Hello, welcome to another lesson. This is all about sharing my process of growth this career path. And I'm all excited about it.

Python is an amazing programming language and vast number of packages/libraries it has makes it even more exciting.

So, once again, i implore you to be encouraged by what you're doing on this path, even if it means doing it poorly till you gain mastery over it.

It will come more naturally as you do it, more consistently.

I'm just all excited about it all, so

Let's get started !!!

This lesson will be mostly about bringing your Learnings in the previous lessons together. And also, you'll be introduced to advanced techniques to make working with Data more efficiently.

Data extraction (which you'll find yourself doing, almost all of the time) will also be introduced.

Just do well to follow along with the examples and do it on your own, as well.

A better way to learn is by doing. So roll your sleeves up and get your hands in motion

The first step is to import all the libraries you need using the standard convention.

And remember to include "%matplotlib inline", so your plottings can display without additional effort. Failure to do this means you have to input "plt.show" after any plot for it to be displayed

The dataset is an excel file gotten from the "WHO" website and is readily available online for the public

Loading the excel file that contains the dataset and reading it the first 5 values directly from it

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
dataset = pd.read_excel("TB Comm_Engage.xlsx", index_col = 0, parse_da
te = "year")
dataset.head()
```

Out[2]:

country	iso2	iso3	iso_numeric	g_whoregion	year	bmu	bmu_community_impl	commu
Afghanistan	AF	AFG	4	EMR	2013	423.885246	232.037879	
Afghanistan	AF	AFG	4	EMR	2014	722.000000	232.037879	
Afghanistan	AF	AFG	4	EMR	2015	708.000000	539.000000	
Afghanistan	AF	AFG	4	EMR	2016	778.000000	778.000000	
Afghanistan	AF	AFG	4	EMR	2017	817.000000	817.000000	

Checking the shape of the DataFrame

In [3]:
```
dataset.shape
```

Out[3]: (558, 15)

Checking the size of the DataFrame

In [4]:
```
dataset.size
```

Out[4]: 8370

Checking the various types of data types present in the DataFrame

In [5]:
```
dataset.dtypes
```

Out[5]:
```
iso2                        object
iso3                        object
iso_numeric                  int64
g_whoregion                 object
year                         int64
bmu                        float64
bmu_community_impl         float64
community_data_available   float64
bmu_ref_data               float64
notified_ref               float64
notified_ref_community     float64
bmu_rxsupport_data         float64
bmu_rxsupport_data_coh     float64
rxsupport_community_coh    float64
rxsupport_community_succ   float64
dtype: object
```

Checking the indexes of the DataFrame

In [6]:
```
dataset.index
```

Out[6]:
```
Index(['Afghanistan', 'Afghanistan', 'Afghanistan', 'Afghanistan',
       'Afghanistan', 'Afghanistan', 'Albania', 'Albania', 'Algeria',
       'Algeria',
       ...
       'Zambia', 'Zambia', 'Zambia', 'Zambia', 'Zimbabwe', 'Zimbabwe',
       'Zimbabwe', 'Zimbabwe', 'Zimbabwe', 'Zimbabwe'],
      dtype='object', name='country', length=558)
```

Checking the columns of the DataFrame

In [7]:
```
dataset.columns
```

Out[7]:
```
Index(['iso2', 'iso3', 'iso_numeric', 'g_whoregion', 'year', 'bmu',
       'bmu_community_impl', 'community_data_available', 'bmu_ref_dat
a',
       'notified_ref', 'notified_ref_community', 'bmu_rxsupport_data',
       'bmu_rxsupport_data_coh', 'rxsupport_community_coh',
       'rxsupport_community_succ'],
      dtype='object')
```

Checking the general info of the DataFrame

In [8]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 558 entries, Afghanistan to Zimbabwe
Data columns (total 15 columns):
iso2                        558 non-null object
iso3                        558 non-null object
iso_numeric                 558 non-null int64
g_whoregion                 558 non-null object
year                        558 non-null int64
bmu                         558 non-null float64
bmu_community_impl          558 non-null float64
community_data_available    558 non-null float64
bmu_ref_data                558 non-null float64
notified_ref                558 non-null float64
notified_ref_community      558 non-null float64
bmu_rxsupport_data          558 non-null float64
bmu_rxsupport_data_coh      558 non-null float64
rxsupport_community_coh     558 non-null float64
rxsupport_community_succ    558 non-null float64
dtypes: float64(10), int64(2), object(3)
memory usage: 61.0+ KB
```

Checking summarized aggregated statistics of the DataFrame

In [9]:
```
dataset.describe()
```

Out[9]:

	iso_numeric	year	bmu	bmu_community_impl	community_data_available	bmu
count	558.000000	558.000000	558.000000	558.000000	558.000000	55
mean	435.387097	2015.824373	423.885246	232.037879	0.549296	23
std	254.444353	1.557278	1025.580114	586.414187	0.435137	34
min	4.000000	2013.000000	0.000000	0.000000	0.000000	
25%	218.000000	2015.000000	30.000000	18.000000	0.000000	6
50%	430.000000	2016.000000	128.000000	125.500000	0.549296	23
75%	645.250000	2017.000000	423.885246	232.037879	1.000000	23
max	894.000000	2018.000000	9746.000000	6819.000000	1.000000	427

Checking for null values in the Dataset

In [10]:
```
dataset.isna().sum()
```

Out[10]:
```
iso2                        0
iso3                        0
iso_numeric                 0
g_whoregion                 0
year                        0
bmu                         0
bmu_community_impl          0
community_data_available    0
bmu_ref_data                0
notified_ref                0
notified_ref_community      0
bmu_rxsupport_data          0
bmu_rxsupport_data_coh      0
rxsupport_community_coh     0
rxsupport_community_succ    0
dtype: int64
```

In [11]:
```
dataset.notna().sum()
```

Out[11]:
```
iso2                        558
iso3                        558
iso_numeric                 558
g_whoregion                 558
year                        558
bmu                         558
bmu_community_impl          558
community_data_available    558
bmu_ref_data                558
notified_ref                558
notified_ref_community      558
bmu_rxsupport_data          558
bmu_rxsupport_data_coh      558
rxsupport_community_coh     558
rxsupport_community_succ    558
dtype: int64
```

There are no null values in the Dataset. Note that's this is as a result of the fact that I'm working with a version of the file that I've worked with and cleaned before now.

