Date: 30/06/2021

Name: ADITI RAJESH DALAVI

Email_id: aditidalavi7337@gmail.com

Module Code: ML-13

Title: Basic Python blocks

Skills/Competencies to be acquired:

1. Accepting values from users
2. Creating function for input validation of the data
3. Exception
4. Use of Control statement

Duration of activity: 1 Hour

1. What is the purpose of this activity?

   To do Descriptive analysis. Interpret the data. Find the frequency distribution of the data. Find central tendency and measures of spread.Interpret it. Find skewness and Kurtosis. Interpret it.

1. Steps performed in this activity.

   Read the data. Import the required libraries. Analyise the data, group them, and explore the data. Plot the distributions of each exploration. Find central tendency and measures of spread.Interpret it. Find skewness and Kurtosis. Interpret it.

1. What resources / materials / equipment / tools did you use for this activity?

   Python for coding. Libraries: pandas, numpy, seaborn, matplotlib, matplot.pyplot, statistics.

1. What skills did you acquire?

   Python skills. Developed logic. Skills to interpret the data. Learnt to plot distribution of data.

1. Time taken to complete the activity?

5 hours.

In [ ]:

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sys
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import style
import statistics as stats
```

In [2]:

```
pk = pd.read_csv(r"C:\Users\aditi\Desktop\ML 13\assignment\Pokemon.csv")
```

In [3]:

```
pk
```

Out[3]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	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	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True

800 rows × 13 columns

In this given data, 13 columns and 800 rows are present.

In [4]:
```
pk.head()
```

Out[4]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	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	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

In [5]:
```
pk.tail()
```

Out[5]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True

In [6]:
```
pk.shape
```

Out[6]:
```
(800, 13)
```

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

Out[7]:
```
Index(['#', 'Name', 'Type 1', 'Type 2', 'Total', 'HP', 'Attack', 'Defense',
       'Sp. Atk', 'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

In [8]:

```
pk.nunique()
```

Out[8]:
```
#             721
Name          800
Type 1         18
Type 2         18
Total         200
HP             94
Attack        111
Defense       103
Sp. Atk       105
Sp. Def        92
Speed         108
Generation      6
Legendary       2
dtype: int64
```

Hence, all the analysis will be done with the help of type 1, generation,and legendary.

In [9]:
```
pk['Type 1'].unique()
```

Out[9]:
```
array(['Grass', 'Fire', 'Water', 'Bug', 'Normal', 'Poison', 'Electric',
       'Ground', 'Fairy', 'Fighting', 'Psychic', 'Rock', 'Ghost', 'Ice',
       'Dragon', 'Dark', 'Steel', 'Flying'], dtype=object)
```

In [10]:
```
pk['Type 1'].value_counts()
```

Out[10]:
```
Water       112
Normal       98
Grass        70
Bug          69
Psychic      57
Fire         52
Rock         44
Electric     44
Dragon       32
Ghost        32
Ground       32
Dark         31
Poison       28
Fighting     27
Steel        27
Ice          24
Fairy        17
```

```
Flying         4
Name: Type 1, dtype: int64
```

In 'Type 1'

Highest count is of WATER TYPE i.e. 112.

Lowest count is of FLYING TYPE i.e 4.

In [11]:

```
pk['Type 2'].unique()
```

Out[11]:

```
array(['Poison', nan, 'Flying', 'Dragon', 'Ground', 'Fairy', 'Grass',
       'Fighting', 'Psychic', 'Steel', 'Ice', 'Rock', 'Dark', 'Water',
       'Electric', 'Fire', 'Ghost', 'Bug', 'Normal'], dtype=object)
```

In "type 2"

nan value is present. Hence, filling this nan with none

In [12]:

```
pk.fillna('none',inplace=True)
```

In [13]:

```
pk['Type 2'].value_counts()
```

Out[13]:

```
none        386
Flying       97
Ground       35
Poison       34
Psychic      33
Fighting     26
Grass        25
Fairy        23
Steel        22
Dark         20
Dragon       18
Ghost        14
Ice          14
Water        14
Rock         14
Fire         12
Electric      6
Normal        4
```

```
Bug            3
Name: Type 2, dtype: int64
```

In [14]:
```
pk['Type 2'].unique()
```

Out[14]:
```
array(['Poison', 'none', 'Flying', 'Dragon', 'Ground', 'Fairy', 'Grass',
       'Fighting', 'Psychic', 'Steel', 'Ice', 'Rock', 'Dark', 'Water',
       'Electric', 'Fire', 'Ghost', 'Bug', 'Normal'], dtype=object)
```

In [15]:
```
pk.isnull().sum()
```

Out[15]:
```
#             0
Name          0
Type 1        0
Type 2        0
Total         0
HP            0
Attack        0
Defense       0
Sp. Atk       0
Sp. Def       0
Speed         0
Generation    0
Legendary     0
dtype: int64
```

no nan value present

In [16]:
```
pk.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   #           800 non-null    int64 
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      800 non-null    object
 4   Total       800 non-null    int64 
 5   HP          800 non-null    int64 
 6   Attack      800 non-null    int64 
 7   Defense     800 non-null    int64 
 8   Sp. Atk     800 non-null    int64 
 9   Sp. Def     800 non-null    int64 
```

```
 10  Speed       800 non-null    int64
 11  Generation  800 non-null    int64
 12  Legendary   800 non-null    bool
dtypes: bool(1), int64(9), object(3)
memory usage: 75.9+ KB
```

In the give data,

3 data types are there: (a.) 9 integer (b.) 3 strings (c.) 1 boolean

There are 800 entries(row) and 13 columns.

In [17]:

```
des=pk.describe()
des
```

Out[17]:

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	800.000000	800.00000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.00000
mean	362.813750	435.10250	69.258750	79.001250	73.842500	72.820000	71.902500	68.277500	3.32375
std	208.343798	119.96304	25.534669	32.457366	31.183501	32.722294	27.828916	29.060474	1.66129
min	1.000000	180.00000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000	1.00000
25%	184.750000	330.00000	50.000000	55.000000	50.000000	49.750000	50.000000	45.000000	2.00000
50%	364.500000	450.00000	65.000000	75.000000	70.000000	65.000000	70.000000	65.000000	3.00000
75%	539.250000	515.00000	80.000000	100.000000	90.000000	95.000000	90.000000	90.000000	5.00000
max	721.000000	780.00000	255.000000	190.000000	230.000000	194.000000	230.000000	180.000000	6.00000

In [18]:

```
pk['Legendary'].value_counts()
```

Out[18]:

```
False    735
True      65
Name: Legendary, dtype: int64
```

There are 65 legendary Pokemon in the given data.

In [19]:

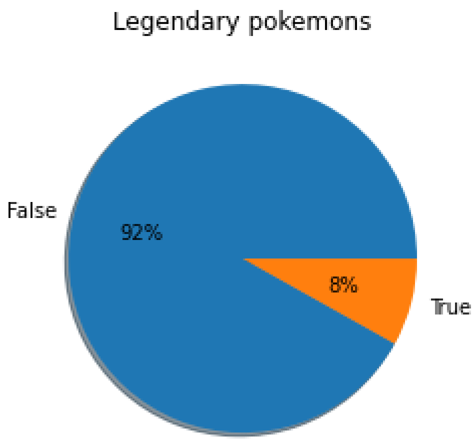
```
Legendary=pk['Legendary'].value_counts()
```

```
label=[False,True]
plt.pie(Legendary,
        labels=label,
       startangle = 0,
       shadow= True,
        autopct='%1.0f%%')


plt.title('Legendary pokemons')
```

Out[19]:

```
Text(0.5, 1.0, 'Legendary pokemons')
```



In [20]:

```
Generation=pk['Generation'].value_counts()
Generation
```

Out[20]:

```
1    166
5    165
3    160
4    121
2    106
6     82
Name: Generation, dtype: int64
```

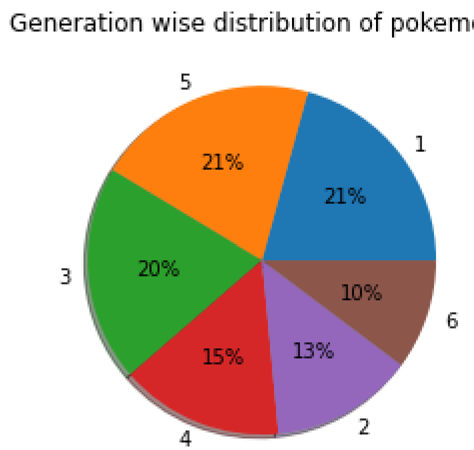
In [21]:

```
Generation=pk['Generation'].value_counts()
label=[1,5,3,4,2,6]
plt.pie(Generation,
        labels=label,
       startangle = 0,
```

```
        shadow= True,
         autopct='%1.0f%%')


plt.title('Generation wise distribution of pokemons')
```

Out[21]: Text(0.5, 1.0, 'Generation wise distribution of pokemons')



Generation 1 has highest Pokemon in number while Generation 6 has lowest Pokemon in number.

In [22]:
```
type1=pk['Type 1'].value_counts()
type1
```

Out[22]:
```
Water       112
Normal       98
Grass        70
Bug          69
Psychic      57
Fire         52
Rock         44
Electric     44
Dragon       32
Ghost        32
Ground       32
Dark         31
Poison       28
Fighting     27
```

```
Steel        27
Ice          24
Fairy        17
Flying        4
Name: Type 1, dtype: int64
```

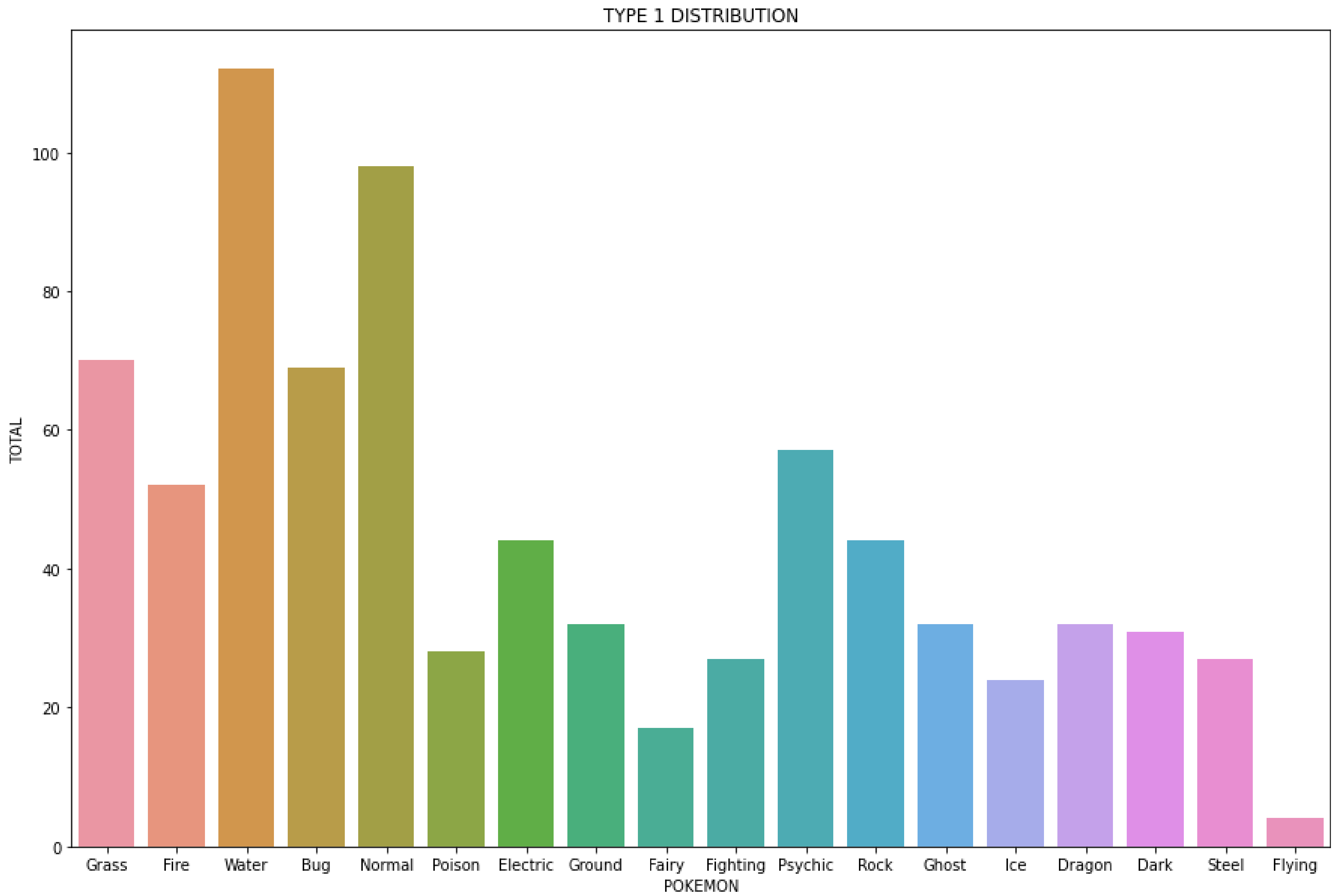
In [23]:

```
plt.figure(figsize=(15,10))
sys.countplot(pk['Type 1'])
plt.title('TYPE 1 DISTRIBUTION')
plt.xlabel('POKEMON')
plt.ylabel('TOTAL')
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[23]:

```
Text(0, 0.5, 'TOTAL')
```



IN TYPE 1:

Water type Pokemon is highest number of pokemon in total.

Flying type Pokemon is less in number of pokemons in total.

Rock, Electric Pokemon has same value count.

Ghost, Dragon, Ground has same value count.

Fighting, Steel has same value count.

In [24]:

```
pk['Type 2'].value_counts()
```

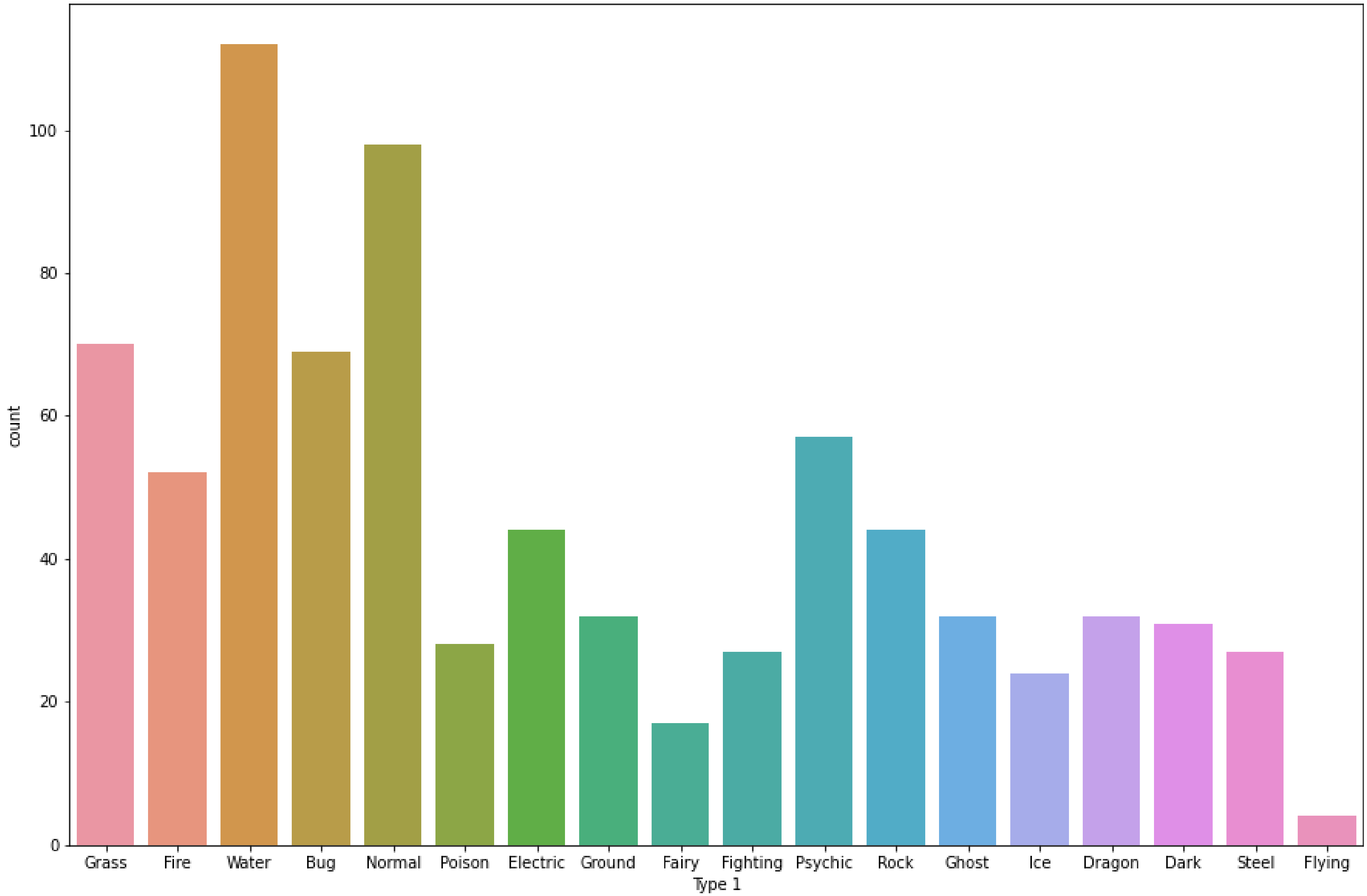
Out[24]:

```
none        386
Flying       97
Ground       35
Poison       34
Psychic      33
Fighting     26
Grass        25
Fairy        23
Steel        22
Dark         20
Dragon       18
Ghost        14
Ice          14
Water        14
Rock         14
Fire         12
Electric      6
Normal        4
Bug           3
Name: Type 2, dtype: int64
```

In [25]:

```
plt.figure(figsize=(15,10))
sys.countplot(pk['Type 1']);
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In type 2

Excluding, the none value,

Flying Pokemon has highest value count.

Bug pokemon has lowest value count.

water, ice, ghost, rock pokemon has same value count.

In [26]:

```
pk.groupby(['Generation','Legendary']).count()
```

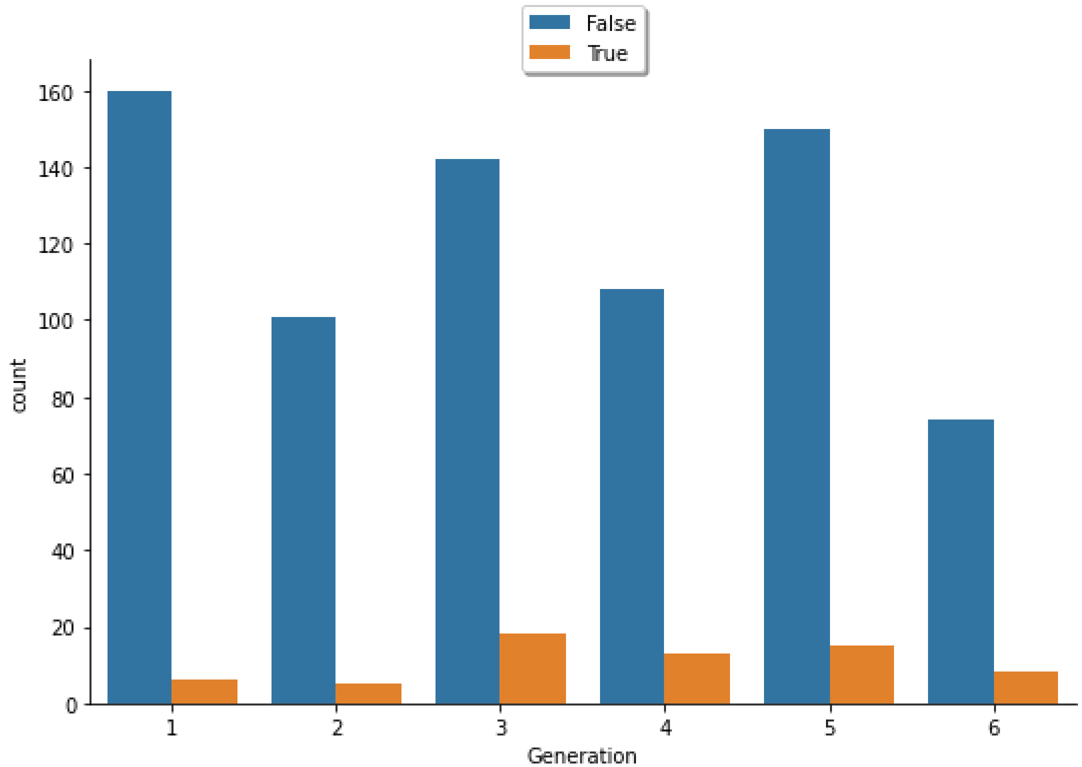
Out[26]:

Generation	Legendary	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
1	**False**	160	160	160	160	160	160	160	160	160	160	160
	True	6	6	6	6	6	6	6	6	6	6	6
2	**False**	101	101	101	101	101	101	101	101	101	101	101
	True	5	5	5	5	5	5	5	5	5	5	5
3	**False**	142	142	142	142	142	142	142	142	142	142	142
	True	18	18	18	18	18	18	18	18	18	18	18
4	**False**	108	108	108	108	108	108	108	108	108	108	108
	True	13	13	13	13	13	13	13	13	13	13	13
5	**False**	150	150	150	150	150	150	150	150	150	150	150
	True	15	15	15	15	15	15	15	15	15	15	15
6	**False**	74	74	74	74	74	74	74	74	74	74	74
	True	8	8	8	8	8	8	8	8	8	8	8

In [27]:

```
bar1= sys.catplot(x='Generation',
                  data=pk,
                  kind='count',
                  hue='Legendary',
                  height=5,
                  aspect=1.5,
                  legend=False,).set_axis_labels('Generation')

bar1.ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.1), shadow=True)
plt.show()
```



Generation 3 has highest legendary pokemon.

Generation 2 has lowest legendary pokemon.

In [28]:

```
pk.groupby(['Type 1','Legendary']).count()
```

Out[28]:

Type 1	Legendary	#	Name	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
Bug	False	69	69	69	69	69	69	69	69	69	69	69
Dark	False	29	29	29	29	29	29	29	29	29	29	29
	True	2	2	2	2	2	2	2	2	2	2	2
Dragon	False	20	20	20	20	20	20	20	20	20	20	20
	True	12	12	12	12	12	12	12	12	12	12	12

