        for row in reader:
            data.append(Country(
                int(row[0]),  # rank
                row[1],       # name
                float(row[2]),  # score
                float(row[3]),  # gdp
                float(row[4]),  # support
                float(row[5]),  # expectancy
                float(row[6]),  # freedom
                float(row[7]),  # generosity
                float(row[8])   # corruption
            ))
    return data
```

Correlation calculation		Working

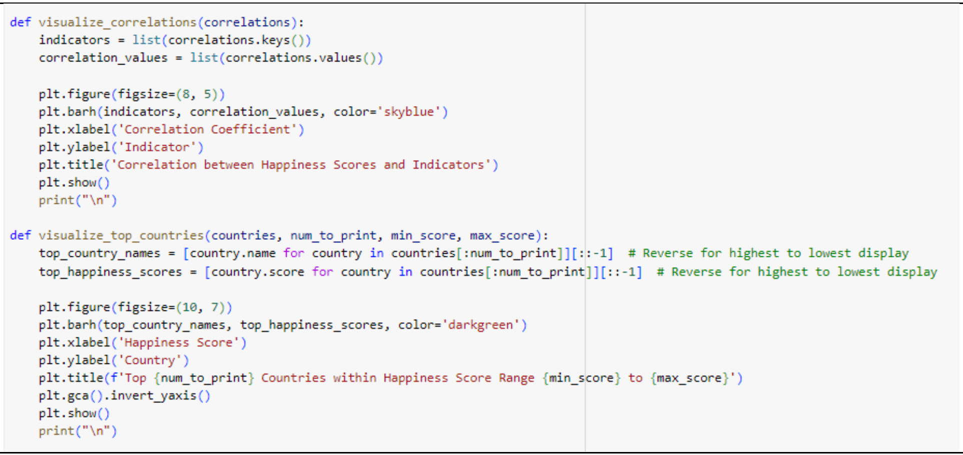
```python
def calc_correlation(countries):
    scores = [country.score for country in countries]
    correlations = {}
    for key in ['gdp', 'support', 'expectancy', 'freedom', 'generosity', 'corruption']:
        values = [getattr(country, key) for country in countries]
        correlation = np.corrcoef(scores, values)[0, 1]
        correlations[key] = correlation
    return correlations
```

Quick Sort		Working

```python
def quicksort(countries, descending=True):
    if len(countries) <= 1:
        return countries
    pivot = countries[len(countries) // 2]
    left = [x for x in countries if x.score > pivot.score]
    middle = [x for x in countries if x.score == pivot.score]
    right = [x for x in countries if x.score < pivot.score]
    sorted_list = quicksort(left, descending) + middle + quicksort(right, descending)
    return sorted_list if descending else sorted_list[::-1]
```

Printing the data		Working

```python
def print_countries(countries, num_to_print):
    print(f"\n{'Rank':<5} {'Country':<25} {'Score':<10} {'GDP':<10} {'Support':<15} {'Expectancy':<15} {'Freedom':<1
    print("-" * 130)
    for i, country in enumerate(countries[:num_to_print], start=1):
        print(f"{i:<5} {country.name:<25} {country.score:<10.3f} {country.gdp:<10.3f} {country.support:<15.3f} {coun
    print("\n")
def print_correlations(correlations):
    print("\nCorrelations with Happiness Score:")
    print(f"\n{'Indicator':<20} {'Correlation':<10}")
    print("-" * 30)
    for key, value in correlations.items():
        print(f"{key.capitalize():<20} {value:<10.2f}")
    print("\n")
```

Visualizing Data		Working

```
def visualize_correlations(correlations):
    indicators = list(correlations.keys())
    correlation_values = list(correlations.values())

    plt.figure(figsize=(8, 5))
    plt.barh(indicators, correlation_values, color='skyblue')
    plt.xlabel('Correlation Coefficient')
    plt.ylabel('Indicator')
    plt.title('Correlation between Happiness Scores and Indicators')
    plt.show()
    print("\n")

def visualize_top_countries(countries, num_to_print, min_score, max_score):
    top_country_names = [country.name for country in countries[:num_to_print]][::-1]  # Reverse for highest to lowest display
    top_happiness_scores = [country.score for country in countries[:num_to_print]][::-1]  # Reverse for highest to lowest display

    plt.figure(figsize=(10, 7))
    plt.barh(top_country_names, top_happiness_scores, color='darkgreen')
    plt.xlabel('Happiness Score')
    plt.ylabel('Country')
    plt.title(f'Top {num_to_print} Countries within Happiness Score Range {min_score} to {max_score}')
    plt.gca().invert_yaxis()
    plt.show()
    print("\n")
```

Table.4. Assertion Table

5. Conclusion:

In conclusion, this report effectively sorts countries based on their happiness scores using relevant algorithms and data structures. The use of Quicksort for sorting was effective for managing the data set. The analysis also confirms the importance of socio-economic in influencing happiness. These insights offer valuable knowledge for policymakers and stakeholders aiming to improve well-being globally. More qualitative studies may be conducted in the future to investigate stronger links and more accurate forecast of the causes of happiness across different countries.

6. References

Carlsen, L. (2017) ‘Happiness as a sustainability factor. the world happiness index: A posetic-based data analysis’, *Sustainability Science*, 13(2), pp. 549–571. doi:10.1007/s11625-017-0482-9.

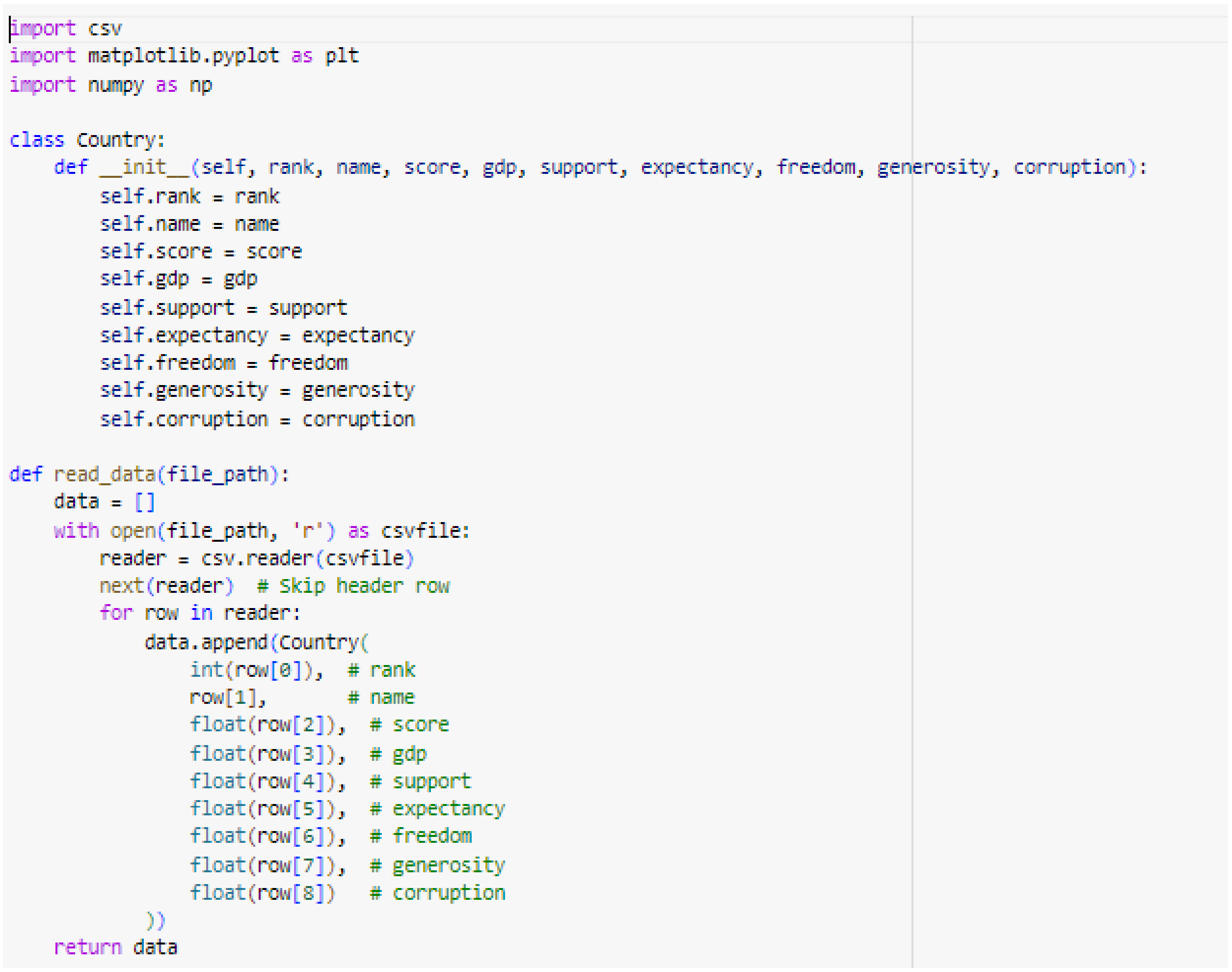
Handayani, D.O., Lubis, M. and Ridho Lubis, A. (2022) ‘Prediction analysis of the happiness ranking of countries based on macro level factors’, *IAES International Journal of Artificial Intelligence (IJ-AI)*, 11(2), p. 666. doi:10.11591/ijai.v11.i2.pp666-678.

Mailund, T. (2021) ‘Data Structures, objects, and classes’, *Introduction to Computational Thinking*, pp. 317–367. doi:10.1007/978-1-4842-7077-6_11.

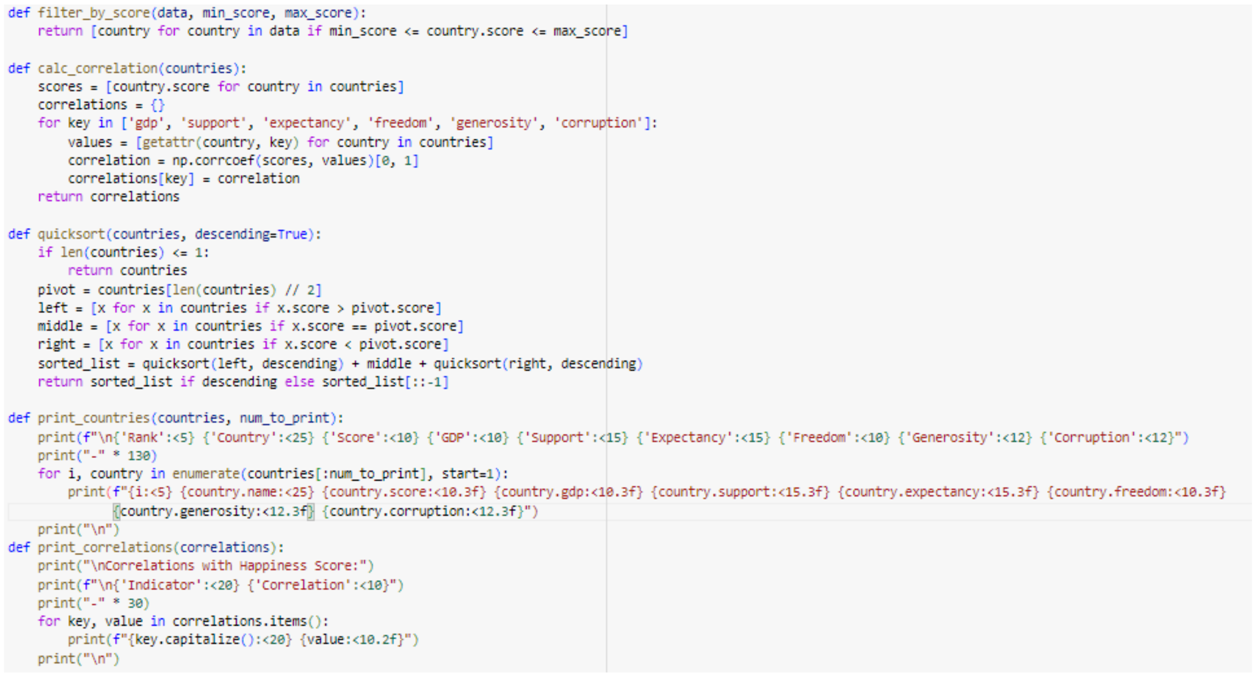
Network, S.D.S. (2019) *World happiness report*, *Kaggle*. Available at: https://www.kaggle.com/datasets/unsdsn/world-happiness?select=2019.csv (Accessed: 19 June 2024).

Wadler, P. (1992) ‘Comprehending monads’, *Mathematical Structures in Computer Science*, 2(4), pp. 461–493. doi:10.1017/s0960129500001560.

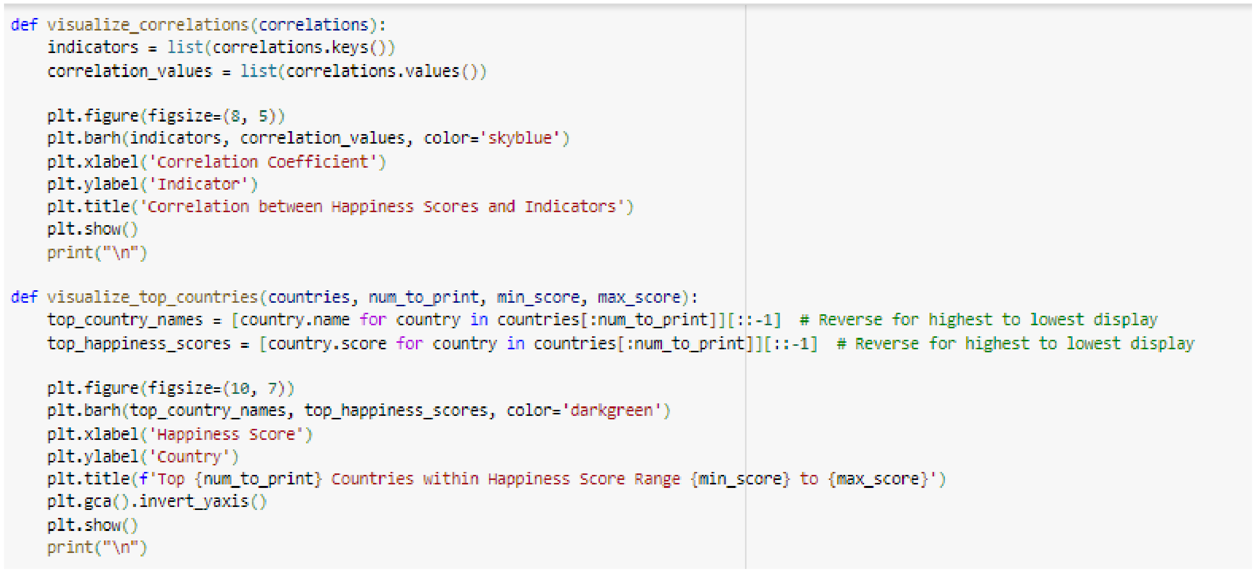
Appendix A: Code of the system:



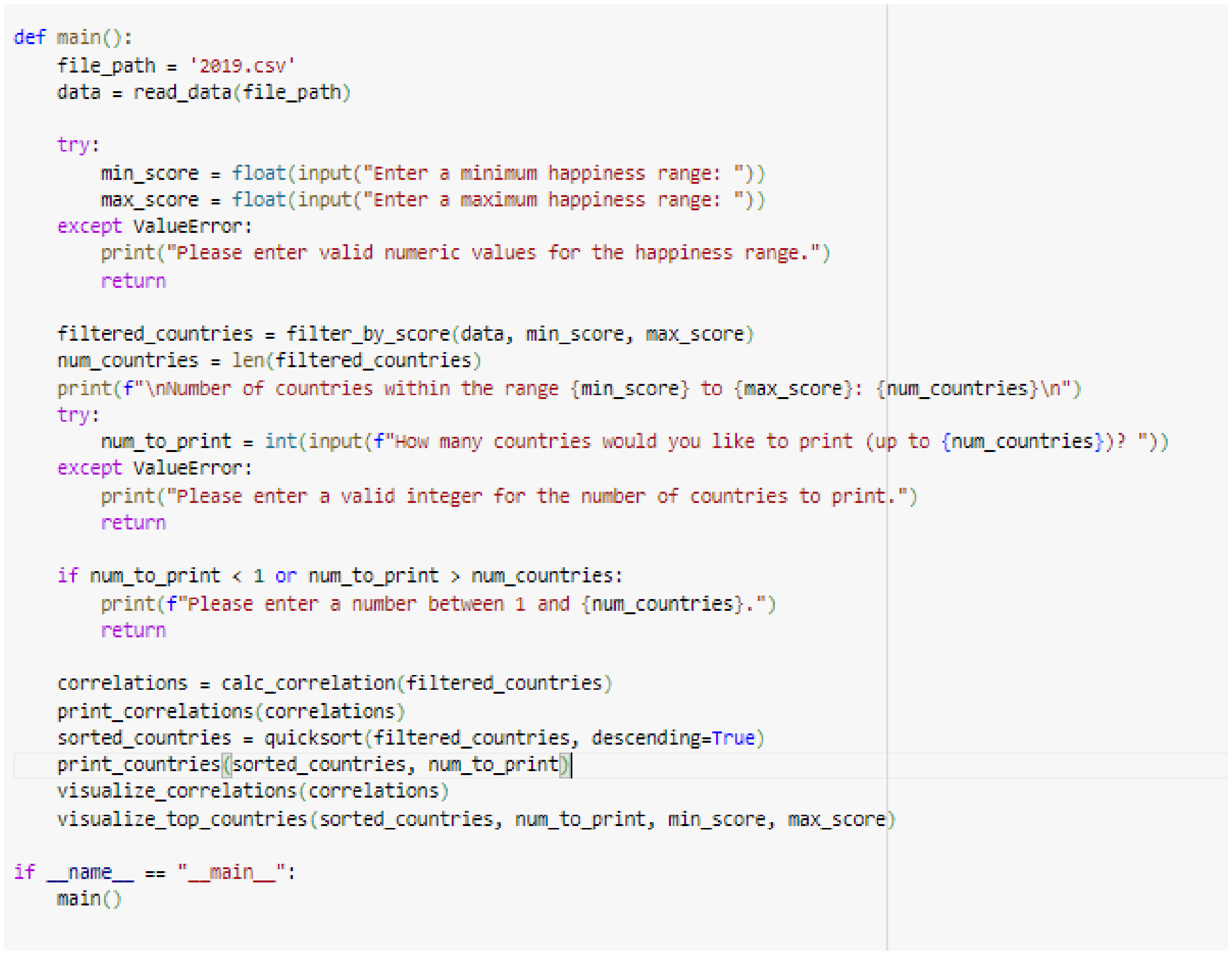
```python
import csv
import matplotlib.pyplot as plt
import numpy as np

class Country:
    def __init__(self, rank, name, score, gdp, support, expectancy, freedom, generosity, corruption):
        self.rank = rank
        self.name = name
        self.score = score
        self.gdp = gdp
        self.support = support
        self.expectancy = expectancy
        self.freedom = freedom
        self.generosity = generosity
        self.corruption = corruption

def read_data(file_path):
    data = []
    with open(file_path, 'r') as csvfile:
        reader = csv.reader(csvfile)
        next(reader)  # Skip header row
        for row in reader:
            data.append(Country(
                int(row[0]),  # rank
                row[1],       # name
                float(row[2]),  # score
                float(row[3]),  # gdp
                float(row[4]),  # support
                float(row[5]),  # expectancy
                float(row[6]),  # freedom
                float(row[7]),  # generosity
                float(row[8])   # corruption
            ))
    return data
```



```python
def filter_by_score(data, min_score, max_score):
    return [country for country in data if min_score <= country.score <= max_score]

def calc_correlation(countries):
    scores = [country.score for country in countries]
    correlations = {}
    for key in ['gdp', 'support', 'expectancy', 'freedom', 'generosity', 'corruption']:
        values = [getattr(country, key) for country in countries]
        correlation = np.corrcoef(scores, values)[0, 1]
        correlations[key] = correlation
    return correlations

def quicksort(countries, descending=True):
    if len(countries) <= 1:
        return countries
    pivot = countries[len(countries) // 2]
    left = [x for x in countries if x.score > pivot.score]
    middle = [x for x in countries if x.score == pivot.score]
    right = [x for x in countries if x.score < pivot.score]
    sorted_list = quicksort(left, descending) + middle + quicksort(right, descending)
    return sorted_list if descending else sorted_list[::-1]

def print_countries(countries, num_to_print):
    print(f"\n{'Rank':<5} {'Country':<25} {'Score':<10} {'GDP':<10} {'Support':<15} {'Expectancy':<15} {'Freedom':<10} {'Generosity':<12} {'Corruption':<12}")
    print("-" * 130)
    for i, country in enumerate(countries[:num_to_print], start=1):
        print(f"{i:<5} {country.name:<25} {country.score:<10.3f} {country.gdp:<10.3f} {country.support:<15.3f} {country.expectancy:<15.3f} {country.freedom:<10.3f}
              {country.generosity:<12.3f} {country.corruption:<12.3f}")
    print("\n")
def print_correlations(correlations):
    print("\nCorrelations with Happiness Score:")
    print(f"\n{'Indicator':<20} {'Correlation':<10}")
    print("-" * 30)
    for key, value in correlations.items():
        print(f"{key.capitalize():<20} {value:<10.2f}")
    print("\n")
```



```python
def visualize_correlations(correlations):
    indicators = list(correlations.keys())
    correlation_values = list(correlations.values())

    plt.figure(figsize=(8, 5))
    plt.barh(indicators, correlation_values, color='skyblue')
    plt.xlabel('Correlation Coefficient')
    plt.ylabel('Indicator')
    plt.title('Correlation between Happiness Scores and Indicators')
    plt.show()
    print("\n")

def visualize_top_countries(countries, num_to_print, min_score, max_score):
    top_country_names = [country.name for country in countries[:num_to_print]][::-1]  # Reverse for highest to lowest display
    top_happiness_scores = [country.score for country in countries[:num_to_print]][::-1]  # Reverse for highest to lowest display

    plt.figure(figsize=(10, 7))
    plt.barh(top_country_names, top_happiness_scores, color='darkgreen')
    plt.xlabel('Happiness Score')
    plt.ylabel('Country')
    plt.title(f'Top {num_to_print} Countries within Happiness Score Range {min_score} to {max_score}')
    plt.gca().invert_yaxis()
    plt.show()
    print("\n")
```



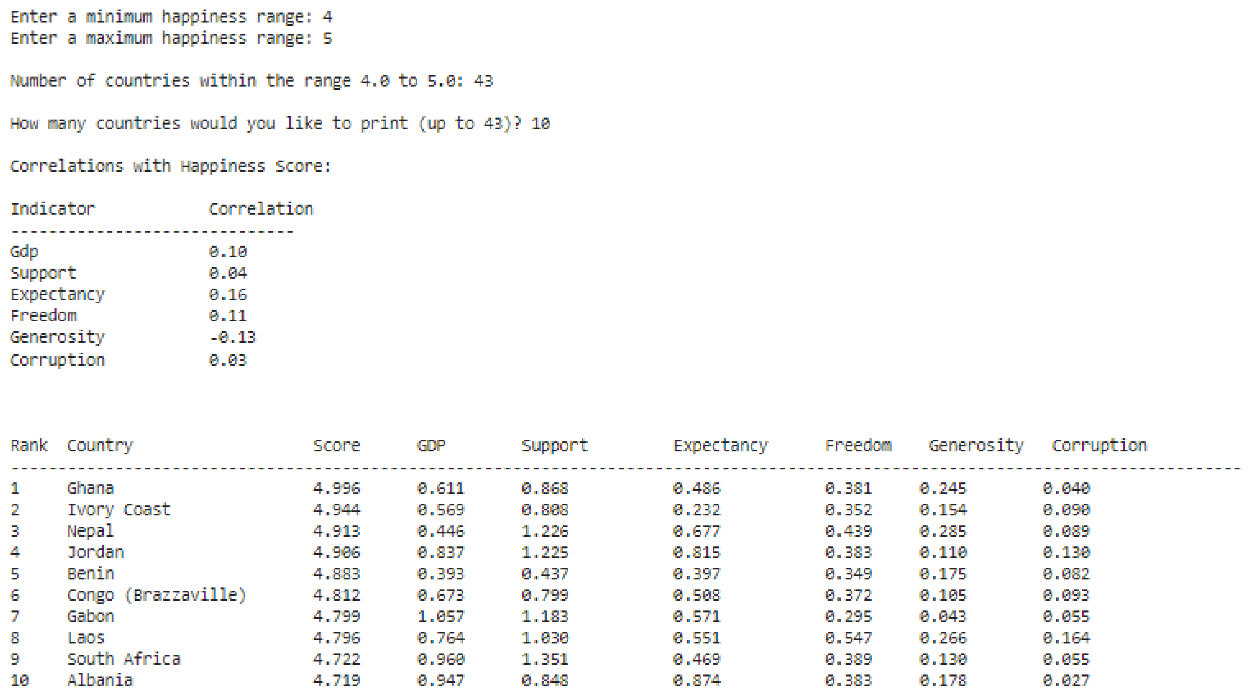
```python
def main():
    file_path = '2019.csv'
    data = read_data(file_path)

    try:
        min_score = float(input("Enter a minimum happiness range: "))
        max_score = float(input("Enter a maximum happiness range: "))
    except ValueError:
        print("Please enter valid numeric values for the happiness range.")
        return

    filtered_countries = filter_by_score(data, min_score, max_score)
    num_countries = len(filtered_countries)
    print(f"\nNumber of countries within the range {min_score} to {max_score}: {num_countries}\n")
    try:
        num_to_print = int(input(f"How many countries would you like to print (up to {num_countries})? "))
    except ValueError:
        print("Please enter a valid integer for the number of countries to print.")
        return

    if num_to_print < 1 or num_to_print > num_countries:
        print(f"Please enter a number between 1 and {num_countries}.")
        return

    correlations = calc_correlation(filtered_countries)
    print_correlations(correlations)
    sorted_countries = quicksort(filtered_countries, descending=True)
    print_countries(sorted_countries, num_to_print)
    visualize_correlations(correlations)
    visualize_top_countries(sorted_countries, num_to_print, min_score, max_score)

if __name__ == "__main__":
    main()
```

Appendix B: Output and Data Sample

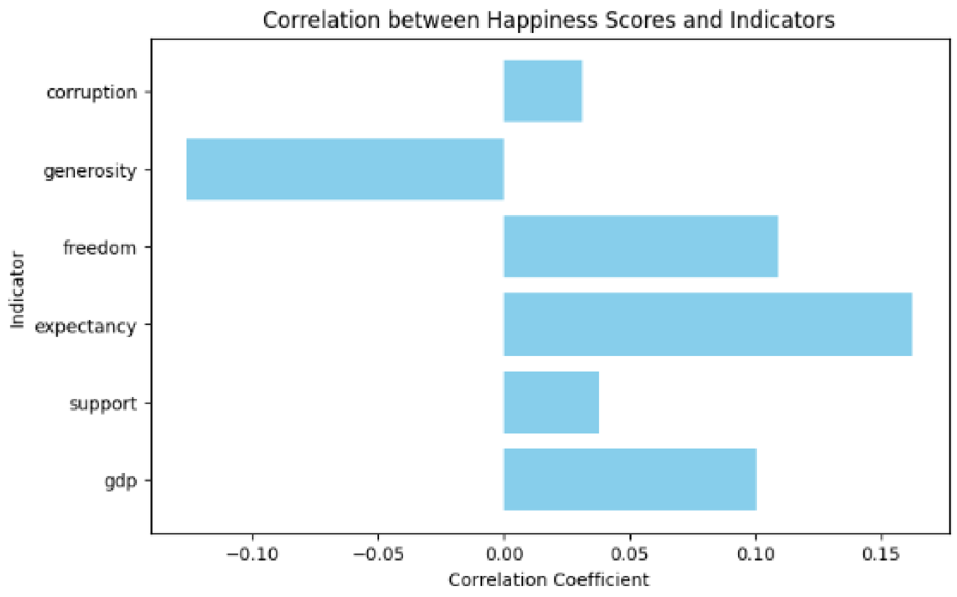
Data Sample: a sample of data used in the project,

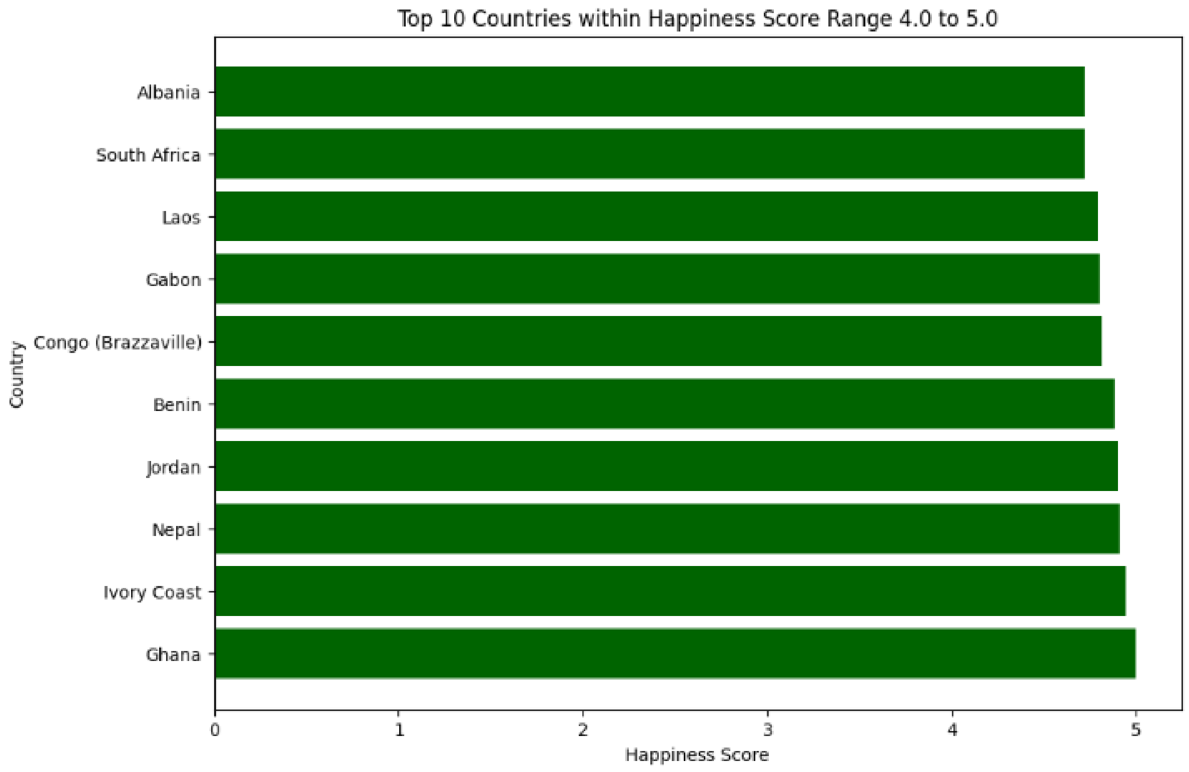
Rank	Countries	Happiness Score
1	Finland	7.769
2	Denmark	7.600
3	Norway	7.554
…	…	…
150	South Sudan	2.853

Output case 1: For happiness Score of 10 countries with index between 4 and 5.

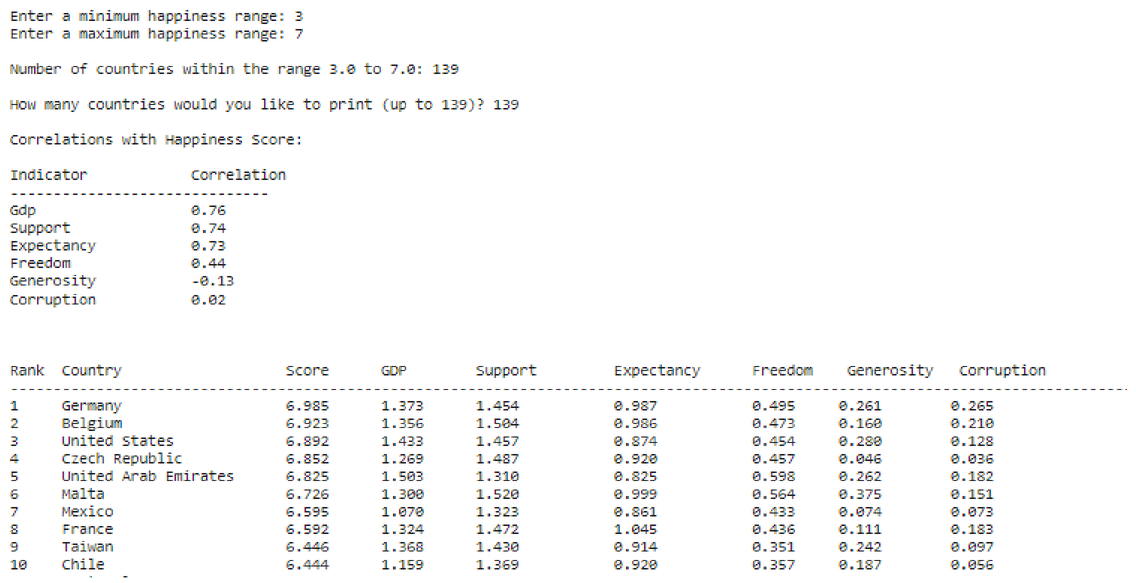


```
Enter a minimum happiness range: 4
Enter a maximum happiness range: 5

Number of countries within the range 4.0 to 5.0: 43

How many countries would you like to print (up to 43)? 10

Correlations with Happiness Score:

Indicator           Correlation
------------------------------
Gdp                 0.10
Support             0.04
Expectancy          0.16
Freedom             0.11
Generosity          -0.13
Corruption          0.03


Rank  Country                   Score     GDP       Support        Expectancy     Freedom    Generosity   Corruption
--------------------------------------------------------------------------------------------------------------------------
1     Ghana                     4.996     0.611     0.868          0.486          0.381      0.245        0.040
2     Ivory Coast               4.944     0.569     0.808          0.232          0.352      0.154        0.090
3     Nepal                     4.913     0.446     1.226          0.677          0.439      0.285        0.089
4     Jordan                    4.906     0.837     1.225          0.815          0.383      0.110        0.130
5     Benin                     4.883     0.393     0.437          0.397          0.349      0.175        0.082
6     Congo (Brazzaville)       4.812     0.673     0.799          0.508          0.372      0.105        0.093
7     Gabon                     4.799     1.057     1.183          0.571          0.295      0.043        0.055
8     Laos                      4.796     0.764     1.030          0.551          0.547      0.266        0.164
9     South Africa              4.722     0.960     1.351          0.469          0.389      0.130        0.055
10    Albania                   4.719     0.947     0.848          0.874          0.383      0.178        0.027
```

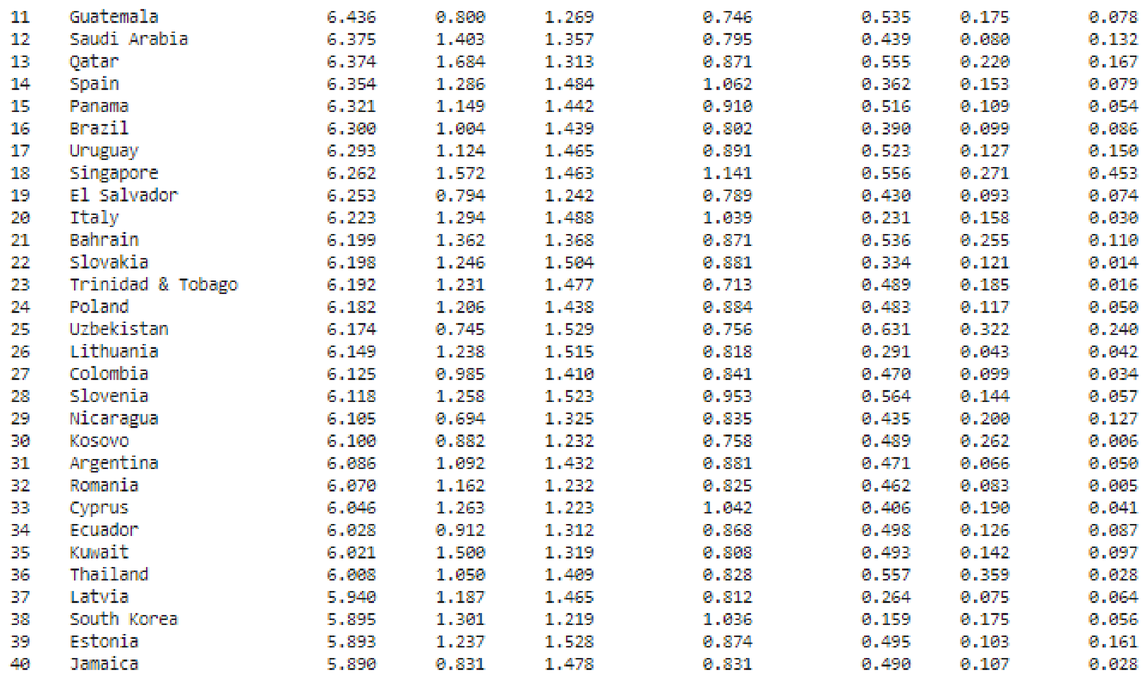

Correlation between Happiness Scores and Indicators
corruption
generosity
freedom
expectancy
support
gdp
Indicator
−0.10
−0.05
0.00
0.05
0.10
0.15
Correlation Coefficient


Top 10 Countries within Happiness Score Range 4.0 to 5.0
Albania
South Africa
Laos
Gabon
Congo (Brazzaville)
Benin
Jordan
Nepal
Ivory Coast
Ghana
Country
0
1
2
3
4
5
Happiness Score

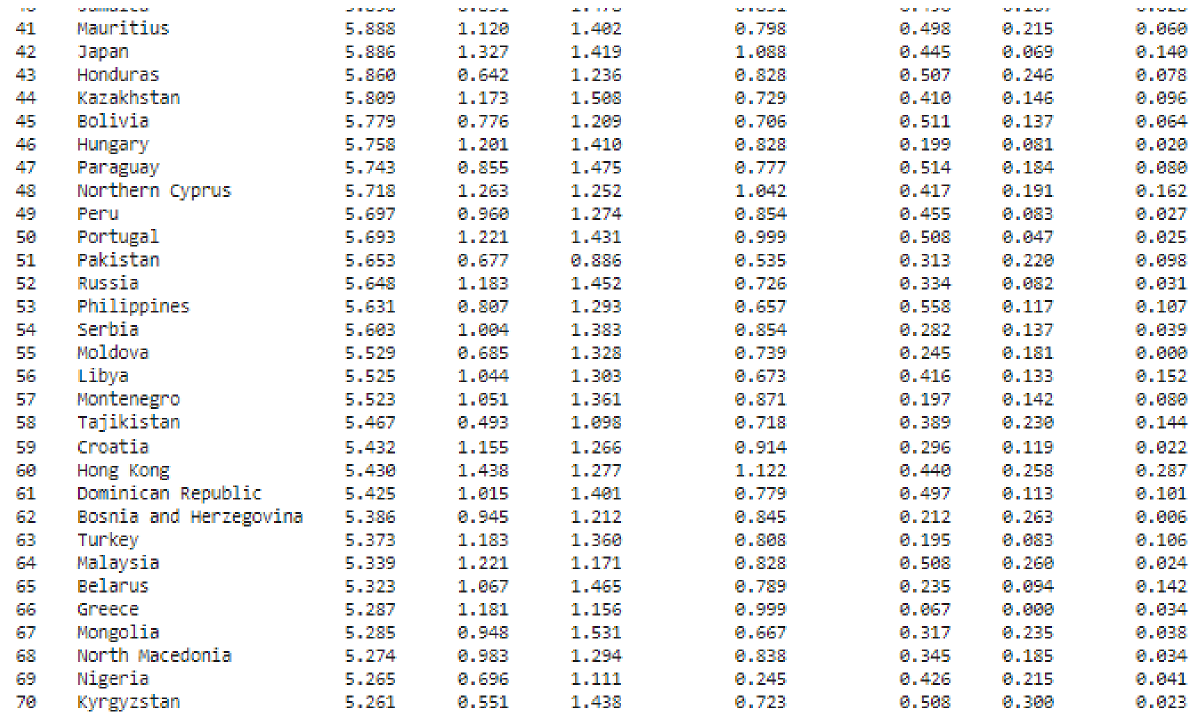
Output case 2: For happiness Score of 139 countries with index between 1 and 7.



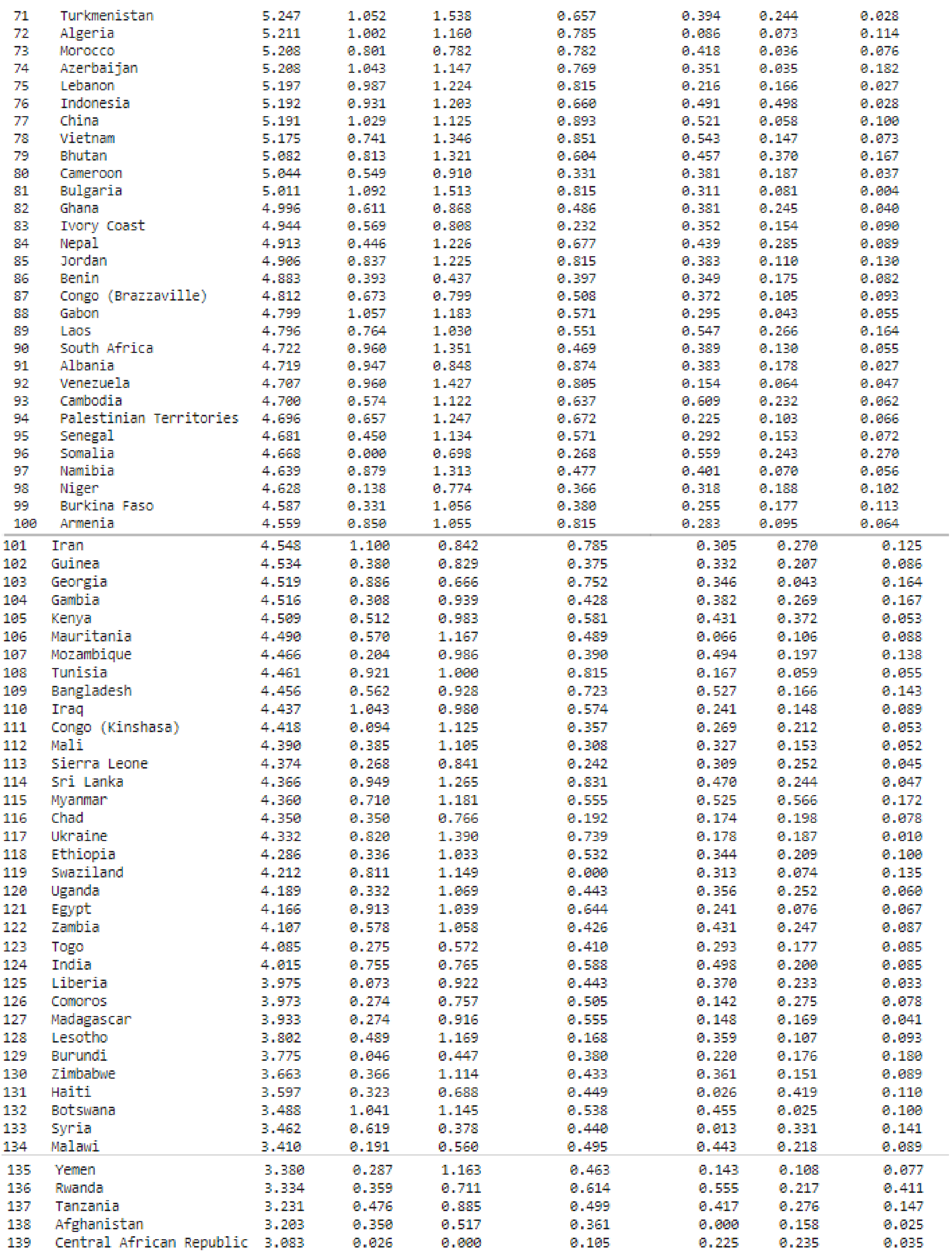
```
Enter a minimum happiness range: 3
Enter a maximum happiness range: 7

Number of countries within the range 3.0 to 7.0: 139

How many countries would you like to print (up to 139)? 139

Correlations with Happiness Score:

Indicator            Correlation
------------------------------
Gdp                  0.76
Support              0.74
Expectancy           0.73
Freedom              0.44
Generosity           -0.13
Corruption           0.02


Rank  Country                    Score      GDP        Support         Expectancy      Freedom    Generosity   Corruption
--------------------------------------------------------------------------------------------------------------------------------
1     Germany                    6.985      1.373      1.454           0.987           0.495     0.261         0.265
2     Belgium                    6.923      1.356      1.504           0.986           0.473     0.160         0.210
3     United States              6.892      1.433      1.457           0.874           0.454     0.280         0.128
4     Czech Republic             6.852      1.269      1.487           0.920           0.457     0.046         0.036
5     United Arab Emirates       6.825      1.503      1.310           0.825           0.598     0.262         0.182
6     Malta                      6.726      1.300      1.520           0.999           0.564     0.375         0.151
7     Mexico                     6.595      1.070      1.323           0.861           0.433     0.074         0.073
8     France                     6.592      1.324      1.472           1.045           0.436     0.111         0.183
9     Taiwan                     6.446      1.368      1.430           0.914           0.351     0.242         0.097
10    Chile                      6.444      1.159      1.369           0.920           0.357     0.187         0.056
```



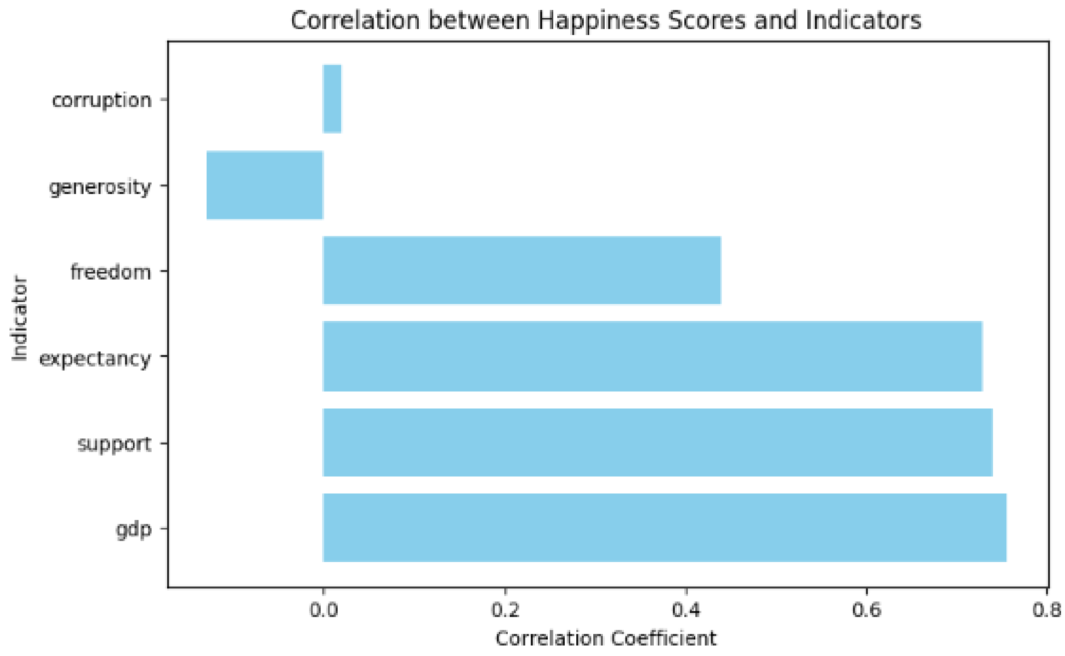
```
11   Guatemala              6.436    0.800    1.269       0.746       0.535   0.175     0.078
12   Saudi Arabia           6.375    1.403    1.357       0.795       0.439   0.080     0.132
13   Qatar                  6.374    1.684    1.313       0.871       0.555   0.220     0.167
14   Spain                  6.354    1.286    1.484       1.062       0.362   0.153     0.079
15   Panama                 6.321    1.149    1.442       0.910       0.516   0.109     0.054
16   Brazil                 6.300    1.004    1.439       0.802       0.390   0.099     0.086
17   Uruguay                6.293    1.124    1.465       0.891       0.523   0.127     0.150
18   Singapore              6.262    1.572    1.463       1.141       0.556   0.271     0.453
19   El Salvador            6.253    0.794    1.242       0.789       0.430   0.093     0.074
20   Italy                  6.223    1.294    1.488       1.039       0.231   0.158     0.030
21   Bahrain                6.199    1.362    1.368       0.871       0.536   0.255     0.110
22   Slovakia               6.198    1.246    1.504       0.881       0.334   0.121     0.014
23   Trinidad & Tobago      6.192    1.231    1.477       0.713       0.489   0.185     0.016
24   Poland                 6.182    1.206    1.438       0.884       0.483   0.117     0.050
25   Uzbekistan             6.174    0.745    1.529       0.756       0.631   0.322     0.240
26   Lithuania              6.149    1.238    1.515       0.818       0.291   0.043     0.042
27   Colombia               6.125    0.985    1.410       0.841       0.470   0.099     0.034
28   Slovenia               6.118    1.258    1.523       0.953       0.564   0.144     0.057
29   Nicaragua              6.105    0.694    1.325       0.835       0.435   0.200     0.127
30   Kosovo                 6.100    0.882    1.232       0.758       0.489   0.262     0.006
31   Argentina              6.086    1.092    1.432       0.881       0.471   0.066     0.050
32   Romania                6.070    1.162    1.232       0.825       0.462   0.083     0.005
33   Cyprus                 6.046    1.263    1.223       1.042       0.406   0.190     0.041
34   Ecuador                6.028    0.912    1.312       0.868       0.498   0.126     0.087
35   Kuwait                 6.021    1.500    1.319       0.808       0.493   0.142     0.097
36   Thailand               6.008    1.050    1.409       0.828       0.557   0.359     0.028
37   Latvia                 5.940    1.187    1.465       0.812       0.264   0.075     0.064
38   South Korea            5.895    1.301    1.219       1.036       0.159   0.175     0.056
39   Estonia                5.893    1.237    1.528       0.874       0.495   0.103     0.161
40   Jamaica                5.890    0.831    1.478       0.831       0.490   0.107     0.028
```

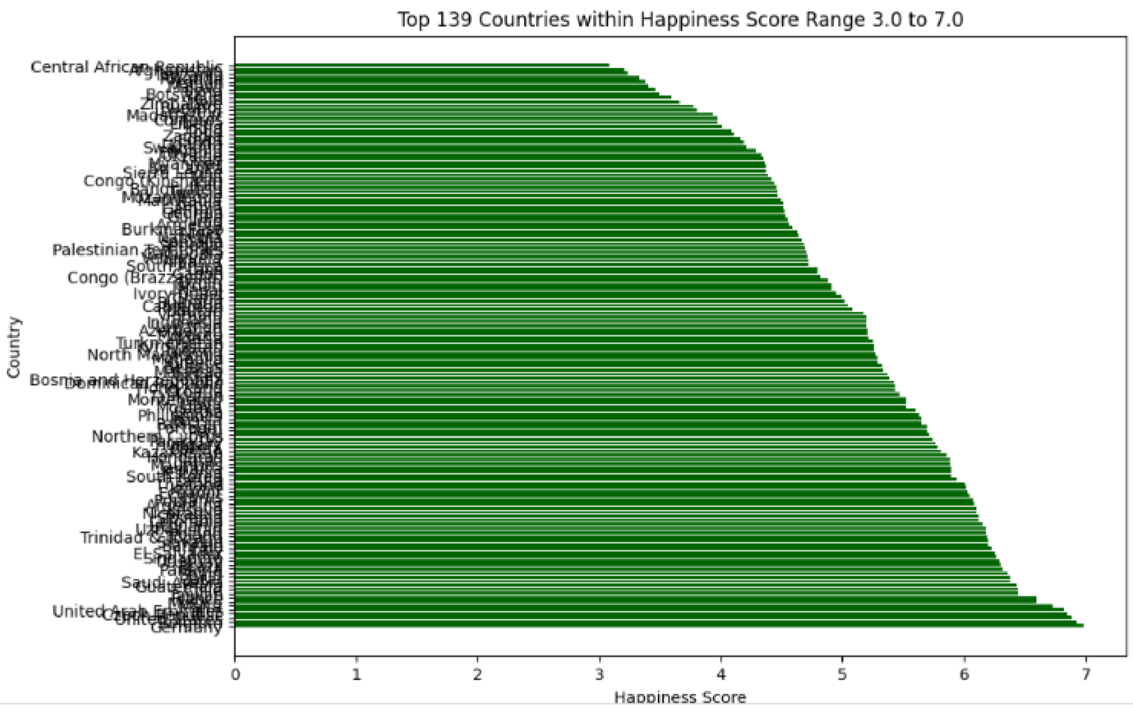


```
41   Mauritius               5.888    1.120    1.402       0.798       0.498   0.215     0.060
42   Japan                   5.886    1.327    1.419       1.088       0.445   0.069     0.140
43   Honduras                5.860    0.642    1.236       0.828       0.507   0.246     0.078
44   Kazakhstan              5.809    1.173    1.508       0.729       0.410   0.146     0.096
45   Bolivia                 5.779    0.776    1.209       0.706       0.511   0.137     0.064
46   Hungary                 5.758    1.201    1.410       0.828       0.199   0.081     0.020
47   Paraguay                5.743    0.855    1.475       0.777       0.514   0.184     0.080
48   Northern Cyprus         5.718    1.263    1.252       1.042       0.417   0.191     0.162
49   Peru                    5.697    0.960    1.274       0.854       0.455   0.083     0.027
50   Portugal                5.693    1.221    1.431       0.999       0.508   0.047     0.025
51   Pakistan                5.653    0.677    0.886       0.535       0.313   0.220     0.098
52   Russia                  5.648    1.183    1.452       0.726       0.334   0.082     0.031
53   Philippines             5.631    0.807    1.293       0.657       0.558   0.117     0.107
54   Serbia                  5.603    1.004    1.383       0.854       0.282   0.137     0.039
55   Moldova                 5.529    0.685    1.328       0.739       0.245   0.181     0.000
56   Libya                   5.525    1.044    1.303       0.673       0.416   0.133     0.152
57   Montenegro              5.523    1.051    1.361       0.871       0.197   0.142     0.080
58   Tajikistan              5.467    0.493    1.098       0.718       0.389   0.230     0.144
59   Croatia                 5.432    1.155    1.266       0.914       0.296   0.119     0.022
60   Hong Kong               5.430    1.438    1.277       1.122       0.440   0.258     0.287
61   Dominican Republic      5.425    1.015    1.401       0.779       0.497   0.113     0.101
62   Bosnia and Herzegovina  5.386    0.945    1.212       0.845       0.212   0.263     0.006
63   Turkey                  5.373    1.183    1.360       0.808       0.195   0.083     0.106
64   Malaysia                5.339    1.221    1.171       0.828       0.508   0.260     0.024
65   Belarus                 5.323    1.067    1.465       0.789       0.235   0.094     0.142
66   Greece                  5.287    1.181    1.156       0.999       0.067   0.000     0.034
67   Mongolia                5.285    0.948    1.531       0.667       0.317   0.235     0.038
68   North Macedonia         5.274    0.983    1.294       0.838       0.345   0.185     0.034
69   Nigeria                 5.265    0.696    1.111       0.245       0.426   0.215     0.041
70   Kyrgyzstan              5.261    0.551    1.438       0.723       0.508   0.300     0.023
```



```
 71   Turkmenistan              5.247    1.052    1.538        0.657        0.394   0.244      0.028
 72   Algeria                   5.211    1.002    1.160        0.785        0.086   0.073      0.114
 73   Morocco                   5.208    0.801    0.782        0.782        0.418   0.036      0.076
 74   Azerbaijan                5.208    1.043    1.147        0.769        0.351   0.035      0.182
 75   Lebanon                   5.197    0.987    1.224        0.815        0.216   0.166      0.027
 76   Indonesia                 5.192    0.931    1.203        0.660        0.491   0.498      0.028
 77   China                     5.191    1.029    1.125        0.893        0.521   0.058      0.100
 78   Vietnam                   5.175    0.741    1.346        0.851        0.543   0.147      0.073
 79   Bhutan                    5.082    0.813    1.321        0.604        0.457   0.370      0.167
 80   Cameroon                  5.044    0.549    0.910        0.331        0.381   0.187      0.037
 81   Bulgaria                  5.011    1.092    1.513        0.815        0.311   0.081      0.004
 82   Ghana                     4.996    0.611    0.868        0.486        0.381   0.245      0.040
 83   Ivory Coast               4.944    0.569    0.808        0.232        0.352   0.154      0.090
 84   Nepal                     4.913    0.446    1.226        0.677        0.439   0.285      0.089
 85   Jordan                    4.906    0.837    1.225        0.815        0.383   0.110      0.130
 86   Benin                     4.883    0.393    0.437        0.397        0.349   0.175      0.082
 87   Congo (Brazzaville)       4.812    0.673    0.799        0.508        0.372   0.105      0.093
 88   Gabon                     4.799    1.057    1.183        0.571        0.295   0.043      0.055
 89   Laos                      4.796    0.764    1.030        0.551        0.547   0.266      0.164
 90   South Africa              4.722    0.960    1.351        0.469        0.389   0.130      0.055
 91   Albania                   4.719    0.947    0.848        0.874        0.383   0.178      0.027
 92   Venezuela                 4.707    0.960    1.427        0.805        0.154   0.064      0.047
 93   Cambodia                  4.700    0.574    1.122        0.637        0.609   0.232      0.062
 94   Palestinian Territories   4.696    0.657    1.247        0.672        0.225   0.103      0.066
 95   Senegal                   4.681    0.450    1.134        0.571        0.292   0.153      0.072
 96   Somalia                   4.668    0.000    0.698        0.268        0.559   0.243      0.270
 97   Namibia                   4.639    0.879    1.313        0.477        0.401   0.070      0.056
 98   Niger                     4.628    0.138    0.774        0.366        0.318   0.188      0.102
 99   Burkina Faso              4.587    0.331    1.056        0.380        0.255   0.177      0.113
 100  Armenia                   4.559    0.850    1.055        0.815        0.283   0.095      0.064
101  Iran                      4.548    1.100    0.842        0.785        0.305   0.270      0.125
102  Guinea                    4.534    0.380    0.829        0.375        0.332   0.207      0.086
103  Georgia                   4.519    0.886    0.666        0.752        0.346   0.043      0.164
104  Gambia                    4.516    0.308    0.939        0.428        0.382   0.269      0.167
105  Kenya                     4.509    0.512    0.983        0.581        0.431   0.372      0.053
106  Mauritania                4.490    0.570    1.167        0.489        0.066   0.106      0.088
107  Mozambique                4.466    0.204    0.986        0.390        0.494   0.197      0.138
108  Tunisia                   4.461    0.921    1.000        0.815        0.167   0.059      0.055
109  Bangladesh                4.456    0.562    0.928        0.723        0.527   0.166      0.143
110  Iraq                      4.437    1.043    0.980        0.574        0.241   0.148      0.089
111  Congo (Kinshasa)          4.418    0.094    1.125        0.357        0.269   0.212      0.053
112  Mali                      4.390    0.385    1.105        0.308        0.327   0.153      0.052
113  Sierra Leone              4.374    0.268    0.841        0.242        0.309   0.252      0.045
114  Sri Lanka                 4.366    0.949    1.265        0.831        0.470   0.244      0.047
115  Myanmar                   4.360    0.710    1.181        0.555        0.525   0.566      0.172
116  Chad                      4.350    0.350    0.766        0.192        0.174   0.198      0.078
117  Ukraine                   4.332    0.820    1.390        0.739        0.178   0.187      0.010
118  Ethiopia                  4.286    0.336    1.033        0.532        0.344   0.209      0.100
119  Swaziland                 4.212    0.811    1.149        0.000        0.313   0.074      0.135
120  Uganda                    4.189    0.332    1.069        0.443        0.356   0.252      0.060
121  Egypt                     4.166    0.913    1.039        0.644        0.241   0.076      0.067
122  Zambia                    4.107    0.578    1.058        0.426        0.431   0.247      0.087
123  Togo                      4.085    0.275    0.572        0.410        0.293   0.177      0.085
124  India                     4.015    0.755    0.765        0.588        0.498   0.200      0.085
125  Liberia                   3.975    0.073    0.922        0.443        0.370   0.233      0.033
126  Comoros                   3.973    0.274    0.757        0.505        0.142   0.275      0.078
127  Madagascar                3.933    0.274    0.916        0.555        0.148   0.169      0.041
128  Lesotho                   3.802    0.489    1.169        0.168        0.359   0.107      0.093
129  Burundi                   3.775    0.046    0.447        0.380        0.220   0.176      0.180
130  Zimbabwe                  3.663    0.366    1.114        0.433        0.361   0.151      0.089
131  Haiti                     3.597    0.323    0.688        0.449        0.026   0.419      0.110
132  Botswana                  3.488    1.041    1.145        0.538        0.455   0.025      0.100
133  Syria                     3.462    0.619    0.378        0.440        0.013   0.331      0.141
134  Malawi                    3.410    0.191    0.560        0.495        0.443   0.218      0.089
135  Yemen                     3.380    0.287    1.163        0.463        0.143   0.108      0.077
136  Rwanda                    3.334    0.359    0.711        0.614        0.555   0.217      0.411
137  Tanzania                  3.231    0.476    0.885        0.499        0.417   0.276      0.147
138  Afghanistan               3.203    0.350    0.517        0.361        0.000   0.158      0.025
139  Central African Republic  3.083    0.026    0.000        0.105        0.225   0.235      0.035
```


Correlation between Happiness Scores and Indicators
corruption
generosity
freedom
expectancy
support
gdp
Indicator
0.0
0.2
0.4
0.6
0.8
Correlation Coefficient


Top 139 Countries within Happiness Score Range 3.0 to 7.0
Central African Republic
Palestinian Territories
Congo (Brazzaville)
North Macedonia
Bosnia and Herzegovina
Trinidad & Tobago
United Arab Emirates
Germany
Country
0
1
2
3
4
5
6
7
Happiness Score