If you get your own version online, it'll be in the raw form and you'll have to clean, prepare manipulate it before bit can be ready for further exploration and analysis

Now, you need to check the dataset for duplicated values in the DataFrame

In [12]:
```
dataset.duplicated().sum()
```

Out[12]:
```
0
```

There are no duplicated values in the dataset, and that's makes your tasks faster and easier. Your next line of action is to begin exploration, visualization and representation of the required insights from the Dataset

In [13]:
```
dataset.head(20)
```

Out[13]:

country	iso2	iso3	iso_numeric	g_whoregion	year	bmu	bmu_community_impl	commu
Afghanistan	AF	AFG	4	EMR	2013	423.885246	232.037879	
Afghanistan	AF	AFG	4	EMR	2014	722.000000	232.037879	
Afghanistan	AF	AFG	4	EMR	2015	708.000000	539.000000	
Afghanistan	AF	AFG	4	EMR	2016	778.000000	778.000000	
Afghanistan	AF	AFG	4	EMR	2017	817.000000	817.000000	
Afghanistan	AF	AFG	4	EMR	2018	887.000000	887.000000	
Albania	AL	ALB	8	EUR	2015	28.000000	28.000000	
Albania	AL	ALB	8	EUR	2016	28.000000	232.037879	
Algeria	DZ	DZA	12	AFR	2013	423.885246	232.037879	
Algeria	DZ	DZA	12	AFR	2014	271.000000	232.037879	
Algeria	DZ	DZA	12	AFR	2015	48.000000	48.000000	
Algeria	DZ	DZA	12	AFR	2016	240.000000	240.000000	
Algeria	DZ	DZA	12	AFR	2017	250.000000	17.000000	
Algeria	DZ	DZA	12	AFR	2018	247.000000	232.037879	
Angola	AO	AGO	24	AFR	2014	193.000000	232.037879	
Angola	AO	AGO	24	AFR	2015	283.000000	232.037879	
Angola	AO	AGO	24	AFR	2016	302.000000	280.000000	
Angola	AO	AGO	24	AFR	2017	302.000000	0.000000	
Angola	AO	AGO	24	AFR	2018	333.000000	9.000000	
Armenia	AM	ARM	51	EUR	2013	423.885246	232.037879	

For every Dataset, the exploration, visualization and insights you obtain from it depends on the tasks assigned to you.

For the sake of practice in this lesson, you'll extract some random but valuable insights and plot visualization where necessary, for the sake of practice and building your skills

To determine how many unique countries are represented in this Dataset

In [14]:
```
countries = dataset.index.unique()
countries1 = countries.value_counts().count()
print("There are",countries1,"countries represented in this dataset")
```

```
There are 113 countries represented in this dataset
```

The counts for the unique values in columns 'iso2', 'iso3', 'iso_numeric will be the same with that of country counts. But just to be double sure, let's run the following codes

In [15]:
```
iso2_uniques = dataset.iso2.unique()
iso2_uniques1 = len(iso2_uniques)
print("The number of unique values in \'iso2\' column is", iso2_unique
s1)
```

```
The number of unique values in 'iso2' column is 112
```

In [16]:
```
iso3_uniques = dataset.iso3.unique()
iso3_uniques1 = len(iso3_uniques)
print("The number of uniques values in \'iso3\' column is", iso3_uniqu
es1)
```

```
The number of uniques values in 'iso3' column is 113
```

In [17]:
```
isonum_uniques = dataset["iso_numeric"].unique()
isonum_uniques1 = len(isonum_uniques)
print("The number of unique values present in \'iso_numeric\' column i
s", isonum_uniques1)
```

```
The number of unique values present in 'iso_numeric' column is 113
```

The number of unique regions represented in this dataset will be checked and confirmed

In [18]:
```
regions_unique = dataset["g_whoregion"].unique()
regions_unique1 = len(regions_unique)
print("The number of unique regions represented in \'g_whoregion\' col
umn of this dataset is",regions_unique1)
```

```
The number of unique regions represented in 'g_whoregion' column of th
is dataset is 6
```

To check the number of years which the dataset covers. Or simply said, the number of unique years represented in this dataset

In [19]:
```
years_unique = dataset.year.unique()
years_unique1 = len(years_unique)
print("The number of unique years present in the \'year\' column of th
is dataset is", years_unique1)
```

```
The number of unique years present in the 'year' column of this datase
t is 6
```

In [20]:
```
print("The range of years covered in the dataset is within the range o
f year", years_unique.min(),"to", years_unique.max())
```

```
The range of years covered in the dataset is within the range of year
2013 to 2018
```

In [21]:
```
len(dataset["community_data_available"].unique())
```

Out[21]: 3

Now, create a DataFrame which gives you a visual clue of the regions and the countries in each region

In [22]:
```
country_regs = dataset["community_data_available"].groupby([dataset["g
_whoregion"], dataset.index])
country_regs1 = country_regs.size()
country_regs11 = country_regs1.unstack(0)
country_regs11
```

Out[22]:

g_whoregion	AFR	AMR	EMR	EUR	SEA	WPR
country						
Afghanistan	NaN	NaN	6.0	NaN	NaN	NaN
Albania	NaN	NaN	NaN	2.0	NaN	NaN
Algeria	6.0	NaN	NaN	NaN	NaN	NaN
Angola	5.0	NaN	NaN	NaN	NaN	NaN
Armenia	NaN	NaN	NaN	5.0	NaN	NaN
...	...	...	...	...	...	...
Venezuela (Bolivarian Republic of)	NaN	4.0	NaN	NaN	NaN	NaN
Viet Nam	NaN	NaN	NaN	NaN	NaN	6.0
Yemen	NaN	NaN	3.0	NaN	NaN	NaN
Zambia	5.0	NaN	NaN	NaN	NaN	NaN
Zimbabwe	6.0	NaN	NaN	NaN	NaN	NaN

113 rows × 6 columns

Pay attention to the 2 lines of codes below. A Series object was obtained and the "unstack()" method was used on it to convert it to a DataFrame object.

As i always say, there's no limitation to the things you can do with the vast amazing toolkits available in the Python libraries.

Just know your tools and the task you aim to accomplish, then unleash your creativity in getting things done

In [23]:
```
type(country_regs1)
```

Out[23]:
```
pandas.core.series.Series
```

In [24]:
```
type(country_regs11)
```

Out[24]:
```
pandas.core.frame.DataFrame
```

Now, it's time to get information about the exact number of countries in each regions

In [25]:
```
AFR_countries = country_regs11.AFR[country_regs11.AFR > 0]
AFR_countries
print("The number of countries present in the \'AFR\' region is", AFR_
countries.count())
```

```
The number of countries present in the 'AFR' region is 42
```

In [26]:
```
AMR_countries = country_regs11.AMR[country_regs11.AMR > 0]
AMR_countries
print("The number of countries present in the \'AMR\' region is", AMR_
countries.count())
```

```
The number of countries present in the 'AMR' region is 17
```

In [27]:
```
EMR_countries = country_regs11.EMR[country_regs11.EMR > 0]
EMR_countries
print("The number of countries present in the \'EMR\' region is", EMR_
countries.count())
```

```
The number of countries present in the 'EMR' region is 13
```

In [28]:
```
EUR_countries = country_regs11.EUR[country_regs11.EUR > 0]
EUR_countries
print("The number of countries present in the \'EUR\' region is", EUR_
countries.count())
```

```
The number of countries present in the 'EUR' region is 20
```

In [29]:
```
SEA_countries = country_regs11.SEA[country_regs11.SEA > 0]
SEA_countries
print("The number of countries present in the \'SEA\' region is", SEA_
countries.count())
```

```
The number of countries present in the 'SEA' region is 9
```

In [30]:
```
WPR_countries = country_regs11.SEA[country_regs11.SEA > 0]
WPR_countries
print("The number of countries present in the \'WPR\' region is", WPR_
countries.count())
```

```
The number of countries present in the 'WPR' region is 9
```

Next is to determine number of countries and regions that was attended to in the program per year

In [31]:
```
dataset.head(20)
```

Out[31]:

country	iso2	iso3	iso_numeric	g_whoregion	year	bmu	bmu_community_impl	commu
Afghanistan	AF	AFG	4	EMR	2013	423.885246	232.037879	
Afghanistan	AF	AFG	4	EMR	2014	722.000000	232.037879	
Afghanistan	AF	AFG	4	EMR	2015	708.000000	539.000000	
Afghanistan	AF	AFG	4	EMR	2016	778.000000	778.000000	
Afghanistan	AF	AFG	4	EMR	2017	817.000000	817.000000	
Afghanistan	AF	AFG	4	EMR	2018	887.000000	887.000000	
Albania	AL	ALB	8	EUR	2015	28.000000	28.000000	
Albania	AL	ALB	8	EUR	2016	28.000000	232.037879	
Algeria	DZ	DZA	12	AFR	2013	423.885246	232.037879	
Algeria	DZ	DZA	12	AFR	2014	271.000000	232.037879	
Algeria	DZ	DZA	12	AFR	2015	48.000000	48.000000	
Algeria	DZ	DZA	12	AFR	2016	240.000000	240.000000	
Algeria	DZ	DZA	12	AFR	2017	250.000000	17.000000	
Algeria	DZ	DZA	12	AFR	2018	247.000000	232.037879	
Angola	AO	AGO	24	AFR	2014	193.000000	232.037879	
Angola	AO	AGO	24	AFR	2015	283.000000	232.037879	
Angola	AO	AGO	24	AFR	2016	302.000000	280.000000	
Angola	AO	AGO	24	AFR	2017	302.000000	0.000000	
Angola	AO	AGO	24	AFR	2018	333.000000	9.000000	
Armenia	AM	ARM	51	EUR	2013	423.885246	232.037879	

Next, let's know the number of regions and the specific regions that were attended to in each year

All information about 2013 activities is extracted below

In [32]:
```
extract013 = dataset[dataset["year"] == 2013]
extract013.head()
```

Out[32]:

country	iso2	iso3	iso_numeric	g_whoregion	year	bmu	bmu_community_impl	commu
Afghanistan	AF	AFG	4	EMR	2013	423.885246	232.037879	
Algeria	DZ	DZA	12	AFR	2013	423.885246	232.037879	
Armenia	AM	ARM	51	EUR	2013	423.885246	232.037879	
Azerbaijan	AZ	AZE	31	EUR	2013	423.885246	232.037879	
Botswana	BW	BWA	72	AFR	2013	423.885246	232.037879	

Then, from the Dataset, information about the countries attended to in 2013 will be obtained

In [33]:
```
countries013 = extract013["year"].groupby(extract013.index)
countries13 = countries013.size()
countries13.head(15)
```

Out[33]:
```
country
Afghanistan                       1
Algeria                           1
Armenia                           1
Azerbaijan                        1
Botswana                          1
Bulgaria                          1
Burkina Faso                      1
Burundi                           1
Cabo Verde                        1
Cameroon                          1
Chad                              1
Côte d'Ivoire                     1
Democratic Republic of the Congo  1
Eritrea                           1
Ethiopia                          1
Name: year, dtype: int64
```

To determine how many regions and countries were attended to in 2013

In [34]:
```
regions013 = extract013.year.groupby(extract013["g_whoregion"])
regions13 = regions013.count()
regions13
```

Out[34]:
```
g_whoregion
AFR    29
EMR     2
EUR     8
SEA     6
WPR     4
Name: year, dtype: int64
```

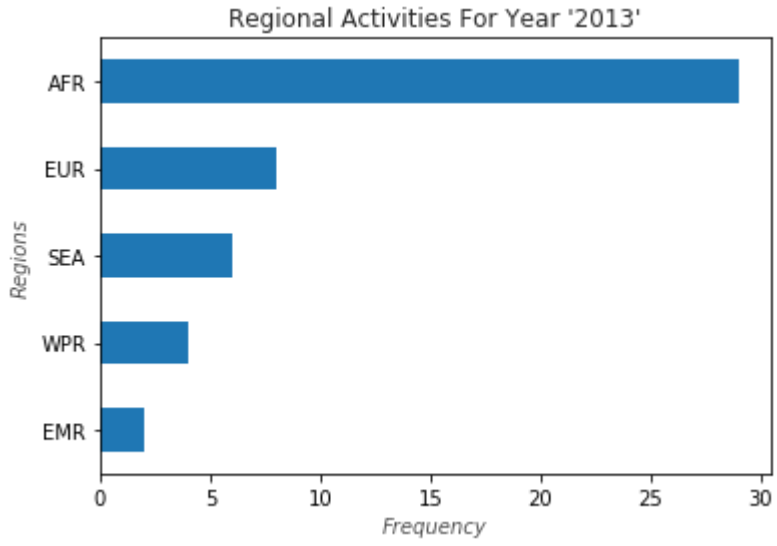
In [35]:
```
print("The number of countries attended to in year 2013 is",countries1
3.values.sum(),"\n\n\nThe number of regions attended to in year 2013 i
s",len(regions13.index))
```

```
The number of countries attended to in year 2013 is 49


The number of regions attended to in year 2013 is 5
```

In [36]:
```
sorted_2013 = regions13.sort_values()
sorted_2013.plot(kind = "barh")
plt.title("Regional Activities For Year \'2013\'", alpha = 0.85)
plt.ylabel("Regions", alpha = 0.7, fontstyle = "italic")
plt.xlabel("Frequency", alpha = 0.7, fontstyle = 'italic')
```

Out[36]:
```
Text(0.5, 0, 'Frequency')
```



All information about 2014 will be extracted

In [37]:
```
extract014 = dataset[dataset.year == 2014]
extract014.head()
```

Out[37]:

country	iso2	iso3	iso_numeric	g_whoregion	year	bmu	bmu_community_impl	community_d
Afghanistan	AF	AFG	4	EMR	2014	722.0	232.037879	
Algeria	DZ	DZA	12	AFR	2014	271.0	232.037879	
Angola	AO	AGO	24	AFR	2014	193.0	232.037879	
Armenia	AM	ARM	51	EUR	2014	66.0	232.037879	
Azerbaijan	AZ	AZE	31	EUR	2014	69.0	232.037879	

To determine the regions and countries that were attended to in the year 2014, the data was extracted from the dataset dataFrame and the needed information was retrieved from it

In [38]:
```
countries014 = extract014.year.groupby(extract014.index)
countries14 = countries014.count()
countries14
```

Out[38]:
```
country
Afghanistan                    1
Algeria                        1
Angola                         1
Armenia                        1
Azerbaijan                     1
                              ..
United Republic of Tanzania    1
Uzbekistan                     1
Viet Nam                       1
Zambia                         1
Zimbabwe                       1
Name: year, Length: 75, dtype: int64
```

In [39]:
```
regions014 = extract014.year.groupby(extract014["g_whoregion"])
regions14 = regions014.size()
regions14
```

Out[39]:
```
g_whoregion
AFR    39
AMR     3
EMR     6
EUR    11
SEA     9
WPR     7
Name: year, dtype: int64
```

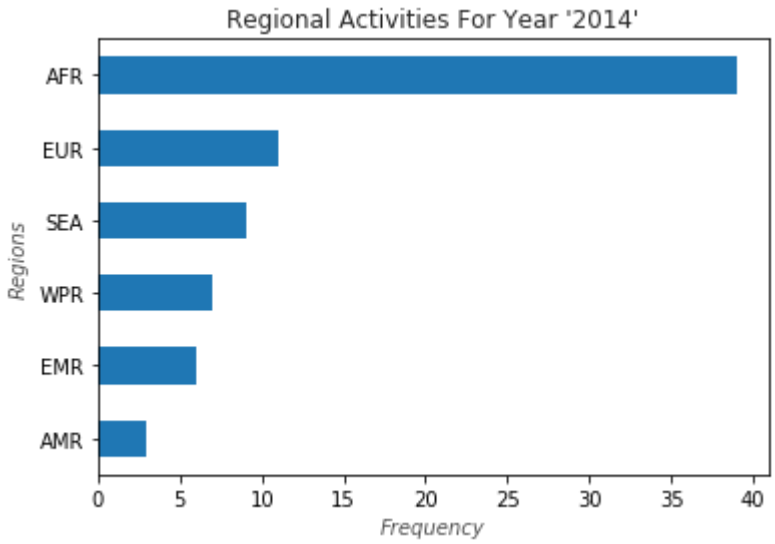
In [40]:
```
print("The number of countries attended to in year 2014 is",countries14.values.sum(),"\n\n\nThe number of regions attended to in year 2014 is",len(regions14.index))
```

```
The number of countries attended to in year 2014 is 75


The number of regions attended to in year 2014 is 6
```

In [41]:
```
sorted_2014 = regions14.sort_values()
sorted_2014.plot(kind = "barh")
plt.title("Regional Activities For Year \'2014\'", alpha = 0.85)
plt.ylabel("Regions", alpha = 0.7, fontstyle = "italic")
plt.xlabel("Frequency", alpha = 0.7, fontstyle = 'italic')
```

Out[41]:
```
Text(0.5, 0, 'Frequency')
```



All information about 2015 is extracted

In [42]:
```
extract015 = dataset[dataset["year"] == 2015]
extract015.head()
```

Out[42]:

country	iso2	iso3	iso_numeric	g_whoregion	year	bmu	bmu_community_impl	community_d
Afghanistan	AF	AFG	4	EMR	2015	708.0	539.000000	
Albania	AL	ALB	8	EUR	2015	28.0	28.000000	
Algeria	DZ	DZA	12	AFR	2015	48.0	48.000000	
Angola	AO	AGO	24	AFR	2015	283.0	232.037879	
Armenia	AM	ARM	51	EUR	2015	66.0	0.000000	

In [43]:
```
extract015["year"].unique()
```

Out[43]:
```
array([2015], dtype=int64)
```

To determine the number of regions and countries that were attended to in the year 2015, the information was extracted from the dataset DataFrame and the needed details was retrieved from it

In [44]:
```
countries015 = extract015.year.groupby(extract015.index)
countries15 = countries015.count()
countries15.head(15)
```

Out[44]:
```
country
Afghanistan                         1
Albania                             1
Algeria                             1
Angola                              1
Armenia                             1
Azerbaijan                          1
Bangladesh                          1
Belarus                             1
Benin                               1
Bhutan                              1
Bolivia (Plurinational State of)    1
Bosnia and Herzegovina              1
Botswana                            1
Brazil                              1
Bulgaria                            1
Name: year, dtype: int64
```

In [45]:
```
regions015 = extract015.year.groupby(extract015["g_whoregion"])
regions15 = regions015.size()
regions15
```

Out[45]:
```
g_whoregion
AFR    42
AMR    17
EMR    13
EUR    20
SEA     9
WPR    12
Name: year, dtype: int64
```

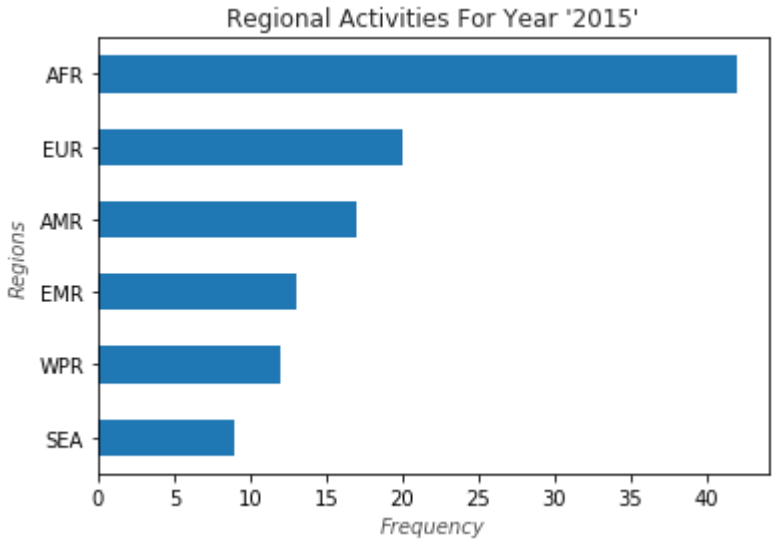
In [46]:
```
print("The number of countries attended to in the year 2015 is",countr
ies15.values.sum(),"\n\n\nThe number p regions attended to in the year
2015 is",len(regions15))
```

```
The number of countries attended to in the year 2015 is 113


The number p regions attended to in the year 2015 is 6
```

In [47]:
```
sorted_2015 = regions15.sort_values()
sorted_2015.plot(kind = "barh")
plt.title("Regional Activities For Year \'2015\'", alpha = 0.85)
plt.ylabel("Regions", alpha = 0.7, fontstyle = "italic")
plt.xlabel("Frequency", alpha = 0.7, fontstyle = 'italic')
```

Out[47]:
```
Text(0.5, 0, 'Frequency')
```



All information about 2016 is extracted

In [48]:
```
extract016 = dataset[dataset["year"] == 2016]
extract016.head()
extract016.year.unique()
```

Out[48]:
```
array([2016], dtype=int64)
```

To determine the number of regions and countries that were attended to in the year 2016, the dataset was extracted from the corresponding DataFrame and the needed details was retrieved from there

In [49]:
```
countries016 = extract016.year.groupby(extract016.index)
countries16 = countries016.count()
countries16.head(15)
```

Out[49]:
```
country
Afghanistan                          1
Albania                              1
Algeria                              1
Angola                               1
Armenia                              1
Azerbaijan                           1
Bangladesh                           1
Belarus                              1
Benin                                1
Bhutan                               1
Bolivia (Plurinational State of)     1
Bosnia and Herzegovina               1
Botswana                             1
Brazil                               1
Bulgaria                             1
Name: year, dtype: int64
```

In [50]:
```
regions016 = extract016.year.groupby(extract016["g_whoregion"])
region16 = regions016.count()
region16
```

Out[50]:
```
g_whoregion
AFR    40
AMR    17
EMR    13
EUR    19
SEA     9
WPR    12
Name: year, dtype: int64
```

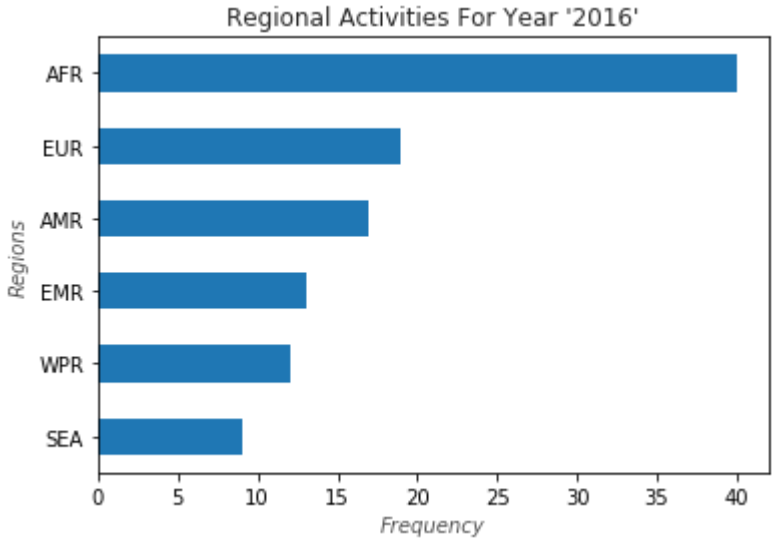
In [51]:
```
print("The number of countries attended to in the year 2016 is", count
ries16.values.sum(),"\n\n\nThe number of regions attended to in the ye
ar 2016 is", len(region16.index))
```

```
The number of countries attended to in the year 2016 is 110


The number of regions attended to in the year 2016 is 6
```

In [52]:
```
sorted_2016 = region16.sort_values()
sorted_2016.plot(kind = "barh")
plt.title("Regional Activities For Year \'2016\'", alpha = 0.85)
plt.ylabel("Regions", alpha = 0.7, fontstyle = "italic")
plt.xlabel("Frequency", alpha = 0.7, fontstyle = 'italic')
```

Out[52]:
```
Text(0.5, 0, 'Frequency')
```



All information about 2017 is extracted

In [53]:
```
extract017 = dataset[dataset.year == 2017]
extract017
extract017.year.unique()
```

Out[53]:
```
array([2017], dtype=int64)
```

To determine the number of regions and countries that was attended to in the year 2017, the details was extracted from dataset DataFrame and the required details was retrieved from there

In [54]:
```
countries017 = extract017.year.groupby(extract017.index)
countries17 = countries017.count()
countries17
```

Out[54]:
```
country
Afghanistan                              1
Algeria                                  1
Angola                                   1
Armenia                                  1
Azerbaijan                               1
                                        ..
Venezuela (Bolivarian Republic of)       1
Viet Nam                                 1
Yemen                                    1
Zambia                                   1
Zimbabwe                                 1
Name: year, Length: 110, dtype: int64
```

In [55]:
```
regions017 = extract017.year.groupby(extract017["g_whoregion"])
regions17 = regions017.count()
regions17
```

Out[55]:
```
g_whoregion
AFR    42
AMR    16
EMR    13
EUR    18
SEA     9
WPR    12
Name: year, dtype: int64
```

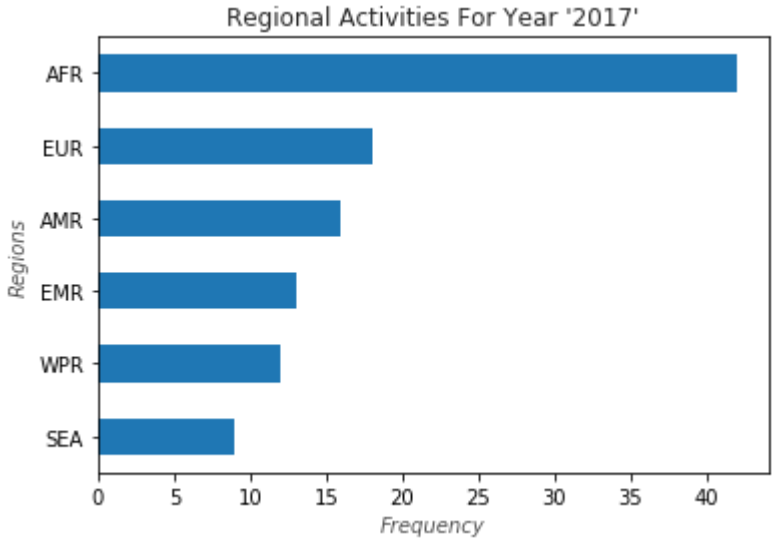
In [56]:
```
print("The number of countries attended to in the year 2017 is",countr
ies17.values.sum(),"\n\n\nThe number of regions attended to in the yea
r 2017 is",len(regions17))
```

```
The number of countries attended to in the year 2017 is 110


The number of regions attended to in the year 2017 is 6
```

In [57]:
```
sorted_2017 = regions17.sort_values()
sorted_2017.plot(kind = "barh")
plt.title("Regional Activities For Year \'2017\'", alpha = 0.85)
plt.ylabel("Regions", alpha = 0.7, fontstyle = "italic")
plt.xlabel("Frequency", alpha = 0.7, fontstyle = 'italic')
```

Out[57]:
```
Text(0.5, 0, 'Frequency')
```



All information about 2018 is extracted

In [58]:
```
extract018 = dataset[dataset["year"] == 2018]
extract018
extract018.year.unique()
```

Out[58]:
```
array([2018], dtype=int64)
```

To determine the number of regions and countries that was attended to in the year 2018, the information was extracted from the dataset DataFrame and the necessary details was retrieved from there

In [59]:
```
countries018 = extract018.year.groupby(extract018.index)
countries18 = countries018.size()
countries18.head(15)
```

Out[59]:
```
country
Afghanistan                          1
Algeria                              1
Angola                               1
Azerbaijan                           1
Bangladesh                           1
Belarus                              1
Benin                                1
Bhutan                               1
Bolivia (Plurinational State of)     1
Bosnia and Herzegovina               1
Botswana                             1
Brazil                               1
Bulgaria                             1
Burkina Faso                         1
Burundi                              1
Name: year, dtype: int64
```

In [60]:
```
regions018 = extract018.year.groupby(extract018["g_whoregion"])
regions18 = regions018.size()
regions18
```

Out[60]:
```
g_whoregion
AFR    42
AMR    16
EMR    10
EUR    13
SEA     9
WPR    11
Name: year, dtype: int64
```

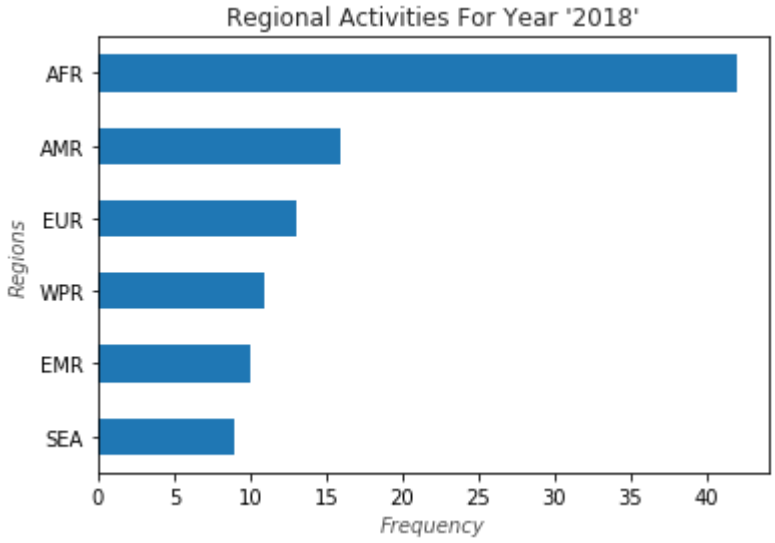
In [61]:
```
print("The number of countries attended to in the year 2018 is",countr
ies18.values.sum(),"\n\n\nThe number of regions attended to in the yea
r 2018 is",len(regions18.values))
```

```
The number of countries attended to in the year 2018 is 101


The number of regions attended to in the year 2018 is 6
```

In [62]:
```
sorted_2018 = regions18.sort_values()
sorted_2018.plot(kind = "barh")
plt.title("Regional Activities For Year \'2018\'", alpha = 0.85)
plt.ylabel("Regions", alpha = 0.7, fontstyle = "italic")
plt.xlabel("Frequency", alpha = 0.7, fontstyle = 'italic')
```

Out[62]:
```
Text(0.5, 0, 'Frequency')
```



I used the below codes to confirm if the total length of the distributed values across various years is equivalent to the total number of values in the actual dataset.

This is just a step to double check if my steps and works are correct

My results proved to be correct

In [63]:
```
len(countries13.values)+len(countries14.values) + len(countries15.valu
es) + len(countries16.values) + len(countries17.values) + len(countrie
s18.values)
```

Out[63]:
```
558
```

In [64]:
```
dataset.year.count()
```

Out[64]:
```
558
```

In [65]:
```
extract013.head(20)
```

Out[65]:

country	iso2	iso3	iso_numeric	g_whoregion	year	bmu	bmu_community_impl	commu
Afghanistan	AF	AFG	4	EMR	2013	423.885246	232.037879	
Algeria	DZ	DZA	12	AFR	2013	423.885246	232.037879	
Armenia	AM	ARM	51	EUR	2013	423.885246	232.037879	
Azerbaijan	AZ	AZE	31	EUR	2013	423.885246	232.037879	
Botswana	BW	BWA	72	AFR	2013	423.885246	232.037879	
Bulgaria	BG	BGR	100	EUR	2013	423.885246	232.037879	
Burkina Faso	BF	BFA	854	AFR	2013	423.885246	232.037879	
Burundi	BI	BDI	108	AFR	2013	423.885246	232.037879	
Cabo Verde	CV	CPV	132	AFR	2013	423.885246	232.037879	
Cameroon	CM	CMR	120	AFR	2013	423.885246	232.037879	
Chad	TD	TCD	148	AFR	2013	423.885246	232.037879	
Côte d'Ivoire	CI	CIV	384	AFR	2013	423.885246	232.037879	
Democratic Republic of the Congo	CD	COD	180	AFR	2013	423.885246	232.037879	
Eritrea	ER	ERI	232	AFR	2013	423.885246	232.037879	
Ethiopia	ET	ETH	231	AFR	2013	423.885246	232.037879	
Gabon	GA	GAB	266	AFR	2013	423.885246	232.037879	
Georgia	GE	GEO	268	EUR	2013	423.885246	232.037879	
Ghana	GH	GHA	288	AFR	2013	423.885246	232.037879	
Guinea	GN	GIN	324	AFR	2013	423.885246	232.037879	
India	IN	IND	356	SEA	2013	423.885246	232.037879	

In [66]:
```
afr13 = extract013[extract013["g_whoregion"] == 'AFR']
afri13 = afr13.index.size
afri13
```

Out[66]: 29

In [67]:
```
afr13.head(10)
```

Out[67]:

country	iso2	iso3	iso_numeric	g_whoregion	year	bmu	bmu_community_impl	commu
Algeria	DZ	DZA	12	AFR	2013	423.885246	232.037879	
Botswana	BW	BWA	72	AFR	2013	423.885246	232.037879	
Burkina Faso	BF	BFA	854	AFR	2013	423.885246	232.037879	
Burundi	BI	BDI	108	AFR	2013	423.885246	232.037879	
Cabo Verde	CV	CPV	132	AFR	2013	423.885246	232.037879	
Cameroon	CM	CMR	120	AFR	2013	423.885246	232.037879	
Chad	TD	TCD	148	AFR	2013	423.885246	232.037879	
Côte d'Ivoire	CI	CIV	384	AFR	2013	423.885246	232.037879	
Democratic Republic of the Congo	CD	COD	180	AFR	2013	423.885246	232.037879	
Eritrea	ER	ERI	232	AFR	2013	423.885246	232.037879	

In [68]:
```
dataset.describe()
```

Out[68]:

	iso_numeric	year	bmu	bmu_community_impl	community_data_available	bmu
count	558.000000	558.000000	558.000000	558.000000	558.000000	55
mean	435.387097	2015.824373	423.885246	232.037879	0.549296	23
std	254.444353	1.557278	1025.580114	586.414187	0.435137	34
min	4.000000	2013.000000	0.000000	0.000000	0.000000	
25%	218.000000	2015.000000	30.000000	18.000000	0.000000	6
50%	430.000000	2016.000000	128.000000	125.500000	0.549296	23
75%	645.250000	2017.000000	423.885246	232.037879	1.000000	23
max	894.000000	2018.000000	9746.000000	6819.000000	1.000000	427

Now, let's introduce you to the concept of joining and merging data or datasets using the "pd.concat()" function

In [69]:
```
regions_concat = pd.concat([regions13,regions14,regions15,region16,reg
ions17,regions18], keys = ["2013","2014","2015","2016","2017","2018"])
regions_concat.head(15)
```

Out[69]:
```
      g_whoregion
2013  AFR            29
      EMR             2
      EUR             8
      SEA             6
      WPR             4
2014  AFR            39
      AMR             3
      EMR             6
      EUR            11
      SEA             9
      WPR             7
2015  AFR            42
      AMR            17
      EMR            13
      EUR            20
Name: year, dtype: int64
```

In [70]:
```
region_concat = regions_concat.unstack(0)
region_concat
```

Out[70]:

g_whoregion	2013	2014	2015	2016	2017	2018
AFR	29.0	39.0	42.0	40.0	42.0	42.0
AMR	NaN	3.0	17.0	17.0	16.0	16.0
EMR	2.0	6.0	13.0	13.0	13.0	10.0
EUR	8.0	11.0	20.0	19.0	18.0	13.0
SEA	6.0	9.0	9.0	9.0	9.0	9.0
WPR	4.0	7.0	12.0	12.0	12.0	11.0

Just like we've done earlier, the concatenation process yields " A Pandas Series Object", and using the "unstack()" method, it's converted back to "A Pandas DataFrame Object". The details below

In [71]:
```
type(regions_concat)
```

Out[71]:
```
pandas.core.series.Series
```

In [72]:
```
type(region_concat)
```

Out[72]:
```
pandas.core.frame.DataFrame
```

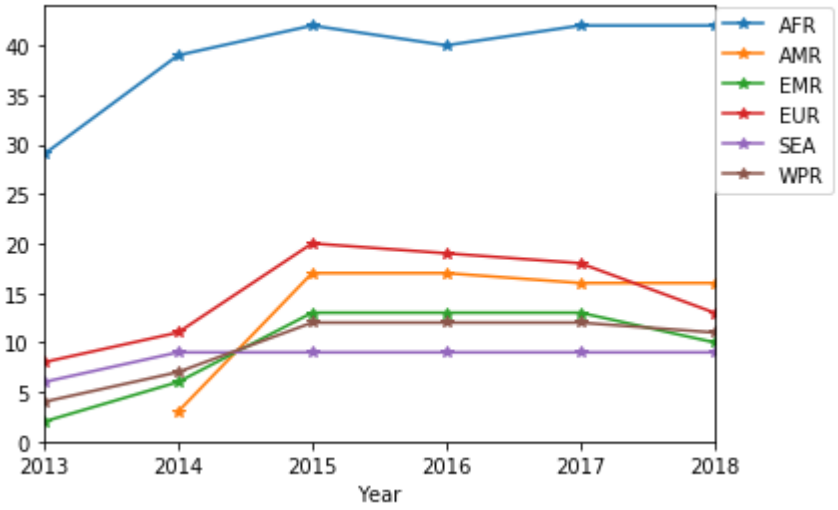
In [73]:
```
region_concat.columns.name = 'Year'
region_concat
```

Out[73]:

Year	2013	2014	2015	2016	2017	2018
g_whoregion						
AFR	29.0	39.0	42.0	40.0	42.0	42.0
AMR	NaN	3.0	17.0	17.0	16.0	16.0
EMR	2.0	6.0	13.0	13.0	13.0	10.0
EUR	8.0	11.0	20.0	19.0	18.0	13.0
SEA	6.0	9.0	9.0	9.0	9.0	9.0
WPR	4.0	7.0	12.0	12.0	12.0	11.0

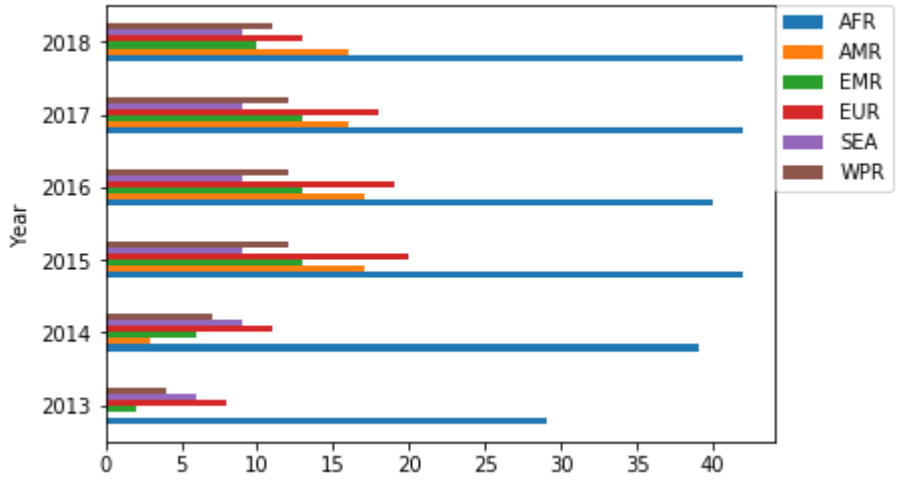
In [74]:
```
new_region = region_concat.T
new_region.plot(marker = '*')
plt.legend(ncol = 1, loc = (1, 0.57))
```

Out[74]: <matplotlib.legend.Legend at 0x98ec5850>



In [75]:
```
new_region.plot(kind = "barh")
plt.legend(ncol = 1, loc = (1, 0.575))
```

Out[75]:
```
<matplotlib.legend.Legend at 0x98ece270>
```



At this juncture I'll leave you to work more on your own and make new discoveries. Python is an amazing programming language with unlimited possibilities to what you can do with it's range of Data Science libraries

Remember, the best way to learn is by doing, so

Practice! Practice!! Practice!!!

And remember to feel encouraged no matter what your progress seems like. With a couple of practice and your commitment, all of the process will come naturally with ease.

Just keep doing it and improve on your abilities.

Keep Practicing at your best

Happy Learning !

In [ ]:

In [ ]: