**Topic: Prevalence Of Native Species In Various Regions Of The United Kingdom.**

**Visualizing Data Through Pie Charts and .csv Format-Based Tables**

Overview:

*This documentation explains the overall process that went into accomplishing the project which essentially includes a. The pseudo-code which represents the various decompositions that help us understand the solution part by part; b. Explanation of the decompositions without the pseudo-code and c. How the system was tested*

**1. Introduction**

**1.1 Introduction**

The goal of the project is to analyze the data provided in 4 different xlsx files and then convert the analysed data into relevant csv files and subsequently into pie charts with the help of Python language. We have been provided with the following files for extracting the data

* Admittals-June-2018.csv
* Districts of the UK.csv
* Mammal species.csv

These files were imported before extracting and systemising the data into tables and pie charts.

The following Python libraries were used while coding for the solution:

* csv reader
* Pandas
* Matplotlib

**1.2** **Aim**

The following results were to be accomplished along with their respective pie charts:

1. region-June2018.csv

which was to be formatted the following way:

|  |  |  |
| --- | --- | --- |
| Region Name | Number In Region | Percentage Of Total Cases |

2. species-June2018.csv

which was to be formatted the following way:

|  |  |  |
| --- | --- | --- |
| Species Name | Number Of Cases | Percentage Of Total Cases |

**2. Pseudo-code For Decomposition Process****Of Region-Based Data**

**2.1 Code For Segregating Regional Categorisation Columns**

**2.1.1 Code for getting the required column from Admittals\_June-2018**

admittals\_file\_var = read\_csv\_file(‘Admittals-June-2018.csv’)

print admittals\_file\_var

admittals\_var\_dataframe = convert\_to\_DataFrame(admittals\_file\_var[‘districts’])

print admittals\_var\_dataframe

districts\_list\_1= create-empty-list()

for value in admittals\_var\_dataframe[‘column’]

append (value) to district\_list

print district\_list\_1

**2.1.2 Code for getting the required column from Districts of the UK.csv**

UK\_district\_var = read\_csv\_file(‘Districts of the UK.csv’)

print UK\_var

districts\_dataframe = convert\_to\_DataFrame(UK\_district\_dataframe)

print districts\_dataframe

district\_list\_2 = convert\_to\_list(districts\_dataframe)

print district\_list\_2

**2.2 Code For Getting Cases Per District**

cases\_per\_district = create-empty-list()

for value in districts\_list\_2:

counter = 0

for value2 in district\_list\_1:

if districts\_list\_2 = districts\_list[value2]

increment counter by 1

append.to.cases\_per\_district(counter)

print cases\_per\_district

**2.3 Pseudo-Code For Working On Regional Data**

**2.3.1 Getting Individual Regions In One Dataframe**

regions = convert\_to\_list(districts\_dataframe[‘Regions’])

print regions

**2.3.2 Creating A Common List Out Of The Above Data**

cases\_districts\_regions = zip\_together\_and\_convert\_to\_list(cases\_per\_district, districts\_list, regions)

**2.3.3 Breaking Down The Repeated Regions List Into Singular Regions List**

regions\_one = create-empty-list()

for value in cases\_districts\_regions:

for value2 in regions:

if cases\_districts\_regions[value][2] == regions[value2]:

if regions[value2] in regions\_one:

pass

else:

region = regions[value2]

append to regions\_one(region)

print(regions\_one)

**2.3.4 Finding Out Individual Regional Cases**

regional\_cases = create-empty-list()

for value in regions\_one:

counter = 0

for value2 in range(cases\_districts\_regions):

if regions\_one[value1] == cases\_districts\_regions[value2][2]:

counter = counter + cases\_districts\_regions[value2][0]

else:

pass

append-to-regional\_cases(counter)

**2.4 Working Arithmetic Calculations On The Data**

**2.4.1 Adding Up All The Regional Cases**

total\_cases = 0

for total in regional\_cases:

total\_cases = regional\_cases[total]+total\_cases

print total\_cases

**2.4.2 Finding Out The Percentage Of Cases**

percentages = create-empty-list()

for value in regional\_cases:

value2 = (regional\_cases/total\_cases)\*100

percentages.append(value2)

print percentages

**2.4.3 Converting Three Lists Into DataFrames**

regions\_dataframe = convert\_to\_DataFrame(column = ‘Region Name’)

regional\_cases = convert\_to\_DataFrame(column = ‘Number In Region’)

percentages\_dataframe = convert\_to\_DataFrame(column = Percentage Of Total Cases)

**2.4.4 Combining All Above Data Frame Into One Single Data Frame**

main\_dataframe = combine(regions\_dataframe, regional\_cases, percentages\_dataframe)

**2.4.5 Finding Percentages Of Cases In A Given Region In Data Frame**

main\_dataframe[‘Percentages Of Total Cases] = round(find-percentage(Number\_In Region/Total)\*100)]

convert-to-integer(main\_dataframe[Percentages Of Total Cases] )

**2.5 Working With The Data To Convert It Into Pie Chart**

initial-color-list = [<list of various colors>]

final\_colors = create-empty-list()

final\_colors = [select random colors from initial-color-list in the range(0,31)]

x = main\_dataframe[‘Region Name’]

y = main\_dataframe[‘Percentage Of Total Cases’]

plot-pie-chart(y, labels=x, colors = final\_colors)

**3. Pseudo-code For Decomposition Process****Of Species-Based Data**

**3.1 Converting File Data In Data Frames**

**3.1.1 Converting Data From ‘Mammal Species.csv’**

the\_mammal\_file = read\_csv\_file(‘Mammal species.csv’)

print(the\_mammal\_file)

all\_species = convert\_to\_DataFrame(the\_mammal\_file[‘Species’])

**3.1.2 Converting Data From ‘Admittals-June-2018.csv’**

all\_data = read\_csv\_file(‘Admittals-June-2018.csv’)

print(all\_data)

random\_species = convert\_to\_DataFrame(all\_data[‘species’])

**3.1.3 Create A Combined List Of Two Data Frames**

combine1 = concatenate(all\_species, random\_species)

**3.2 Creating A Combined List Of Species And Their Count**

animal-list = create-empty-list()

for animals in combine1[0]:

count = 0

print animals

for all in combine1[1]:

if animals == all:

count = count + 1

print(count)

append(animals,count)to(animals\_list)

**3.3 Operations Carried Out After Converting The Above Lists Into Data Frame**

**3.3.1 Converting The Lists Into Data Frames**

d3 = convert-to-dataframe(animal\_list, columns)

d3.columns = [‘Species Name’ , ‘Number Of Cases’]

**3.3.2 Finding Total Of Number Of Cases**

Total = d3[‘Number Of Cases’].sum()

**3.3.3 Finding Total Percentage Of Cases**

d3[‘Percentage Of Total Cases’] = [round(Number Of Cases/Total)\*100]

d3[‘Percentage Of Total Cases’] = [int(to\_integer) for to\_integer in d3['Percentage Of Total Cases']]

**3.4 Working With The File ‘species-June2018.csv’**

**3.4.1 Reading The File To A Variable**

d3.to\_csv('species-June2018.csv')

species\_June2018 = pd.read\_csv('species-June2018.csv')

**3.5 Working With The Data To Convert It Into Pie Chart**

initial\_color\_list = [<insert random colors>]

final\_colors = create-empty-list()

x = df[‘Species Name’]

y = df[‘Percentage Of Total Cases’]

plot-pie-chart(y, labels=x, colors = final\_colors)

**4. Decomposition Description**

**4.1 Identifying The Crucial Steps:**

This is the initial and perhaps the most complicated of all steps, where we have to figure out what combinations of the various columns inside the files would be required to work upon to obtain the above listed columns.

The process applied to the species category was comparatively easier than when applied to the regions category. This was so because the data that makes up for the building blocks of the solution was much ‘readily available’. However, when it came to aligning regions in a list(both by Python syntactical meaning and in general), the raw data was not obvious and had to be combed iteration after iteration.

**4.2 Decomposing The Monthly Species Report**

**4.2.1 The Approach Taken Step By Step**

The first approach in the solution is to get the .csv provided as data and then convert them into Panda dataframes since dataframes are the standard method of getting data into their final form, before it is returned to a new .csv file.

Going by the above approach, we convert the relevant columns in ‘Mammals.csv’ and ‘Admittals-June-2018.csv’ to their respective dataframes.

Then we create a new dataframe ‘d3’, where we store these two dataframes as its constituent columns.

Next, we need to get a list where the species and their numbers in the given district are juxtaposed to each other. And we accomplish this with the help of a complex snippet consisting of nested for loops and if conditions.

Once we have got all the relevant data in the form of either lists or dataframe, we convert the former into dataframes and add them to the the combined dataframe ‘d3’ as its columns.

All that we are left to do within the complex calculations is that we have to find the sum of total number of cases and then find the percentage for each of the cases out of the total number of cases.

The final step is formatting the data into string format so that we can add symbols such as single quotes, commas and percentage signs.

And then we build the pie chart based on the above steps.

**4.2.2 Idea Behind The Whole Process**

The idea behind building the solution for the whole process is based on raw logic. After considering the required solution, we compare that with the given data. This is how we begin to reach the meta-understanding as to what has to be done to accomplish the solution.

After the ‘meta-steps’ have been figured out, we need to ‘get our hands dirty’ on the raw code such as finding how to create lists populated with entirely new data.

These steps are the core of the solution to this problem.

**4.3 Decomposing The Monthly Regions Report**

**4.3.1**  **The Approach Taken Step By Step**

The first step in the solution is to read the files and identify what data from which files is required to build the solution. Going by that, we convert the data from the ‘Admittals-June2018.csv’ and ‘Districts Of The UK.csv’ into lists and then into dataframes.

Then we get cases from individual districts by the implementation of a complex nested for loop with nested if conditions. This will help us to compare districts to their respecitve regions and get the cases per region later.

However, before we proceed with that, we need to list individual regions just once in a new list since the given data has the individual regions repeated. Later, with the help of two lists, namely, a. individual regions in the UK b. Number of species in a given district, we find the number of cases in each region of the UK.

Then we find the sum total of all the cases in all regions of the UK and then find the percentage of each region’s case with respect to this total. Once the data is cleaned and converted to pure ‘int’ values, we combine the three dataframes into a single dataframe which is then converted into a single .csv file.

Based on the data available in the above mentioned dataframe, we construct a pie chart.

**5. How The System Was Tested**

There was particular ‘test’ undertaken to check whether the system works or not. And this is since the system doesn’t make use of any functions or classess. All the code contained in the solution is in the form of standalone snippets.

While formulating the complex nested for loops containing the nested if-conditions, dummy data was created with counterparts for every column that were to be worked with for finding out the result.

For the regions data, 3 Regions/States were created, containing 3 cities each and each city having assigned certain points. The task was to assign each region/state a total of all the points of the cities located within them. The model was run on Python shell in IDLE and the results were desirable.

Similar models were used for all other for loop snippets that yielded desirable results.

**6. Assumptions About The Data**

There we no assumptions made about the data in particular. The data was already present in the ‘integer’ and ‘string’ data types, so the inferred logic was to be applied on the given data. That’s all that was there.

The logic was built on the observation that the data was chiefly ‘interger’ and ‘string’. If you want to know more about how the logic was derived, check sections 4. and 5. for detailed information.