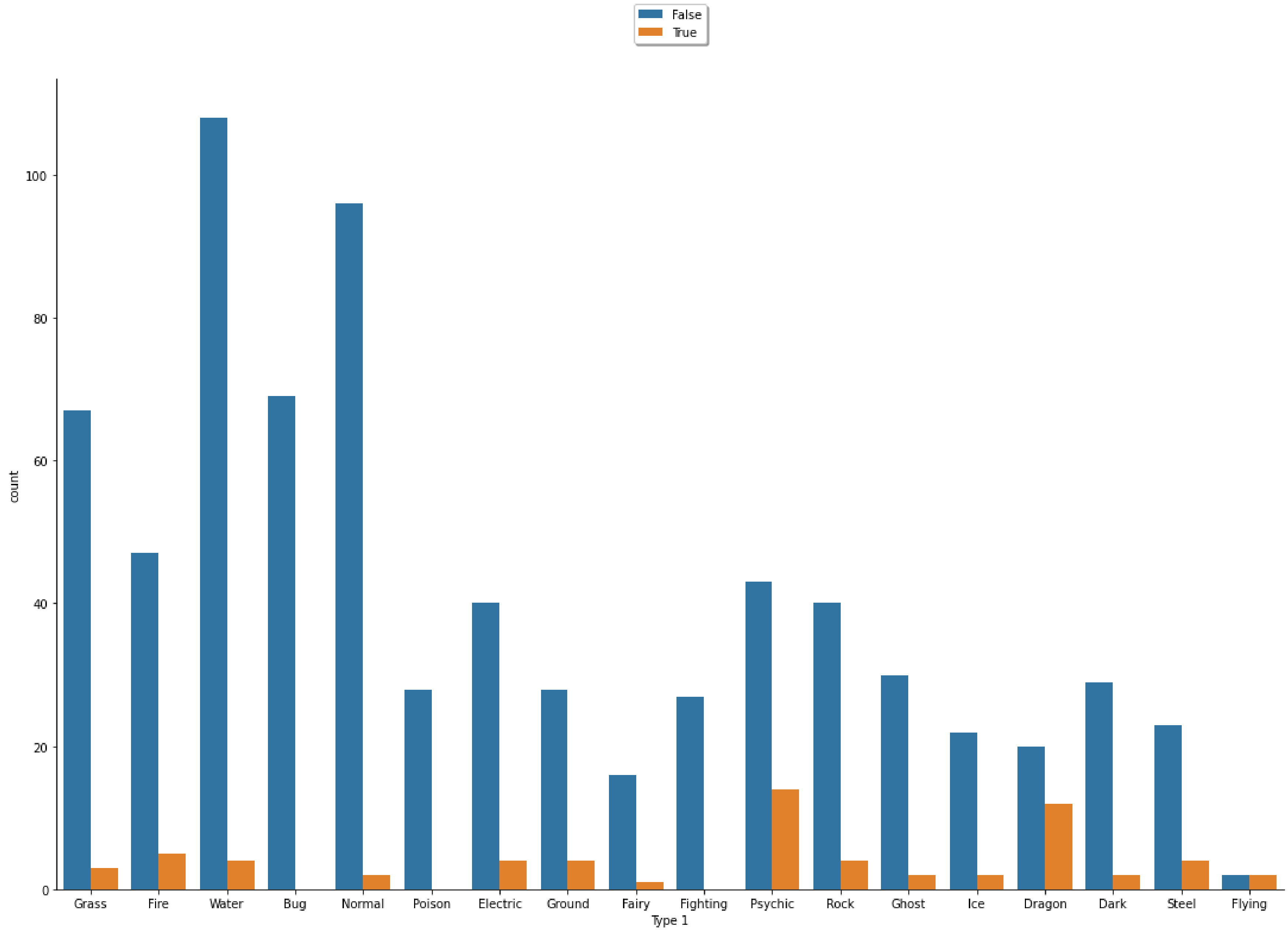
Type 1	Legendary	#	Name	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
Electric	False	40	40	40	40	40	40	40	40	40	40	40
	True	4	4	4	4	4	4	4	4	4	4	4
Fairy	False	16	16	16	16	16	16	16	16	16	16	16
	True	1	1	1	1	1	1	1	1	1	1	1
Fighting	False	27	27	27	27	27	27	27	27	27	27	27
Fire	False	47	47	47	47	47	47	47	47	47	47	47
	True	5	5	5	5	5	5	5	5	5	5	5
Flying	False	2	2	2	2	2	2	2	2	2	2	2
	True	2	2	2	2	2	2	2	2	2	2	2
Ghost	False	30	30	30	30	30	30	30	30	30	30	30
	True	2	2	2	2	2	2	2	2	2	2	2
Grass	False	67	67	67	67	67	67	67	67	67	67	67
	True	3	3	3	3	3	3	3	3	3	3	3
Ground	False	28	28	28	28	28	28	28	28	28	28	28
	True	4	4	4	4	4	4	4	4	4	4	4
Ice	False	22	22	22	22	22	22	22	22	22	22	22
	True	2	2	2	2	2	2	2	2	2	2	2
Normal	False	96	96	96	96	96	96	96	96	96	96	96
	True	2	2	2	2	2	2	2	2	2	2	2
Poison	False	28	28	28	28	28	28	28	28	28	28	28
Psychic	False	43	43	43	43	43	43	43	43	43	43	43
	True	14	14	14	14	14	14	14	14	14	14	14
Rock	False	40	40	40	40	40	40	40	40	40	40	40
	True	4	4	4	4	4	4	4	4	4	4	4

Type 1	Legendary	#	Name	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
Steel	False	23	23	23	23	23	23	23	23	23	23	23
	True	4	4	4	4	4	4	4	4	4	4	4
Water	False	108	108	108	108	108	108	108	108	108	108	108
	True	4	4	4	4	4	4	4	4	4	4	4

In [29]:

```
bar2= sys.catplot(x='Type 1',
                  data=pk,
                  kind='count',
                  hue='Legendary',
                  height=10,
                  aspect=1.5,
                  legend=False).set_axis_labels('Type 1')

bar2.ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.1), shadow=True)
plt.show()
```



IN TYPE 1,

Bug, Fighting, Poison Pokemon doesnt have legendary pokemon.

Psychic pokemon has highest legendary pokemon.

Dark, Flying, Ghost, Ice, Normal Pokemon has 2 legendary pokemon each.

Water, Steel, Rock, Ground, Elecric Pokemon has 4 legendary pokemon each.

Fariy pokemon has only one legendary pokemon.

In [30]:
```
pk.groupby(['Type 2','Legendary']).count()
```

Out[30]:

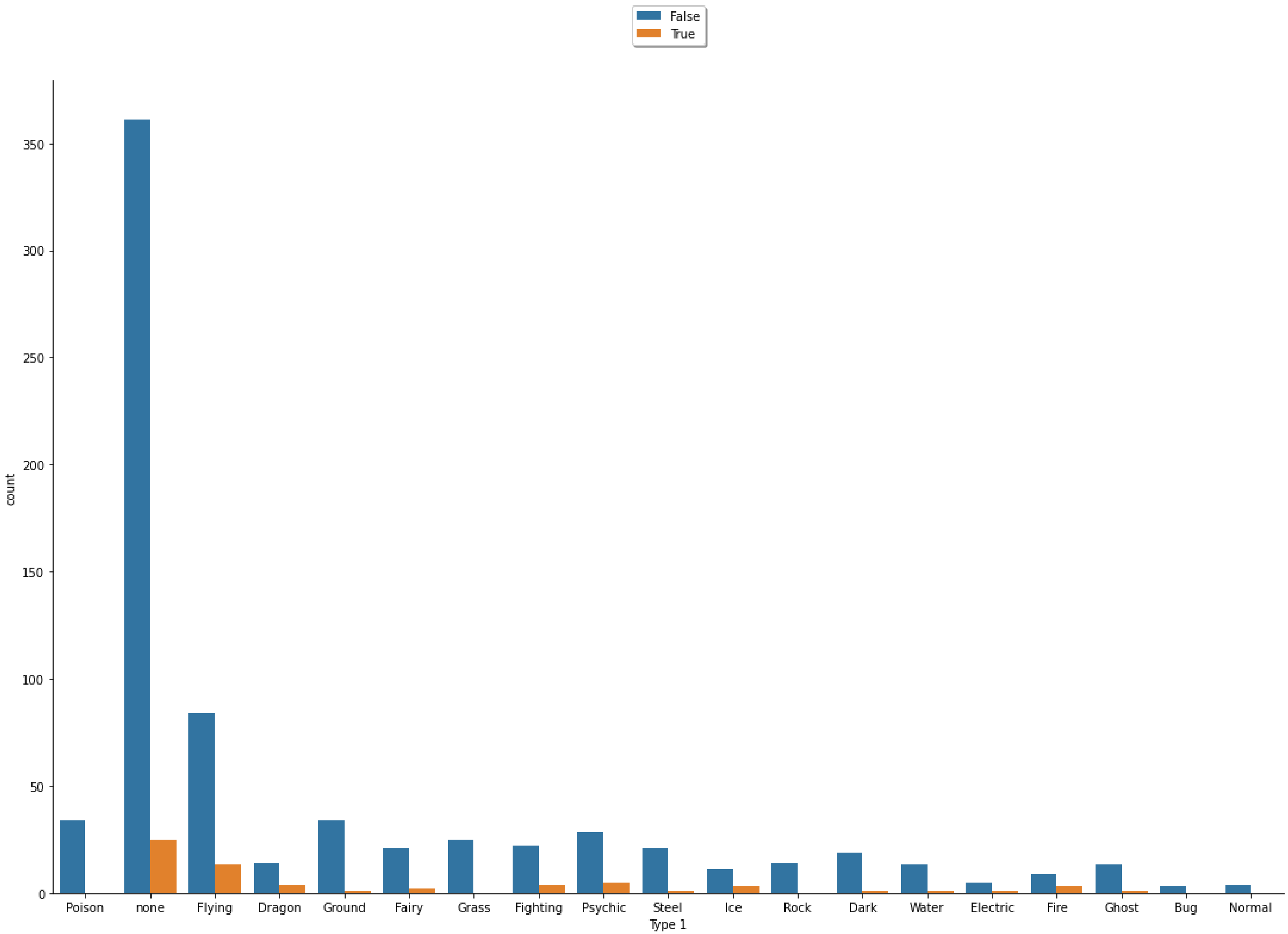
Type 2	Legendary	#	Name	Type 1	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
Bug	False	3	3	3	3	3	3	3	3	3	3	3
Dark	False	19	19	19	19	19	19	19	19	19	19	19
	True	1	1	1	1	1	1	1	1	1	1	1
Dragon	False	14	14	14	14	14	14	14	14	14	14	14
	True	4	4	4	4	4	4	4	4	4	4	4
Electric	False	5	5	5	5	5	5	5	5	5	5	5
	True	1	1	1	1	1	1	1	1	1	1	1
Fairy	False	21	21	21	21	21	21	21	21	21	21	21
	True	2	2	2	2	2	2	2	2	2	2	2
Fighting	False	22	22	22	22	22	22	22	22	22	22	22
	True	4	4	4	4	4	4	4	4	4	4	4
Fire	False	9	9	9	9	9	9	9	9	9	9	9
	True	3	3	3	3	3	3	3	3	3	3	3
Flying	False	84	84	84	84	84	84	84	84	84	84	84
	True	13	13	13	13	13	13	13	13	13	13	13
Ghost	False	13	13	13	13	13	13	13	13	13	13	13

Type 2	Legendary	#	Name	Type 1	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
	True	1	1	1	1	1	1	1	1	1	1	1
Grass	False	25	25	25	25	25	25	25	25	25	25	25
Ground	False	34	34	34	34	34	34	34	34	34	34	34
	True	1	1	1	1	1	1	1	1	1	1	1
Ice	False	11	11	11	11	11	11	11	11	11	11	11
	True	3	3	3	3	3	3	3	3	3	3	3
Normal	False	4	4	4	4	4	4	4	4	4	4	4
Poison	False	34	34	34	34	34	34	34	34	34	34	34
Psychic	False	28	28	28	28	28	28	28	28	28	28	28
	True	5	5	5	5	5	5	5	5	5	5	5
Rock	False	14	14	14	14	14	14	14	14	14	14	14
Steel	False	21	21	21	21	21	21	21	21	21	21	21
	True	1	1	1	1	1	1	1	1	1	1	1
Water	False	13	13	13	13	13	13	13	13	13	13	13
	True	1	1	1	1	1	1	1	1	1	1	1
none	False	361	361	361	361	361	361	361	361	361	361	361
	True	25	25	25	25	25	25	25	25	25	25	25

In [31]:

```
bar3= sys.catplot(x='Type 2',
                  data=pk,
                  kind='count',
                  hue='Legendary',
                  height=10,
                  aspect=1.5,
                  legend=False).set_axis_labels('Type 1')

bar3.ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.1), shadow=True)
plt.show()
```



In type 2,

None value has 25 legendary pokemon.

Flying pokemon has 13 legendary pokemon.

Bug, grass, normal, poison,rock pokemon doesnt have legendary pokemon.

Dark, Electric, Ghost, Ground, Steel, Water has 1 legendary pokemon each.

Fire and Ice pokemon has 3 legendary pokemon.

Dragon and Fighting pokemon has 4 legendary pokemon.

In [32]:

```
pk['HP'].describe().astype(int)
```

Out[32]:

```
count    800
mean      69
std       25
min        1
25%       50
50%       65
75%       80
max      255
Name: HP, dtype: int32
```

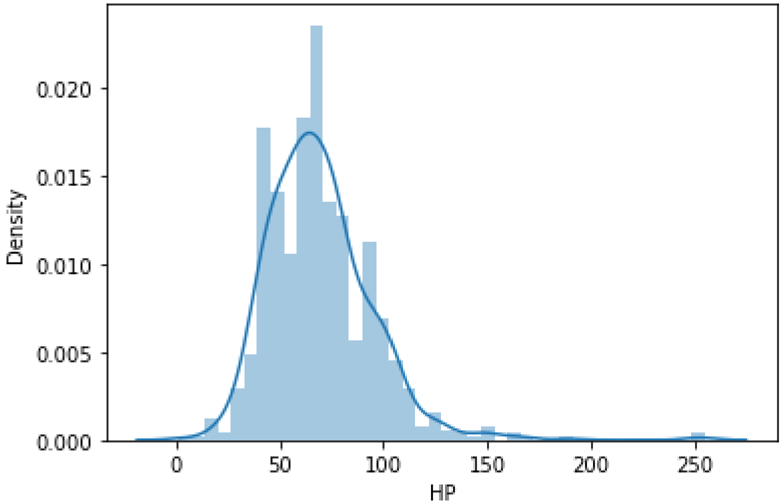
In [33]:

```
sys.distplot(pk.HP)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[33]:

```
<AxesSubplot:xlabel='HP', ylabel='Density'>
```



Max. HP is 255

Min HP is 1

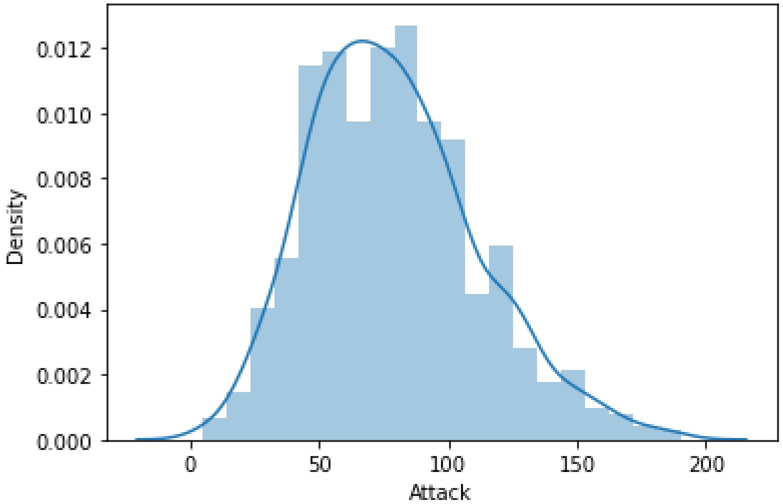
In [34]:
```
pk['Attack'].describe().astype(int)
```

Out[34]:
```
count    800
mean      79
std       32
min        5
25%       55
50%       75
75%      100
max      190
Name: Attack, dtype: int32
```

In [35]:
```
sys.distplot(pk.Attack)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[35]:
```
<AxesSubplot:xlabel='Attack', ylabel='Density'>
```



Maximum Attack points is 190

Minimum attack point is 5

In [36]:

```
pk['Defense'].describe().astype(int)
```

Out[36]:

```
count    800
mean      73
std       31
min        5
25%       50
50%       70
75%       90
max      230
Name: Defense, dtype: int32
```

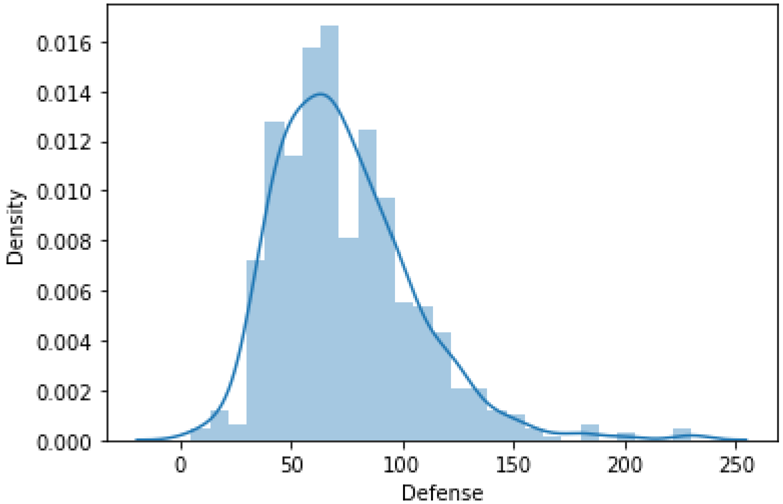
In [37]:

```
sys.distplot(pk.Defense)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[37]:

```
<AxesSubplot:xlabel='Defense', ylabel='Density'>
```



Maximum Defence points is 230

Minimum Defence point is 5

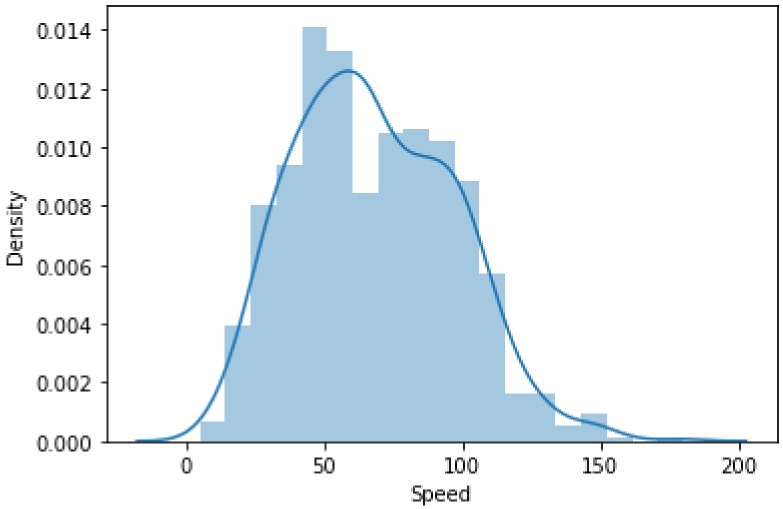
In [38]:
```
pk['Speed'].describe().astype(int)
```

Out[38]:
```
count    800
mean      68
std       29
min        5
25%       45
50%       65
75%       90
max      180
Name: Speed, dtype: int32
```

In [39]:
```
sys.distplot(pk.Speed)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[39]:
```
<AxesSubplot:xlabel='Speed', ylabel='Density'>
```



Maximum Speed of pokemon is 180.

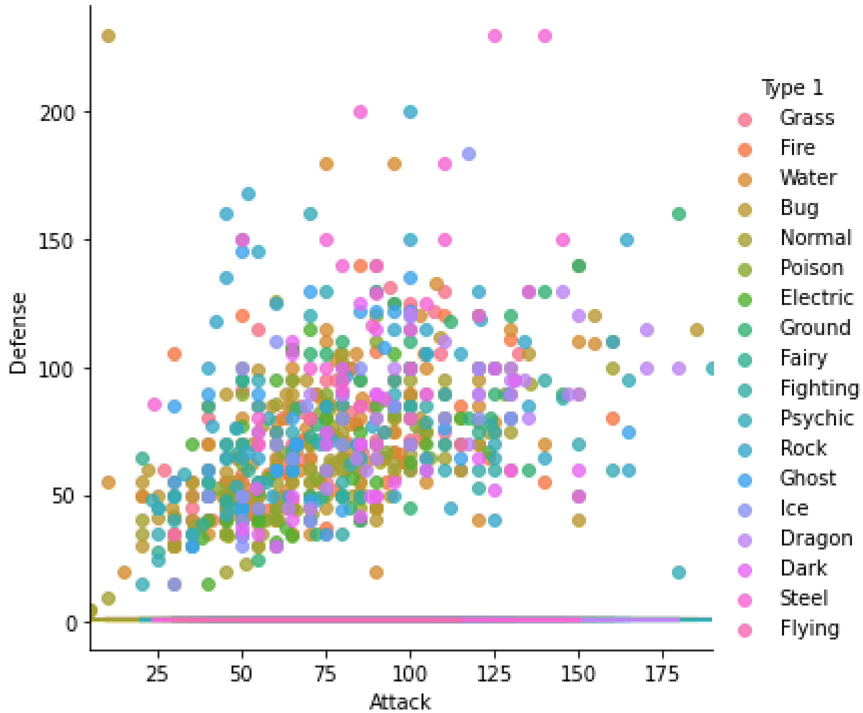
Minimum speed of pokemon is 5.

In [40]:

```
#TYPE 1: ATTACK VS DEFENCE:
sys.lmplot(x="Attack",
           y="Defense",
           data=pk,
           hue='Type 1',
           logistic=True)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
```

Out[40]: `<seaborn.axisgrid.FacetGrid at 0x1a81e1f4310>`



In type 1,

Almost all pokemons are trained in attack and defense.

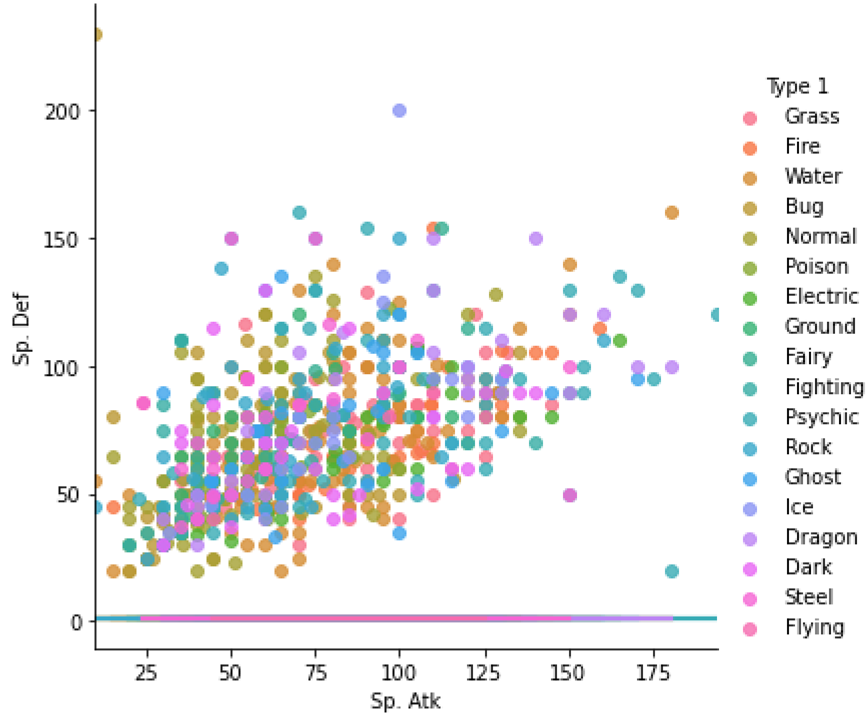
Outliers in this explains, few are highly defensive and attacking pokemon.

In [136…

```
# TYPE 1: Sp. Atk VS Sp. Def:
sys.lmplot(x="Sp. Atk",
           y="Sp. Def",
           data=pk,
           hue='Type 1',
           logistic=True)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
```

Out[136… <seaborn.axisgrid.FacetGrid at 0x1a8225c2310>



In type 1,

Almost all pokemons are trained in sp. attack and sp. defense.

Outliers in this explains, few are highly trained in sp.def and sp. attack.

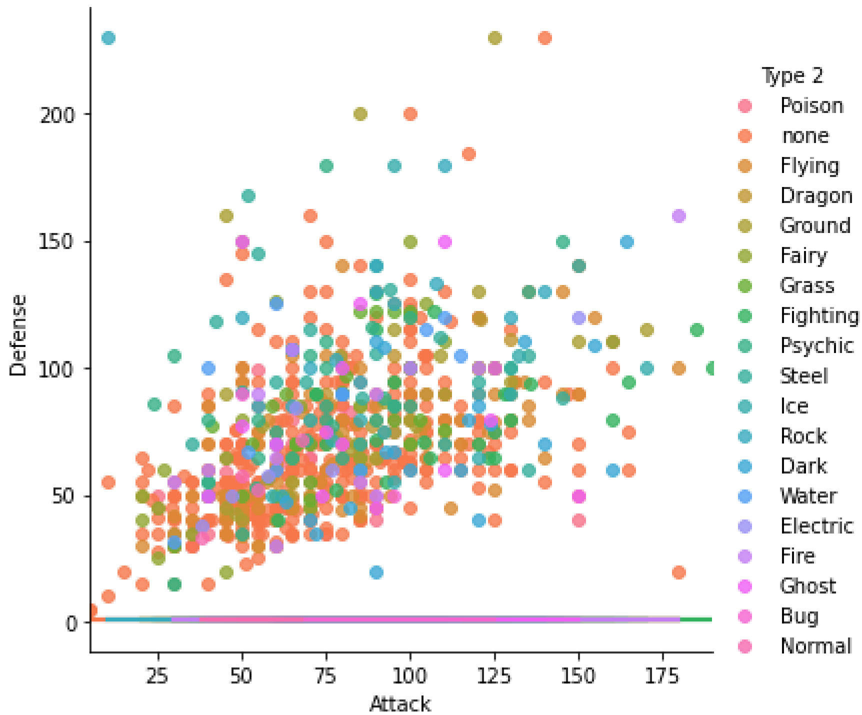
In [41]:

```
#TYPE 2: ATTACK VS DEFENCE:
sys.lmplot(x="Attack",
           y="Defense",
           data=pk,
           hue='Type 2',
           logistic=True)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
```

Out[41]: <seaborn.axisgrid.FacetGrid at 0x1a81efa9df0>



In type 2,

Almost all pokemons are trained in attack and defense.

Outliers in this explains, few are highly defensive and attacking pokemon.

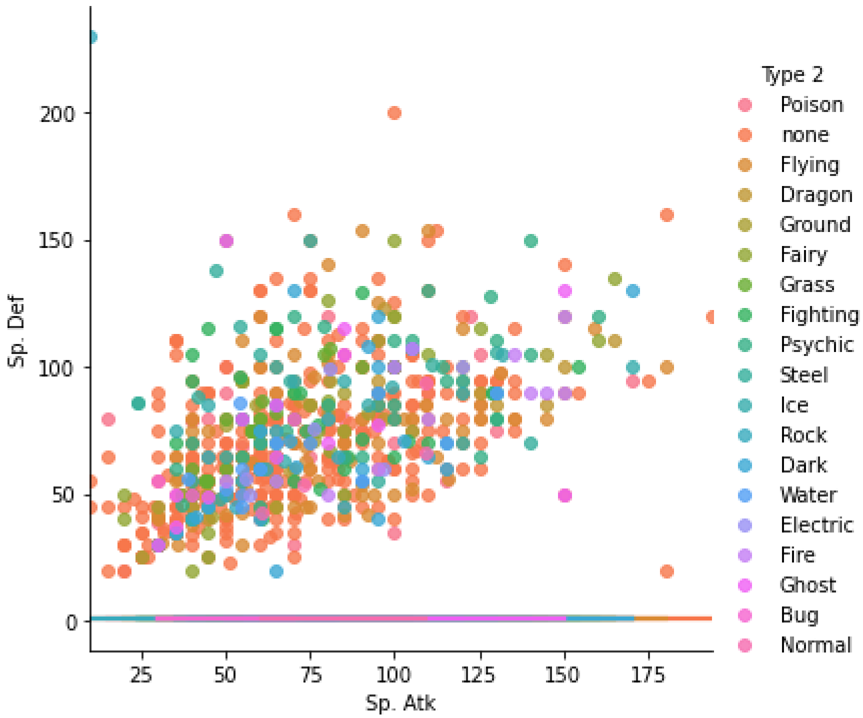
In [42]:

```
# TYPE 2: Sp. Atk VS Sp. Def:
sys.lmplot(x="Sp. Atk",
           y="Sp. Def",
```

```
            data=pk,
            hue='Type 2',
            logistic=True)
```

```
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
C:\Users\aditi\anaconda_7june\lib\site-packages\statsmodels\genmod\families\links.py:188: RuntimeWarning: overflow encoun
tered in exp
  t = np.exp(-z)
```

Out[42]: <seaborn.axisgrid.FacetGrid at 0x1a81efe86d0>



In type 2,

Almost all pokemons are trained in sp. attack and sp. defense.

Outliers in this explains, few are highly trained in sp.def and sp. attack.

CENTRAL TENDENCY:

In [43]:
```
#mean
pk.mean().astype(int)
```

Out[43]:
```
#              362
Total          435
HP              69
Attack          79
Defense         73
Sp. Atk         72
Sp. Def         71
Speed           68
Generation       3
Legendary        0
dtype: int32
```

The mean of HP is 69.

The mean of Defence is 73.

The mean of Attack is 79.

The mean of Special Defence is 72.

The mean of Special Attack is 71.

The mean of speed is 68.

In [44]:
```
#median
pk.median().astype(int)
```

Out[44]:
```
#              364
Total          450
HP              65
Attack          75
Defense         70
Sp. Atk         65
Sp. Def         70
Speed           65
Generation       3
```

```
Legendary       0
dtype: int32
```

The median of HP is 65.

The median of Attack is 75.

The median of Defence is 70.

The median of Special Attack is 65.

The median of Special Defence is 70.

The median of speed is 65.

MEASURES OF SPREAD

In [45]:

```
#variance
pk.var().astype(int)
```

Out[45]:

```
#             43407
Total         14391
HP              652
Attack         1053
Defense         972
Sp. Atk        1070
Sp. Def         774
Speed           844
Generation        2
Legendary         0
dtype: int32
```

The variation of HP is 652.

The variation of Attack is 1053.

The variation of Defence is 972.

The variation of Special Attack is 1070.

The variation of Special Defence is 774.

The variation of speed is 844.

In [46]:

```
#standard deviation
pk.std().astype(int)
```

Out[46]:

```
#              208
Total          119
HP              25
Attack          32
Defense         31
Sp. Atk         32
Sp. Def         27
Speed           29
Generation       1
Legendary        0
dtype: int32
```

From mean, HP is deviated by 25

From mean, Attack is deviated by 32.

From mean, Defence is deviated by 31.

From mean, Special Attack is deviated by 32.

From mean, Special Defence is deviated by 27.

From mean, speed is deviated by 29.

In [47]:

```
#describe:
pk.describe().astype(int)
```

Out[47]:

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	800	800	800	800	800	800	800	800	800
mean	362	435	69	79	73	72	71	68	3
std	208	119	25	32	31	32	27	29	1
min	1	180	1	5	5	10	20	5	1
25%	184	330	50	55	50	49	50	45	2
50%	364	450	65	75	70	65	70	65	3
75%	539	515	80	100	90	95	90	90	5
max	721	780	255	190	230	194	230	180	6

Quantiles:

TOTAL:

In [48]:
```
qt1=pk['Total'].quantile(0.25)
qt1
```

Out[48]: 330.0

In [49]:
```
qt2=pk['Total'].quantile(0.50)
qt2
```

Out[49]: 450.0

In [50]:
```
qt3=pk['Total'].quantile(0.75)
qt3
```

Out[50]: 515.0

In [51]:
```
#inter=quantile region:
IQR_T= qt3-qt1
IQR_T
```

Out[51]: 185.0

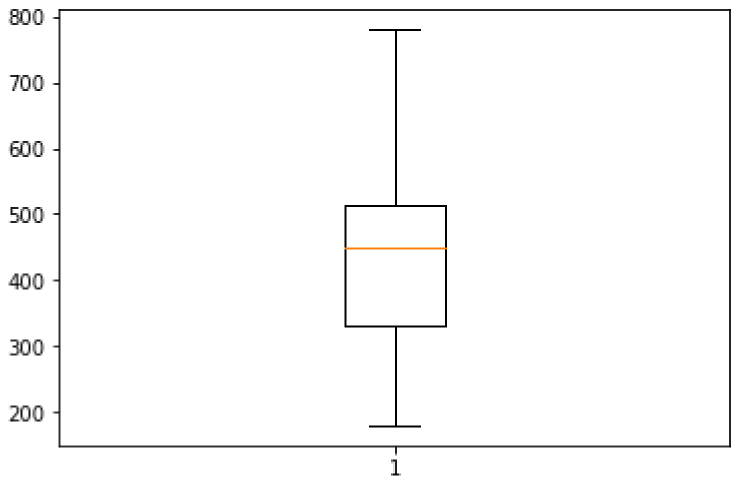
In [52]:
```
lbt = qt1-1.5*IQR_T
lbt
```

Out[52]: 52.5

In [53]:
```
ubt = qt3+1.5*IQR_T
ubt
```

Out[53]: 792.5

In [54]:
```
plt.boxplot(pk['Total']);
```



NO OUTLIERS ARE PRESENT

In [55]:
```
pk['Total'].skew()
```

Out[55]: 0.1525299233953993

total is fairly symmetrical

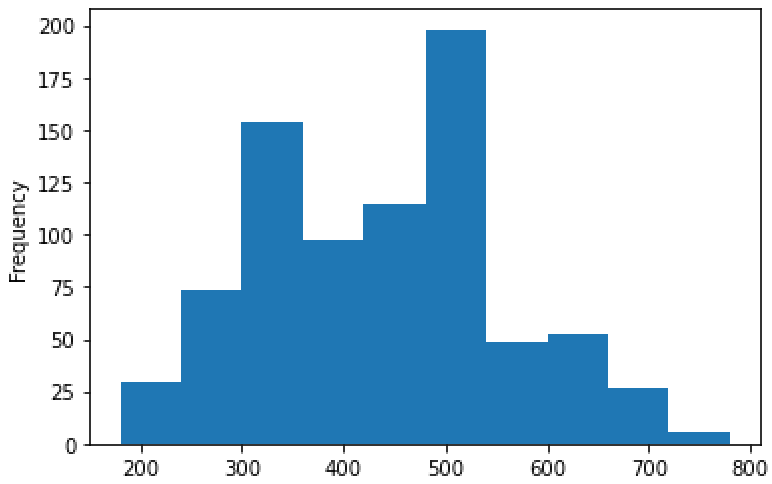
In [56]:
```
pk['Total'].kurt()
```

Out[56]: -0.5074607103228463

kurtosis is negative. hence curve maybe flat

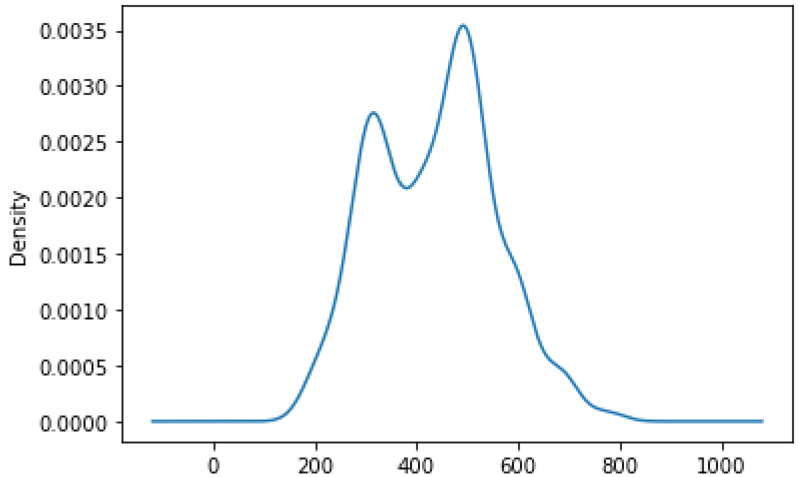
In [57]:
```
pk['Total'].plot(kind='hist')
```

Out[57]: <AxesSubplot:ylabel='Frequency'>



In [58]:
```
pk['Total'].plot(kind='kde')
```

Out[58]: <AxesSubplot:ylabel='Density'>



TOTAL is platykurtic in nature.

HP

In [59]:
```
qh1=pk['HP'].quantile(0.25)
qh1
```

Out[59]: 50.0

In [60]:
```
qh2=pk['HP'].quantile(0.50)
qh2
```

Out[60]: 65.0

In [61]:
```
qh3=pk['HP'].quantile(0.75)
qh3
```

Out[61]: 80.0

In [62]:
```
#inter=quantile region:
IQR_H=qh3-qh1
IQR_H
```

Out[62]: 30.0

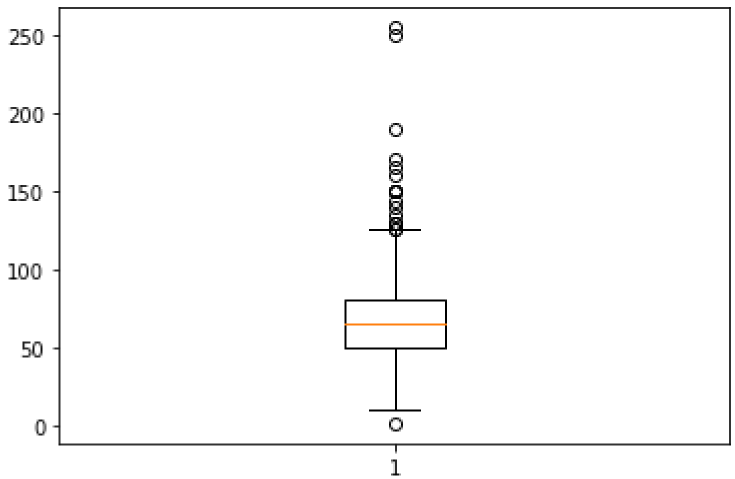
In [63]:
```
lbh = qh1-1.5*IQR_H
lbh
```

Out[63]: 5.0

In [64]:
```
ubh = qh3+1.5*IQR_H
ubh
```

Out[64]: 125.0

In [65]:
```
plt.boxplot(pk['HP']);
```



HEAVY OUTLIERS ARE PRESENT

In [66]:
```
pk['HP'].skew()
```

Out[66]: 1.5682243758418617

HP is highly skewed

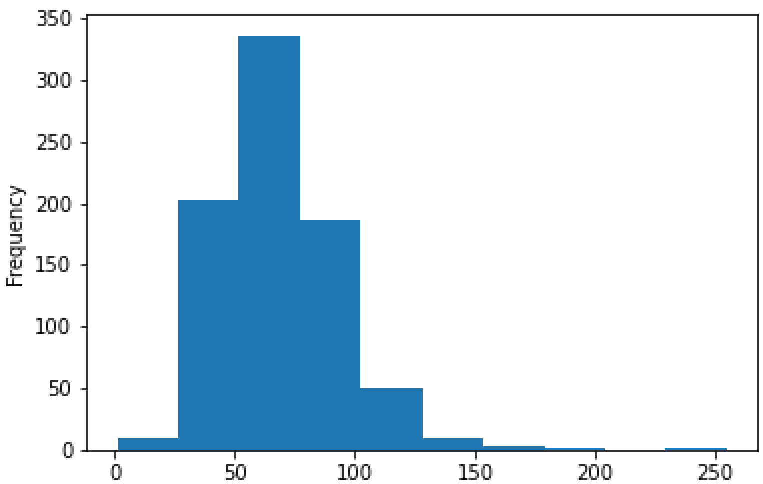
In [67]:
```
pk['HP'].kurt()
```

Out[67]: 7.232078374375156

kurtosis is positive and greater than 3. Hence heavy outliers are present and peak is pointy.

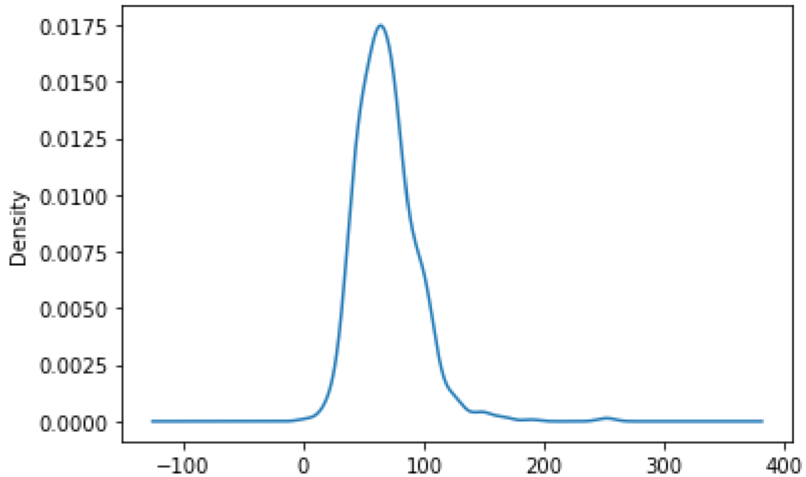
In [68]:
```
pk['HP'].plot(kind='hist')
```

Out[68]: <AxesSubplot:ylabel='Frequency'>



In [69]:
```
pk['HP'].plot(kind='kde')
```

Out[69]: <AxesSubplot:ylabel='Density'>



HP IS PLATYKURTIC IN NATURE.

ATTACK

In [70]:
```
qa1=pk['Attack'].quantile(0.25)
qa1
```

Out[70]: 55.0

In [71]:
```
qa2=pk['Attack'].quantile(0.50)
qa2
```

Out[71]: 75.0

In [72]:
```
qa3=pk['Attack'].quantile(0.75)
qa3
```

Out[72]: 100.0

In [73]:
```
#inter=quantile region:
IQR_A=qa3-qa1
IQR_A
```

Out[73]: 45.0

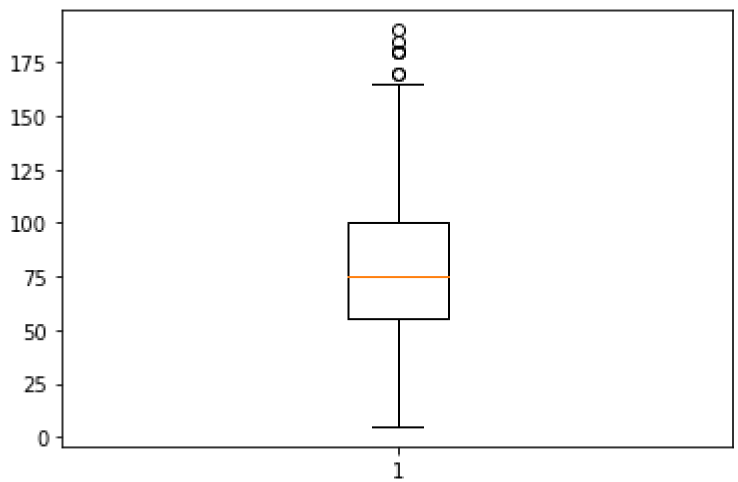
In [74]:
```
lba = qa1-1.5*IQR_A
lba
```

Out[74]: -12.5

In [75]:
```
uba = qa3+1.5*IQR_A
uba
```

Out[75]: 167.5

In [76]:
```
plt.boxplot(pk['Attack']);
```



OUTLIERS ARE PRESENT

In [77]:
```
pk['Attack'].skew()
```

Out[77]: 0.5516137480269772

Attack is moderately skewed

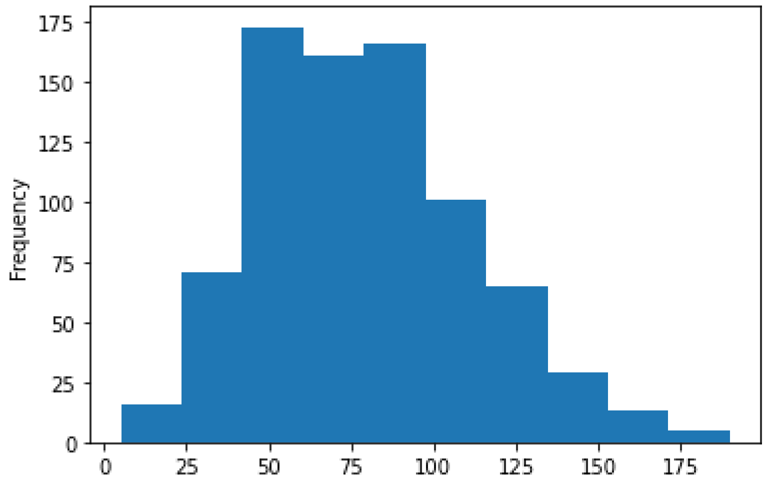
In [78]:
```
pk['Attack'].kurt()
```

Out[78]: 0.1697173149230906

kurtosis is positive and less than 3. hence low outliers are present and peak is pointy

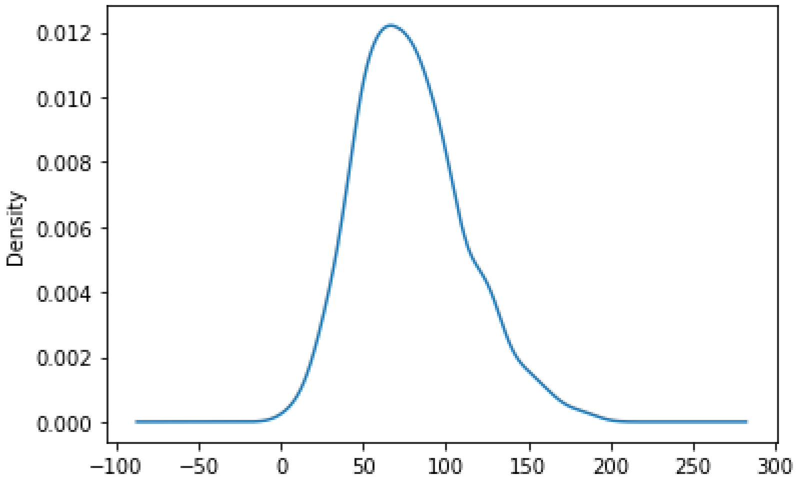
In [79]:
```
pk['Attack'].plot(kind='hist')
```

Out[79]: <AxesSubplot:ylabel='Frequency'>



In [80]:
```
pk['Attack'].plot(kind='kde')
```

Out[80]: <AxesSubplot:ylabel='Density'>



ATTACK IS PLATYKURTIC IN NATURE.

DEFENSE

In [81]:
```
qd1=pk['Defense'].quantile(0.25)
qd1
```

Out[81]: 50.0

In [82]:
```
qd2=pk['Defense'].quantile(0.50)
qd2
```

Out[82]: 70.0

In [83]:
```
qd3=pk['Defense'].quantile(0.75)
qd3
```

Out[83]: 90.0

In [84]:
```
#inter=quantile region:
IQR_D= qd3-qd1
IQR_D
```

Out[84]:
```
40.0
```

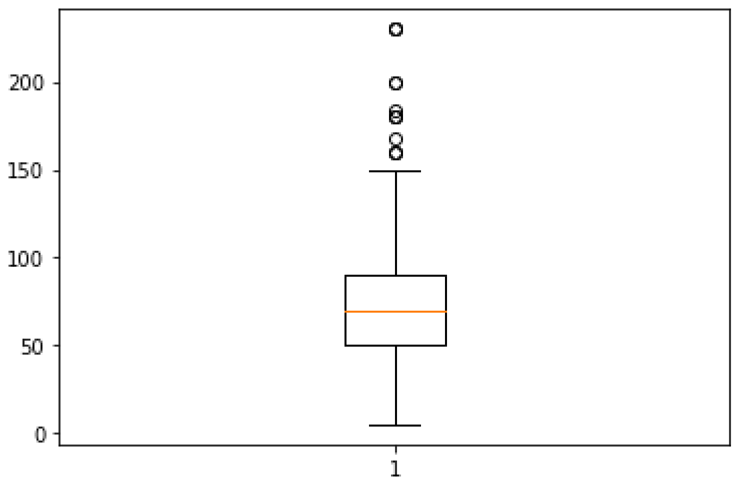
In [85]:
```
lbd = qd1-1.5*IQR_D
lbd
```

Out[85]:
```
-10.0
```

In [86]:
```
ubd = qd3+1.5*IQR_D
ubd
```

Out[86]:
```
150.0
```

In [87]:
```
plt.boxplot(pk['Defense']);
```



OUTLIERS ARE PRESENT

In [88]:
```
pk['Defense'].skew()
```

Out[88]:
```
1.1559123029560856
```

Defense is highly skewed

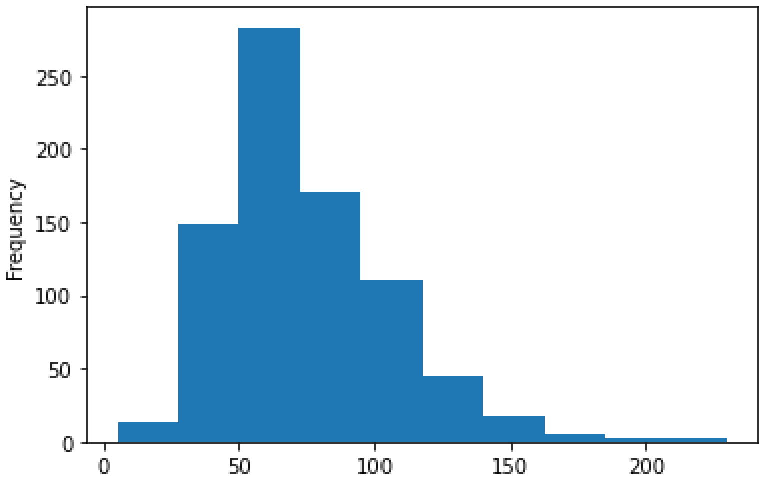
In [89]:
```
pk['Defense'].kurt()
```

Out[89]: 2.726260359939344

Kurtosis is positive and less than 3. Hence peak is pointy and has lack of outliers.

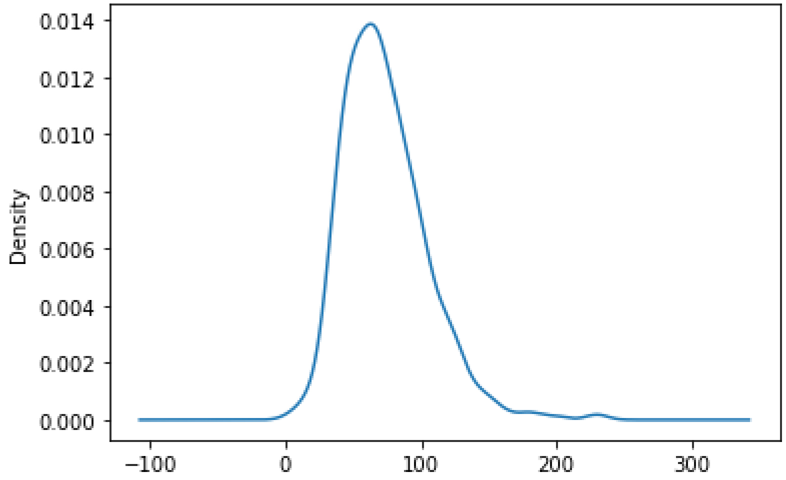
In [90]:
```
pk['Defense'].plot(kind='hist')
```

Out[90]: <AxesSubplot:ylabel='Frequency'>



In [91]:
```
pk['Defense'].plot(kind='kde')
```

Out[91]: <AxesSubplot:ylabel='Density'>



DEFENSE IS PLATYKURTIC IN NATURE.

SP.ATK

In [92]:
```
qsa1=pk['Sp. Atk'].quantile(0.25)
qsa1
```

Out[92]: 49.75

In [93]:
```
qsa2= pk['Sp. Atk'].quantile(0.50)
qsa2
```

Out[93]: 65.0

In [94]:
```
qsa3= pk['Sp. Atk'].quantile(0.75)
qsa3
```

Out[94]: 95.0

In [95]:
```
#inter=quantile region:
IQR_SA= qsa3-qsa1
IQR_SA
```

Out[95]: 45.25

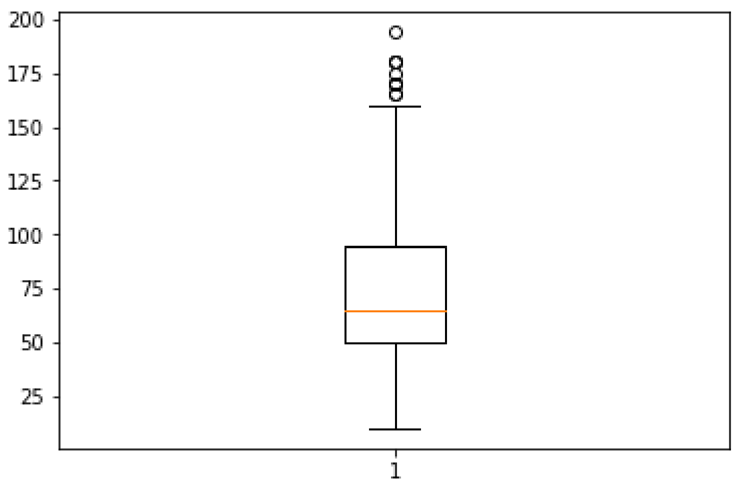
In [96]:
```
lbsa = qsa1-1.5*IQR_SA
lbsa
```

Out[96]: -18.125

In [97]:
```
ubsa = qsa3+1.5*IQR_SA
ubsa
```

Out[97]: 162.875

In [98]:
```
plt.boxplot(pk['Sp. Atk']);
```



FEW OUTLIERS ARE PRESENT

In [99]:
```
pk['Sp. Atk'].skew()
```

Out[99]: 0.7446624978300574

Sp. Atk is slightly skewed

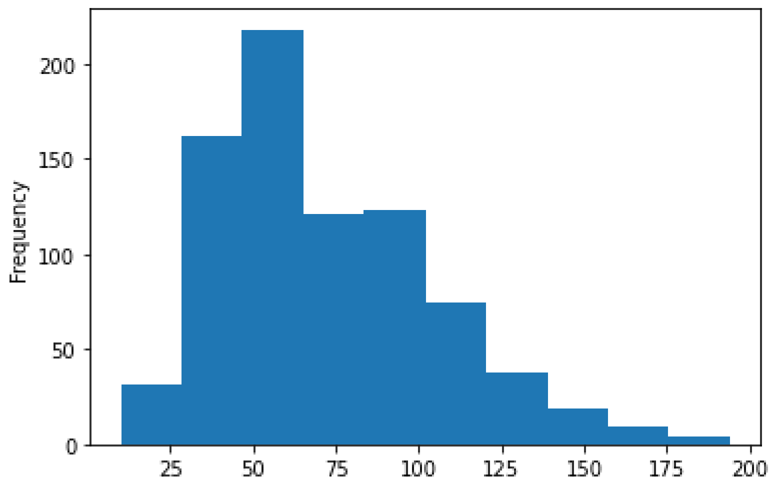
In [100…
```
pk['Sp. Atk'].kurt()
```

Out[100… 0.29789366073147416

Kurtosis is positive and approx to zero. hence lack of outliers and peak is pointy

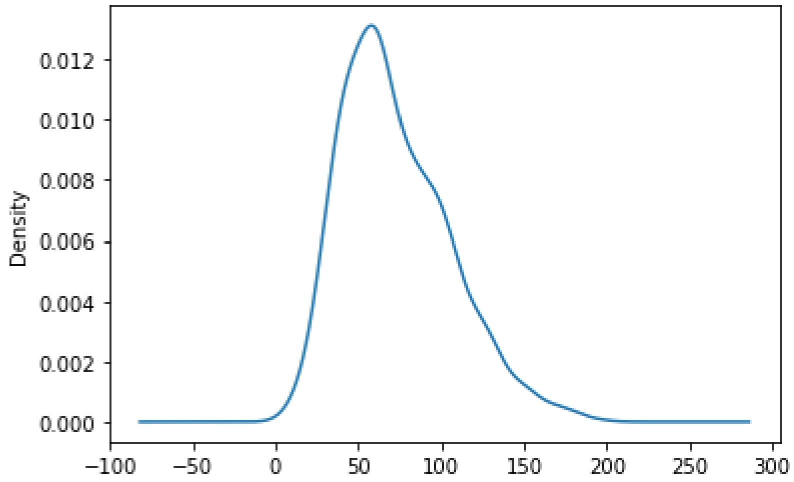
In [101…
```
pk['Sp. Atk'].plot(kind='hist')
```

Out[101… <AxesSubplot:ylabel='Frequency'>



In [102…
```
pk['Sp. Atk'].plot(kind='kde')
```

Out[102… <AxesSubplot:ylabel='Density'>



SP. ATK IS SLIGHTLY POINTY TO NORMAL DISTRIBUTION IN NATURE.

SP.DEF

In [103…

```
qsd1=pk['Sp. Def'].quantile(0.25)
qsd1
```

Out[103… 50.0

In [104…

```
qsd2=pk['Sp. Def'].quantile(0.50)
qsd2
```

Out[104… 70.0

In [105…

```
qsd3=pk['Sp. Def'].quantile(0.75)
qsd3
```

Out[105… 90.0

In [106…

```
#inter=quantile region:
IQR_SD= qsd3-qsd1
IQR_SD
```

Out[106… 40.0

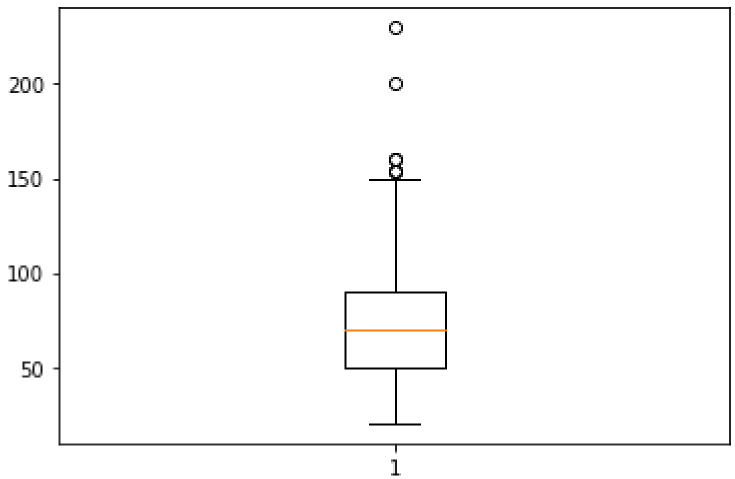
In [107…
```
lbsd = qt1-1.5*IQR_SD
lbsd
```

Out[107… 270.0

In [108…
```
ubsd = qsd3+1.5*IQR_SD
ubsd
```

Out[108… 150.0

In [109…
```
plt.boxplot(pk['Sp. Def']);
```



OUTLIERS ARE PRESENT

In [110…
```
pk['Sp. Def'].skew()
```

Out[110… 0.8540186115468782

Sp. def is moderately skewed

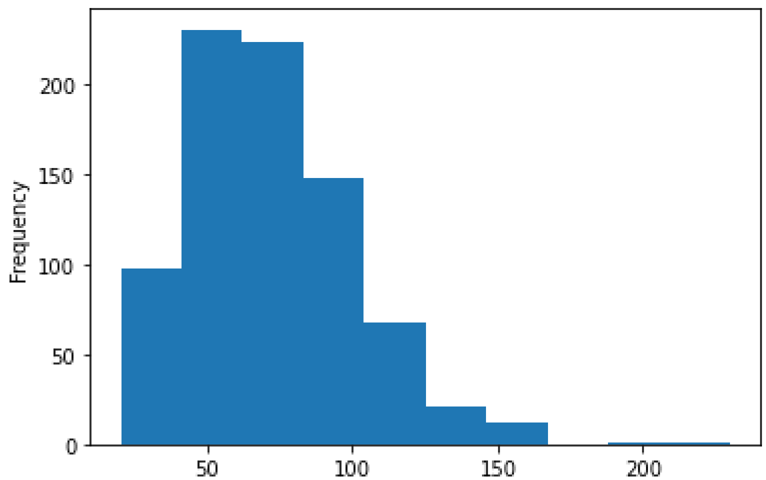
In [111…
```
pk['Sp. Def'].kurt()
```

Out[111…
```
1.628394056784738
```

Kurtosis is positive and less than 3. Hence peak is pointy and lack of outliers.

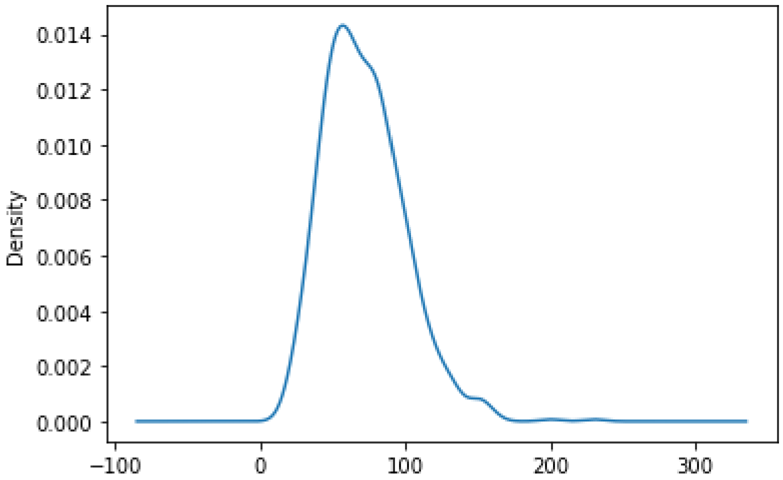
In [112…
```
pk['Sp. Def'].plot(kind='hist')
```

Out[112…
```
<AxesSubplot:ylabel='Frequency'>
```



In [113…
```
pk['Sp. Def'].plot(kind='kde')
```

Out[113…
```
<AxesSubplot:ylabel='Density'>
```



SP. DEF IS PLATYKURTIC IN NATURE.

SPEED

In [114…
```
qs1=pk['Speed'].quantile(0.25)
qs1
```
Out[114… 45.0

In [115…
```
qs2=pk['Speed'].quantile(0.50)
qs2
```
Out[115… 65.0

In [116…
```
qs3= pk['Speed'].quantile(0.75)
qs3
```
Out[116… 90.0

In [117…
```
#inter=quantile region:
IQR_S=qs3-qs1
IQR_S
```

Out[117… 45.0

In [118…

```
lbs = qt1-1.5*IQR_S
lbs
```

Out[118… 262.5

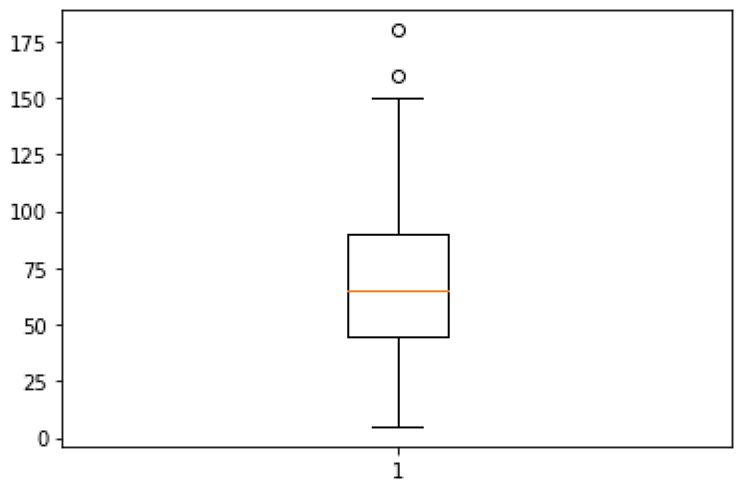
In [119…

```
ubs = qt3+1.5*IQR_S
ubs
```

Out[119… 582.5

In [120…

```
plt.boxplot(pk['Speed']);
```



OUTLIERS ARE PRESENT

In [121…

```
pk['Speed'].skew()
```

Out[121… 0.35793329506082994

Speed is fairly skewed

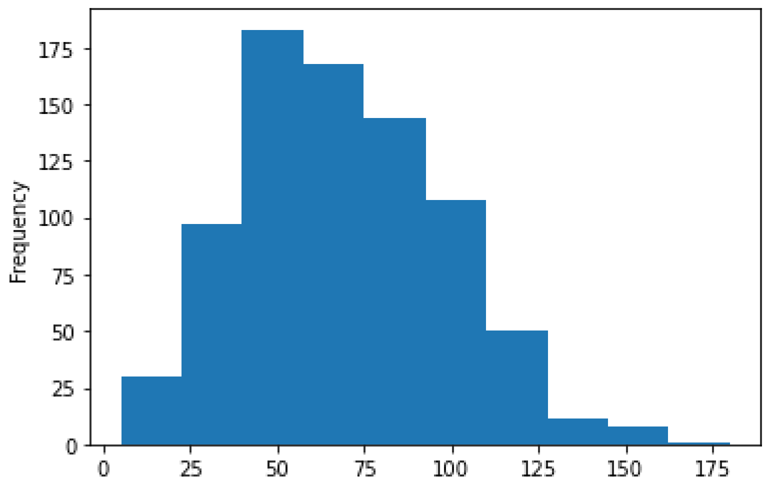
In [122…
```
pk['Speed'].kurt()
```

Out[122…
```
-0.2364366728440488
```

Kurtosis is negative and less than 3. Hence peak is flat and has lack of outliers

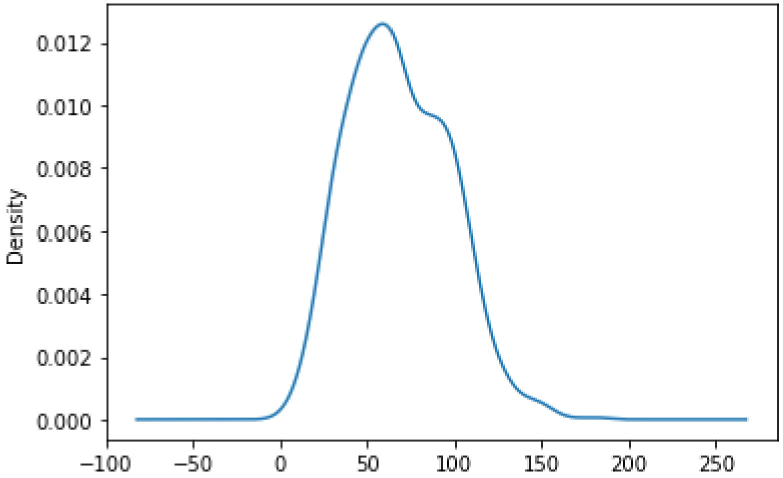
In [123…
```
pk['Speed'].plot(kind='hist')
```

Out[123…
```
<AxesSubplot:ylabel='Frequency'>
```



In [124…
```
pk['Speed'].plot(kind='kde')
```

Out[124…
```
<AxesSubplot:ylabel='Density'>
```



SPEED IS PLATYKURTIC IN NATURE.

GENERATION

In [125…
```
qg1=pk['Generation'].quantile(0.25)
qg1
```
Out[125… 2.0

In [126…
```
qg2=pk['Generation'].quantile(0.50)
qg2
```
Out[126… 3.0

In [127…
```
qg3= pk['Generation'].quantile(0.75)
qg3
```
Out[127… 5.0

In [128…
```
#inter=quantile region:
IQR_G=qg3-qg1
IQR_G
```

Out[128… 3.0

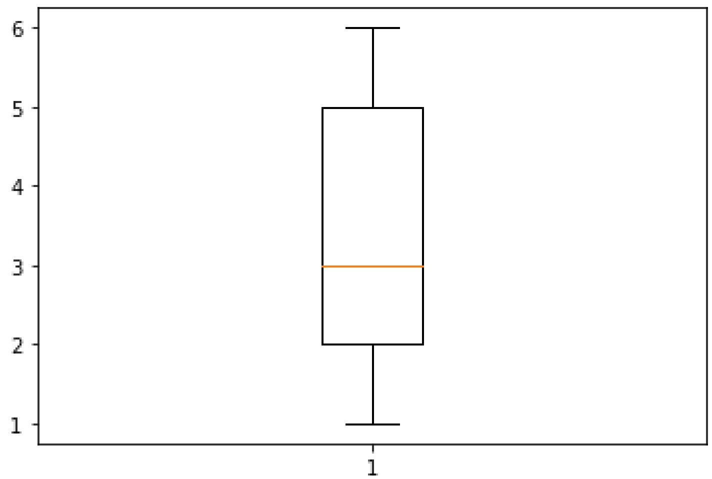
In [129…
```
lbg = qg1-1.5*IQR_G
lbg
```

Out[129… -2.5

In [130…
```
ubg = qg3+1.5*IQR_G
ubg
```

Out[130… 9.5

In [131…
```
plt.boxplot(pk['Generation']);
```



NO OUTLIERS ARE PRESENT

In [132…
```
pk['Generation'].skew()
```

Out[132… 0.014258100279990539

Generation is fairly skewed

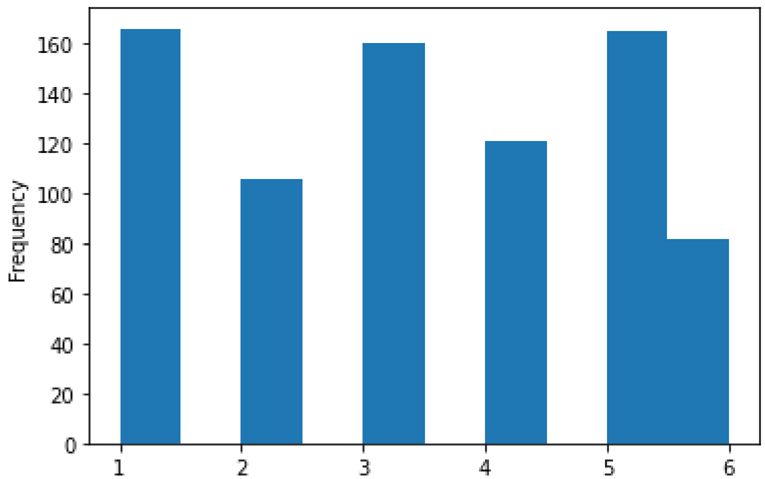
In [133…
```
pk['Generation'].kurt()
```

Out[133… -1.2395757575999116

Kurtosis is negative and less than 3. Hence the peak is pointy and has lack of outliers.

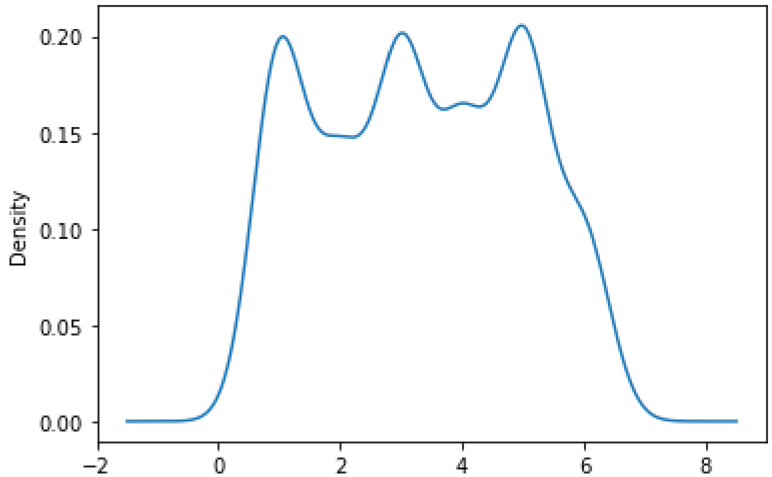
In [134…
```
pk['Generation'].plot(kind='hist')
```

Out[134… <AxesSubplot:ylabel='Frequency'>



In [135…
```
pk['Generation'].plot(kind='kde')
```

Out[135… <AxesSubplot:ylabel='Density'>



GENERATION IS PLATYKURTIC IN NATURE.

In [ ]: