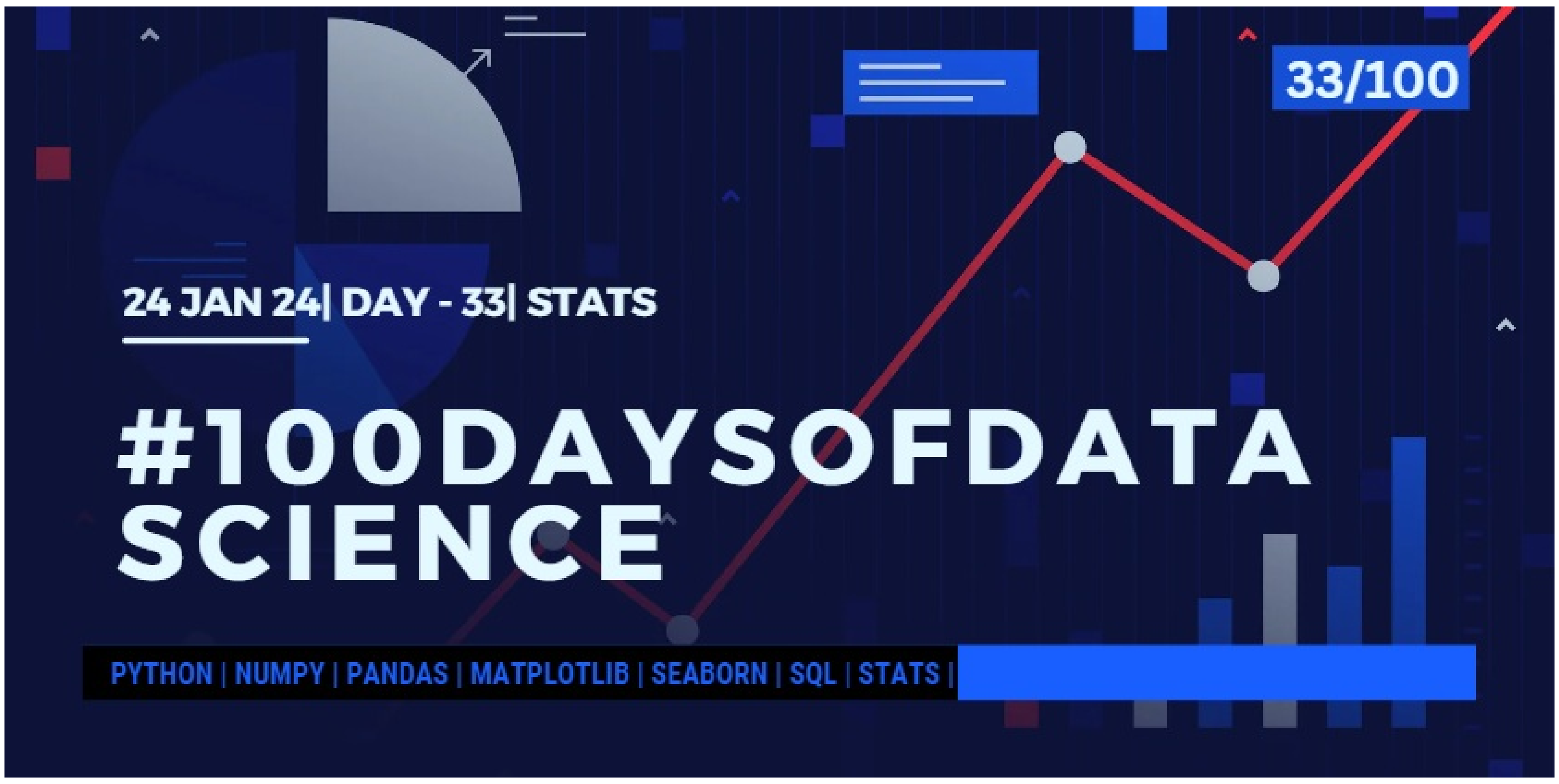


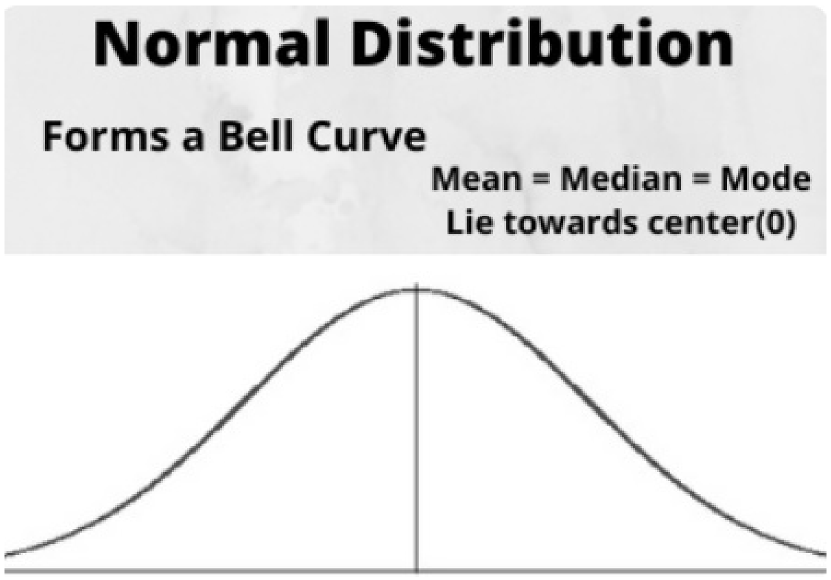
In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew
from scipy.stats import kurtosis
import warnings
warnings.filterwarnings('ignore')
```

Statistics - 2

Normal Distribution in statistics

A normal distribution is a distribution in form of a bell curve and most of the datasets in machine learning follow a normal distribution and if not then we try to transform it into normal distribution and many machine learning algorithms work very well on this distribution because in real-world scenario also many use cases follow this distribution like salary, very fewer employees will be there that are having less salary, and very less employee with very high salary and most of the employees will lie in middle or in the medium range.

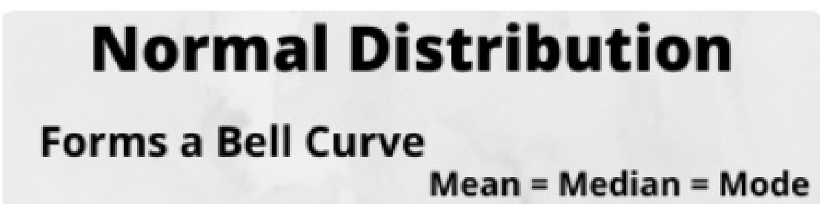


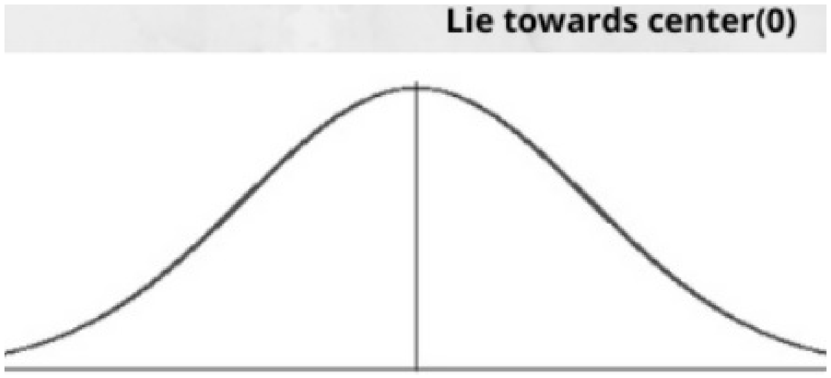
Distribution of Data:

In distribution we check the symmetry of data & heaviness of its tails when we are working with symmetry we term it as skewness and we are working with heaviness of tails we term it as kurtosis

Skewness:

- **Skewness is a measure of the symmetry of distribution that you plot in form of a histogram, KDE which has a high peak towards the mode of data.**
- **Skewness range from -1 to +1**
- **Skewness is generally of 3 types as**
  1. **Left-skewed data**
  2. **Right-skewed data**
  3. **Symmetric distribution means Normal distribution.**





1. Positively Skewed

- Right skewed distribution means data that has a long tail towards the right side(Positive axis). A
- It ranges from 0.5 to 1
- In Positively Skewed data where
  - mode < median < mean
  - mean > median > mode

2. Negatively Skewed

- Left skewed distribution means data that has a long tail towards the left side(negative axis).
- It ranges from -0.5 to -1
- In Negatively Skewed data where
  - mode > median > mean
  - mean < median < mode

3. Normally distribution Data

- In a normal distribution, data is symmetrically distributed with no skew.
- It ranges from -0.5 to 0 or 0 to 0.5
- In Normal distributed data where
  - mean = mode = median

In [2]:

```
df = pd.read_csv(r'C:\Users\hp\Desktop\100DaysOfDataScience\Day 33\Pokemon.csv')
df.head()
```

Out[2]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Stage	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	2	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	3	False
3	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
4	5	Charmeleon	Fire	NaN	405	58	64	58	80	65	80	2	False

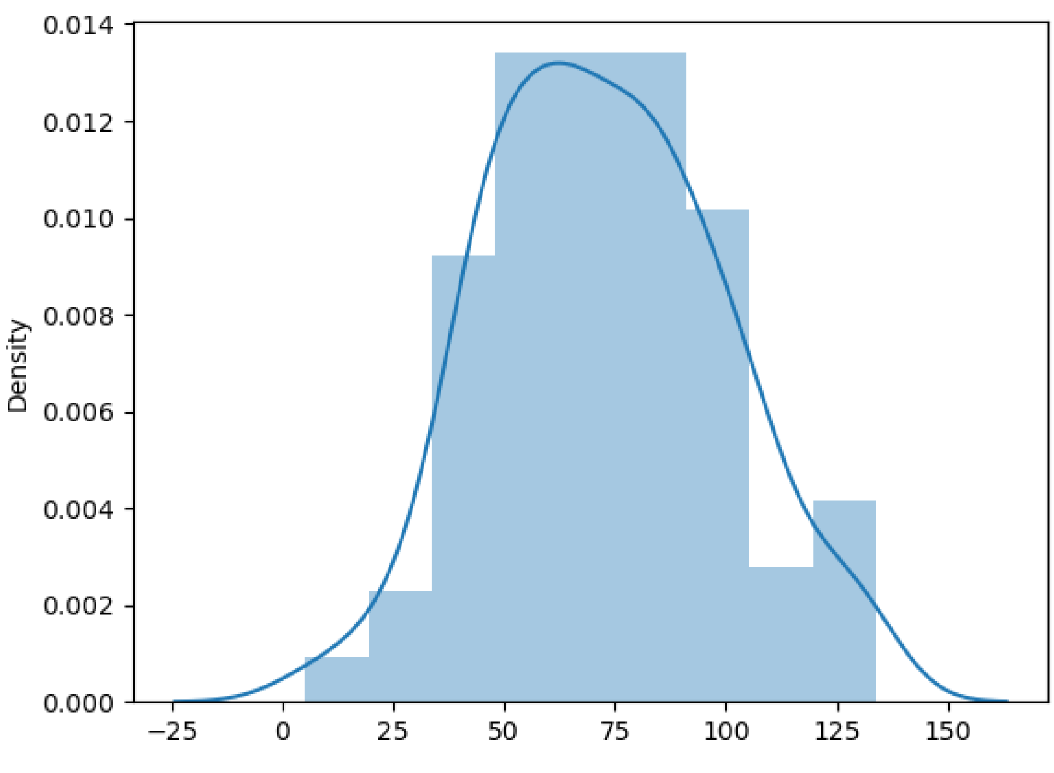
Approx Normal Distributed

In [3]:

```
sns.distplot(x=df['Attack'])
```

Out[3]:

```
<Axes: ylabel='Density'>
```



In [4]:

```
skew_attack = skew(df.Attack)
```

```
skew_attack
```

Out[4]:

```
0.14469715499053246
```

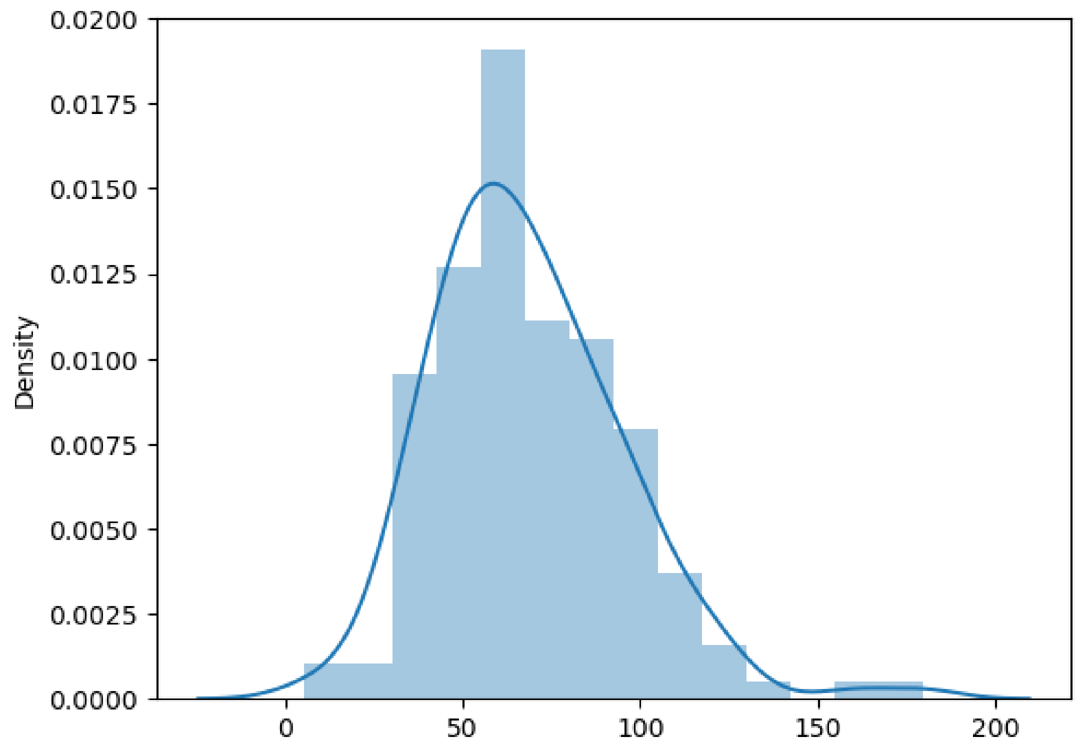
Positive Skewed

In [5]:

```
sns.distplot(x=df['Defense'])
```

Out[5]:

```
<Axes: ylabel='Density'>
```



In [6]:

```
skew_defense = skew(df.Defense)
skew_defense
```

Out[6]:

```
0.8303632725600234
```

Deleteing outliers to make it normal distribution

In [7]:

```
new_df = df[df['Defense']<=150]
new_df.head()
```

Out[7]:

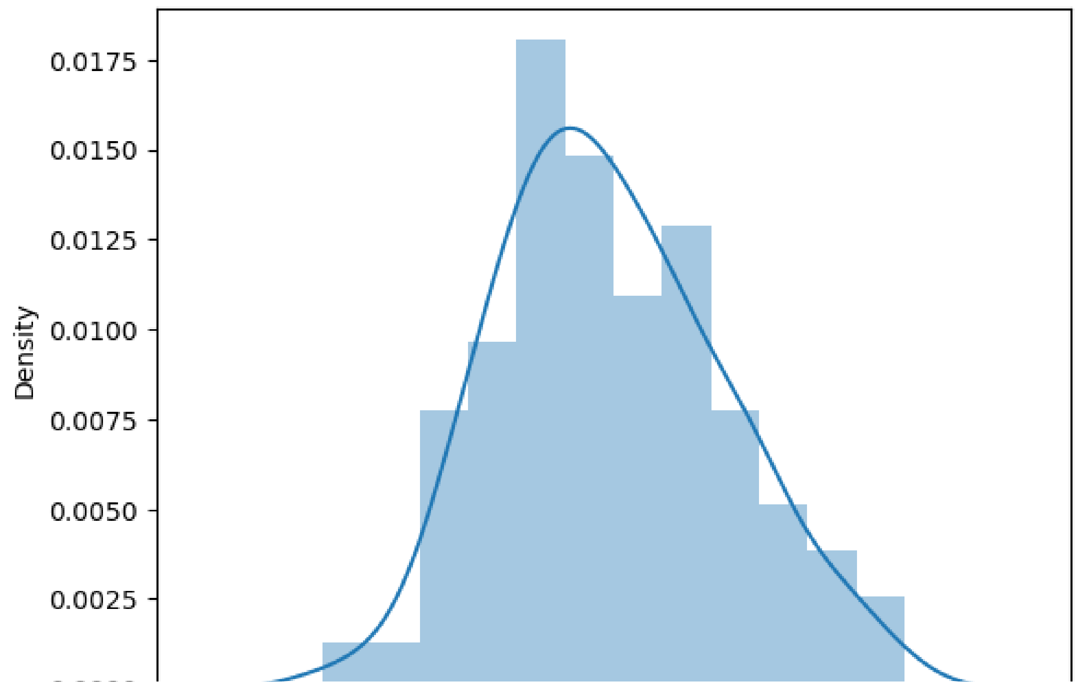
	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Stage	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	2	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	3	False
3	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
4	5	Charmeleon	Fire	NaN	405	58	64	58	80	65	80	2	False

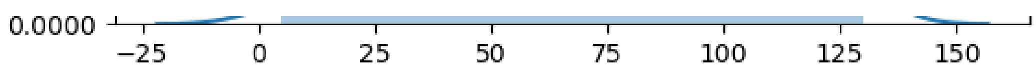
In [8]:

```
sns.distplot(x=new_df['Defense'])
```

Out[8]:

```
<Axes: ylabel='Density'>
```





In [9]:

```
skew_new_defense = skew(new_df.Defense) #Normal Skewed
skew_new_defense ##data is skewed skewness close to -1 to +1 (>0.75 or <-0.75) it need to changes
```

Out[9]:

```
0.2879498324533908
```

Kurtosis

- It is also a statistical term and an important characteristic of frequency distribution.
- It determines whether a distribution is heavy-tailed in respect of the normal distribution.
- It provides information about the shape of a frequency distribution.
- There are 3 types of kurtosis:
  1. In MesoKurtic if Fishers = 0, Pearsons = 3 (Normal)
  2. In PlatyKurtic if Fishers < 0, Pearsons < 3 (Flat)
  3. In LeptoKurtic if Fishers > 0, Pearsons > 3 (Sharp Peak)

PlatyKurtic

In [10]:

```
kurt_attack = kurtosis(df.Attack) #Fisher value
print("Fisher value: ",kurt_attack)
kurt_attack1 = kurtosis(df.Attack,fisher=False) #fisher =false --->means pearsons (Pearsons value)
print("Pearsons value: ",kurt_attack1)
```

```
Fisher value:  -0.3571247790751304
Pearsons value:  2.6428752209248696
```

LeptoKurtic

In [11]:

```
kurt_defense = kurtosis(df.Defense) #Fisher value
print("Fisher value: ",kurt_defense)
kurt_defense1 = kurtosis(df.Defense,fisher=False) #fisher =false --->means pearsons (Pearsons value)
print("Pearsons value: ",kurt_defense1)
```

```
Fisher value:  1.661203142349927
Pearsons value:  4.661203142349927
```

MesoKurtic

In [12]:

```
kurt_total = kurtosis(df.Total) #Fisher value
print("Fisher value: ",kurt_total)
kurt_total1 = kurtosis(df.Total,fisher=False) #fisher =false --->means pearsons (Pearsons value)
print("Pearsons value: ",kurt_total1)
```

```
Fisher value:  -0.7767216421709247
Pearsons value:  2.2232783578290753
```

Co Variance

- Co variance shows us how two variables are varying with each other that mean if one variable is increasing and other one is also increasing we ternm it as positive co variance
- When one goes up and other goes down we term it as negative co variance
  eg height and temperature
- When there is no relation between two variables we term as no co variance
  eg height and iq level

Co Relation

- It is measure of strength of relationship between two variables we mostly use Pearsons co relation coeffiecient which has a range from -1 to +1 in that 0 - no corealton close height and iq level to 1 is for high positive co relation PClass and Fare close to -1 is for high negative co relation height and temperature

Why co relation is preferred over co variance

To check the dependency of one variable over other co relation is mostly preferred because it has a range which defines the strength how strongly or weakly the variables are related to each other

In [13]:

```
X = df[['HP','Attack','Defense','Sp. Atk','Sp. Def','Speed']]
X.head()
```

Out[13]:

	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
0	45	49	49	65	65	45

	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
1	60	62	63	80	80	60
2	80	82	83	100	100	80
3	39	52	43	60	50	65
4	58	64	58	80	65	80

In [15]:

```
cov_df = X.cov()
cov_df
```

Out[15]:

	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
HP	817.394790	233.262737	92.178631	193.056998	339.669095	-31.305872
Attack	233.262737	707.355850	352.188742	111.036380	237.522340	138.503311
Defense	92.178631	352.188742	724.508962	144.061810	91.128830	-38.338322
Sp. Atk	193.056998	111.036380	144.061810	814.200530	361.050552	314.069272
Sp. Def	339.669095	237.522340	91.128830	361.050552	585.539603	254.134658
Speed	-31.305872	138.503311	-38.338322	314.069272	254.134658	715.395585

Heatmap

Heatmap are useful to visualize the magnitude of relationship between multiple variable using correlation matrix. Pairs which show high correlation < -0.75 or >0.75, should be considered and we can eliminate one variable out of the pair leading to feture selectuion

In [16]:

```
corr_df = X.corr(method='pearson')
corr_df
```

Out[16]:

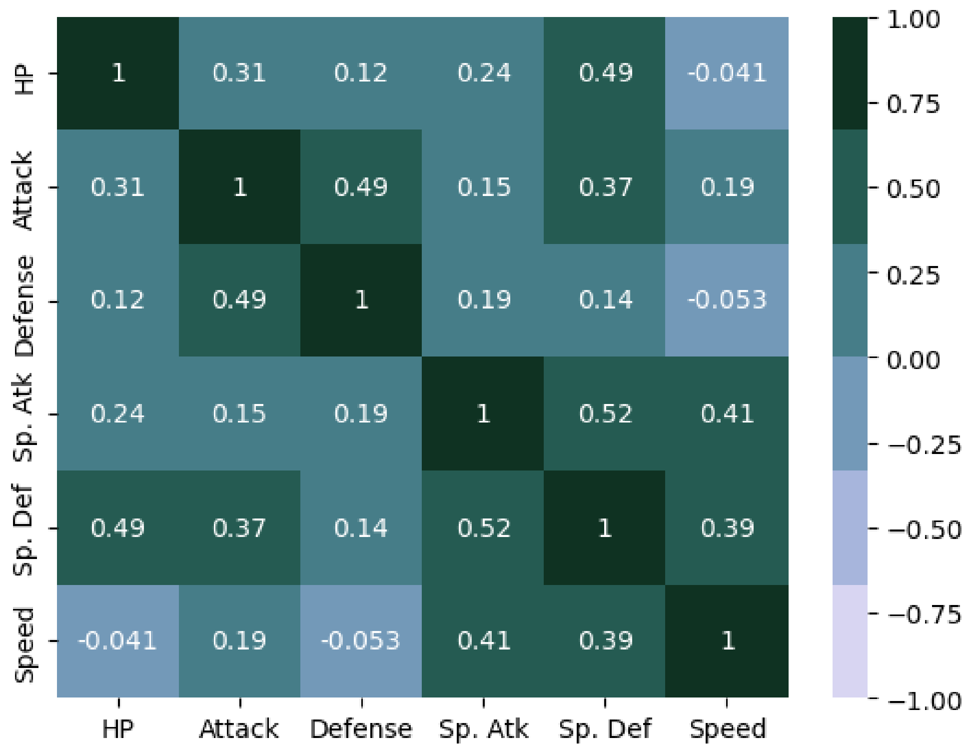
	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
HP	1.000000	0.306768	0.119782	0.236649	0.490978	-0.040939
Attack	0.306768	1.000000	0.491965	0.146312	0.369069	0.194701
Defense	0.119782	0.491965	1.000000	0.187569	0.139912	-0.053252
Sp. Atk	0.236649	0.146312	0.187569	1.000000	0.522907	0.411516
Sp. Def	0.490978	0.369069	0.139912	0.522907	1.000000	0.392656
Speed	-0.040939	0.194701	-0.053252	0.411516	0.392656	1.000000

In [19]:

```
sns.heatmap(corr_df,vmax=1.0,vmin=-1.0,annot=True,cmap=sns.cubehelix_palette(start=2))
```

Out[19]:

```
<Axes: >
```



In [ ]: