Warum Pandas?

- Übung basierend auf Material von Prof. Dietzsch
- Pandas ist eine der leistungsfähigsten Bibliotheken, die man für Data Science in Python braucht
- Pandas bietet eine schnelle, flexible, leichte und intuitive Art der Manipulation von strukturierten/tabellenförmigen Daten (Data-Frames)
- Es ermöglicht die Arbeit mit großen Datensätzen und/oder mehreren Dateien zugleich.

Was zeichnet Pandas aus?

- `pandas` baut auf `numpy` auf und nutzt dieses als Backend
- Stellt eine High-Level Daten-Struktur zur Verfügung, die Spreadsheets von Tabellenkalkulationen ähnelt
- Verfügt über eingebaute Funktionen, um Daten zu bereinigen, zu filtern, zu gruppieren, zu kombinieren ... etc.
- Es bietet die einfache Möglichkeit `numpy` - und `scipy` -Funktionen auf Datentabellen anzuwenden

Wie ist Pandas zu installieren?

Über den Anaconda-Propmpt und Eingabe von `conda install pandas`

oder über den `Anaconda Navigator` und der Auswahl von `pandas` installieren

oder öffnene einer Kommandozeile und Eingabe von `pip install pandas`

In [1]:
```
# Importieren von pandas
import pandas as pd
#import numpy as np
```

Matplotlib (Standard)

https://matplotlib.org/api/pyplot_summary.html https://matplotlib.org/examples/ https://matplotlib.org/gallery.html

Im weiteren Verlauf werden wir noch einige verschiede Grafik-Typen sehen bzw. benutzen.

In [2]:
```
# Einlesen der Daten aus einer CSV-Datei in eine Variable
# import pandas.plotting

pokemon_data = pd.read_csv('pokemon_data.csv')

# Überprüfen Sie Anfang und Ende
pokemon_data
```

Out[2]:

#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Lege

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Lege
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	
...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	6	
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	6	
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6	
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6	
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6	

800 rows × 12 columns

`numpy.loadtxt` lädt Daten aus einem File in ein Numpy-Array. Standardmäßig wird von `float` -Werten ausgegangen.
Dokumentation: https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.loadtxt.html

Anmerkung: Es gibt auch ein `numpy.savetxt(filename, array, ...)` zum Abspeichern
Dokumentation: https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.savetxt.html

In [3]:
```
pokemon_data.columns              # Header der Daten
```

Out[3]:
```
Index(['#', 'Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. Atk',
       'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

In [4]:
```
pokemon_data.shape  # Größe der Datentabelle
```

Out[4]:
```
(800, 12)
```

In [5]:
```
pokemon_data.iloc[60] # Überprüfen des 60-sten Pokemons auf der Liste
```

Out[5]:
```
#                55
Name         Golduck
Type 1         Water
Type 2           NaN
HP                80
Attack            82
Defense           78
Sp. Atk           95
```

```
Sp. Def            80
Speed              85
Generation          1
Legendary       False
Name: 60, dtype: object
```

In [6]:
```
pokemon_data[['Name','Attack','Speed']] # Untertabelle anhand mehrerer selektierter
```

Out[6]:

	Name	Attack	Speed
0	Bulbasaur	49	45
1	Ivysaur	62	60
2	Venusaur	82	80
3	VenusaurMega Venusaur	100	80
4	Charmander	52	65
...	...	...	...
795	Diancie	100	50
796	DiancieMega Diancie	160	110
797	HoopaHoopa Confined	110	70
798	HoopaHoopa Unbound	160	80
799	Volcanion	110	70

800 rows × 3 columns

In [7]:
```
pokemon_data.iloc[300:400] # Slice-by-Rows
```

Out[7]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendar
300	277	Swellow	Normal	Flying	60	85	60	50	50	125	3	Fals
301	278	Wingull	Water	Flying	40	30	30	55	30	85	3	Fals
302	279	Pelipper	Water	Flying	60	50	100	85	70	65	3	Fals
303	280	Ralts	Psychic	Fairy	28	25	25	45	35	40	3	Fals
304	281	Kirlia	Psychic	Fairy	38	35	35	65	55	50	3	Fals
...	...	...	...	...	...	...	...	...	...	...	...	
395	361	Snorunt	Ice	NaN	50	50	50	50	50	50	3	Fals
396	362	Glalie	Ice	NaN	80	80	80	80	80	80	3	Fals
397	362	GlalieMega Glalie	Ice	NaN	80	120	80	120	80	100	3	Fals
398	363	Spheal	Ice	Water	70	40	50	55	50	25	3	Fals
399	364	Sealeo	Ice	Water	90	60	70	75	70	45	3	Fals

100 rows × 12 columns

Slicing:

- Linke Seite → Zeilen
- Rechte Seite → Spalten

In [8]:
```
pokemon_data.iloc[:,4:8] # alle Zeilen, Spalten 5 bis 9
```

Out[8]:

	HP	Attack	Defense	Sp. Atk
0	45	49	49	65
1	60	62	63	80
2	80	82	83	100
3	80	100	123	122
4	39	52	43	60
...	...	...	...	...
795	50	100	150	100
796	50	160	110	160
797	80	110	60	150
798	80	160	60	170
799	80	110	120	130

800 rows × 4 columns

In [9]:
```
pokemon_data.iloc[9:19,4:8] # Zeilen 10:20, Spalten 5 bis 9
```

Out[9]:

	HP	Attack	Defense	Sp. Atk
9	44	48	65	50
10	59	63	80	65
11	79	83	100	85
12	79	103	120	135
13	45	30	35	20
14	50	20	55	25
15	60	45	50	90
16	40	35	30	20
17	45	25	50	25
18	65	90	40	45

In [10]:
```
pokemon_data.sort_values('Attack') # Sortieren nach Werten
```

Out[10]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Le
488	440	Happiny	Normal	NaN	100	5	5	15	65	30	4	
121	113	Chansey	Normal	NaN	250	5	5	35	105	50	1	

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Le
230	213	Shuckle	Bug	Rock	20	10	230	10	230	5	2	
261	242	Blissey	Normal	NaN	255	10	10	75	135	55	2	
139	129	Magikarp	Water	NaN	20	10	55	15	20	80	1	
...	...	...	...	...	...	...	...	...	...	...	...	
429	386	DeoxysAttack Forme	Psychic	NaN	50	180	20	180	20	150	3	
426	384	RayquazaMega Rayquaza	Dragon	Flying	105	180	100	180	100	115	3	
424	383	GroudonPrimal Groudon	Ground	Fire	100	180	160	150	90	90	3	
232	214	HeracrossMega Heracross	Bug	Fighting	80	185	115	40	105	75	2	
163	150	MewtwoMega Mewtwo X	Psychic	Fighting	106	190	100	154	100	130	1	

800 rows × 12 columns

In [11]:
```
pokemon_data.describe()
```

Out[11]:

	#	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generati
count	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000
mean	362.813750	69.258750	79.001250	73.842500	72.820000	71.902500	68.277500	3.323
std	208.343798	25.534669	32.457366	31.183501	32.722294	27.828916	29.060474	1.661
min	1.000000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000	1.000
25%	184.750000	50.000000	55.000000	50.000000	49.750000	50.000000	45.000000	2.000
50%	364.500000	65.000000	75.000000	70.000000	65.000000	70.000000	65.000000	3.000
75%	539.250000	80.000000	100.000000	90.000000	95.000000	90.000000	90.000000	5.000
max	721.000000	255.000000	190.000000	230.000000	194.000000	230.000000	180.000000	6.000

In [12]:
```
#pokemon_data.dropna(axis = 0) # alle Zeilen mit NaN's löschen
#pokemon_data.dropna(axis = 1) # alle Spalten mit NaN's löschen
```

In [13]:
```
pokemon_data['Type 2'].value_counts() # "unique Werte" einer Spalte
```

Out[13]:
```
Flying      97
Ground      35
Poison      34
Psychic     33
Fighting    26
Grass       25
Fairy       23
Steel       22
```

```
Dark        20
Dragon      18
Water       14
Ghost       14
Ice         14
Rock        14
Fire        12
Electric     6
Normal       4
Bug          3
Name: Type 2, dtype: int64
```

In [14]:
```
pokemon_data['Type 2'].value_counts(normalize = True) # Anteil der "unique Werte"
```

Out[14]:
```
Flying      0.234300
Ground      0.084541
Poison      0.082126
Psychic     0.079710
Fighting    0.062802
Grass       0.060386
Fairy       0.055556
Steel       0.053140
Dark        0.048309
Dragon      0.043478
Water       0.033816
Ghost       0.033816
Ice         0.033816
Rock        0.033816
Fire        0.028986
Electric    0.014493
Normal      0.009662
Bug         0.007246
Name: Type 2, dtype: float64
```

In [15]:
```
pokemon_data['Attack'] + pokemon_data['Defense']
```

Out[15]:
```
0       98
1      125
2      165
3      223
4       95
      ...
795    250
796    270
797    170
798    220
799    230
Length: 800, dtype: int64
```

In [16]:
```
overall = pokemon_data['HP'] + pokemon_data['Attack'] + pokemon_data['Defense'] + po
pokemon_data['Overall'] = overall
pokemon_data.head(10)
```

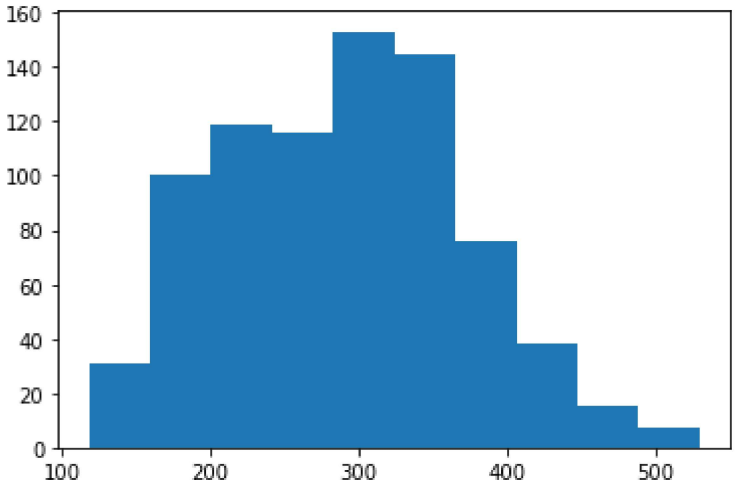
Out[16]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendar
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	Fals
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	Fals
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	Fals

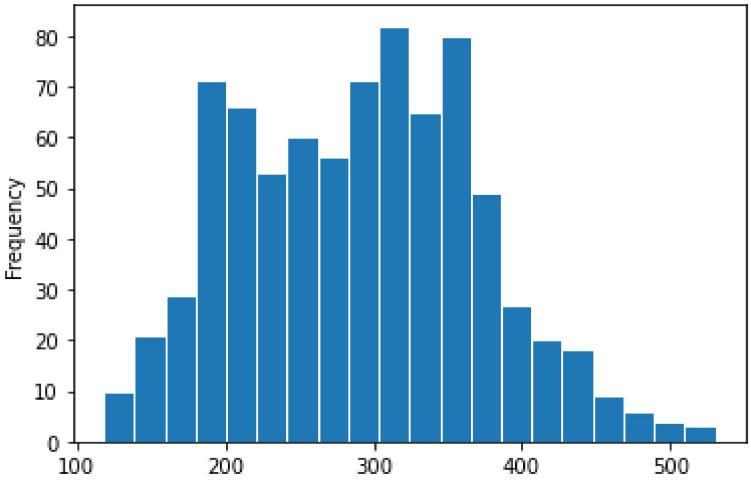
	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	Fals
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	Fals
5	5	Charmeleon	Fire	NaN	58	64	58	80	65	80	1	Fals
6	6	Charizard	Fire	Flying	78	84	78	109	85	100	1	Fals
7	6	CharizardMega Charizard X	Fire	Dragon	78	130	111	130	85	100	1	Fals
8	6	CharizardMega Charizard Y	Fire	Flying	78	104	78	159	115	100	1	Fals
9	7	Squirtle	Water	NaN	44	48	65	50	64	43	1	Fals

In [17]:
```
pokemon_data.to_csv('my_new_pokemon_table.txt', sep = '\t', index = False)
```

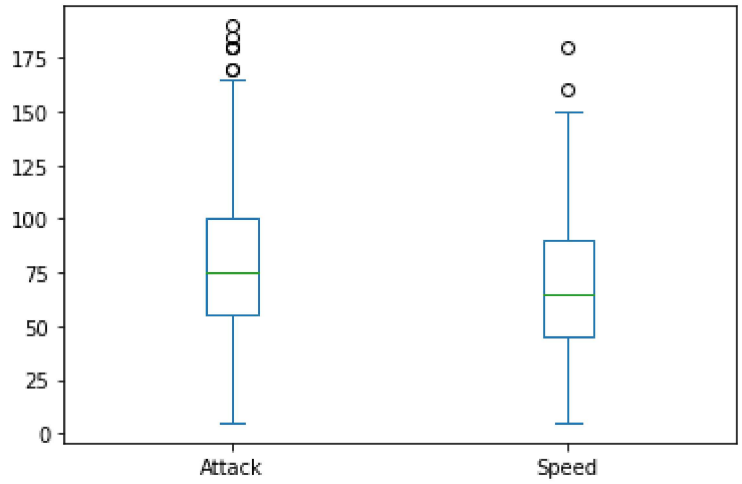
In [18]:
```
import matplotlib.pyplot as plt

plt.hist(pokemon_data['Overall']);
```



In [19]:
```
_ = pokemon_data['Overall'].plot(kind = 'hist', bins = 20, edgecolor = 'white')
```



In [20]:
```
_ = pokemon_data[['Attack','Speed']].plot(kind = 'box')
```



Aufgabe 1: Untersuchen Sie alle Stats mit Boxplots!

Aufgabe 2: Erstellen Sie eine Scatterplotmatrix!

Aufgabe 3: Erweitern Sie die Scatterplotmatrix um den "Type 1"-Typ (einfärben)!

Aufgabe 4: Stellen Sie den "Type 1"-Typ dar vs. Anzahl der Pokemons hierfür.

Aufgabe 5: Untersuchen Sie, wie sich Type1 zu Type2 verhält!

Aufgabe 6: Erzeugen Sie eine Grafik für Attack Efficiency, welche Angriffsgeschwindigkeit und Angriffsstärke sinnvoll visualisiert.

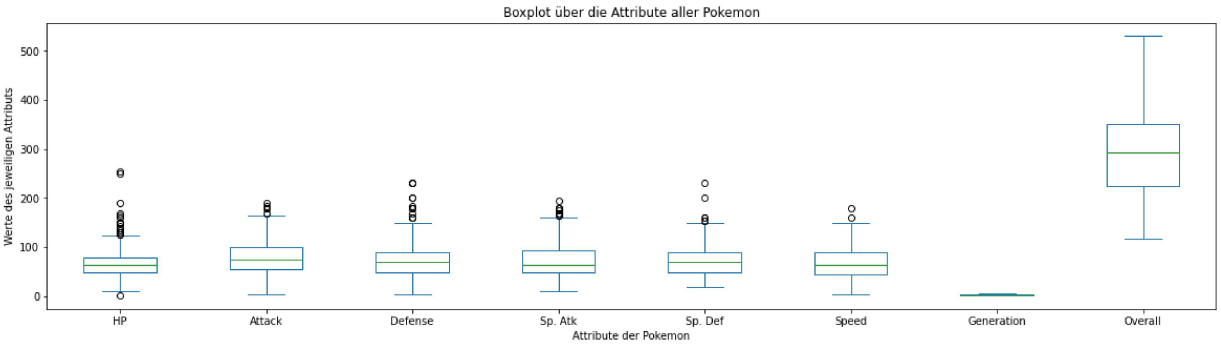
Aufgabe 7: Stellen Sie eine Übersicht über defensive und angriffsfreudige Pokemons dar.

Aufgabe 8: Untersuchen Sie, wie sich die Pokemons über die Generationen verändert haben in den relevanten Werten. Werden diese besser? Schlechter? Sind bestimmte Elemente "in"?

Aufgabe 9: Überlegen Sie sich mindestens 3 weitere sinnvolle Plots für diesen Datensatz.

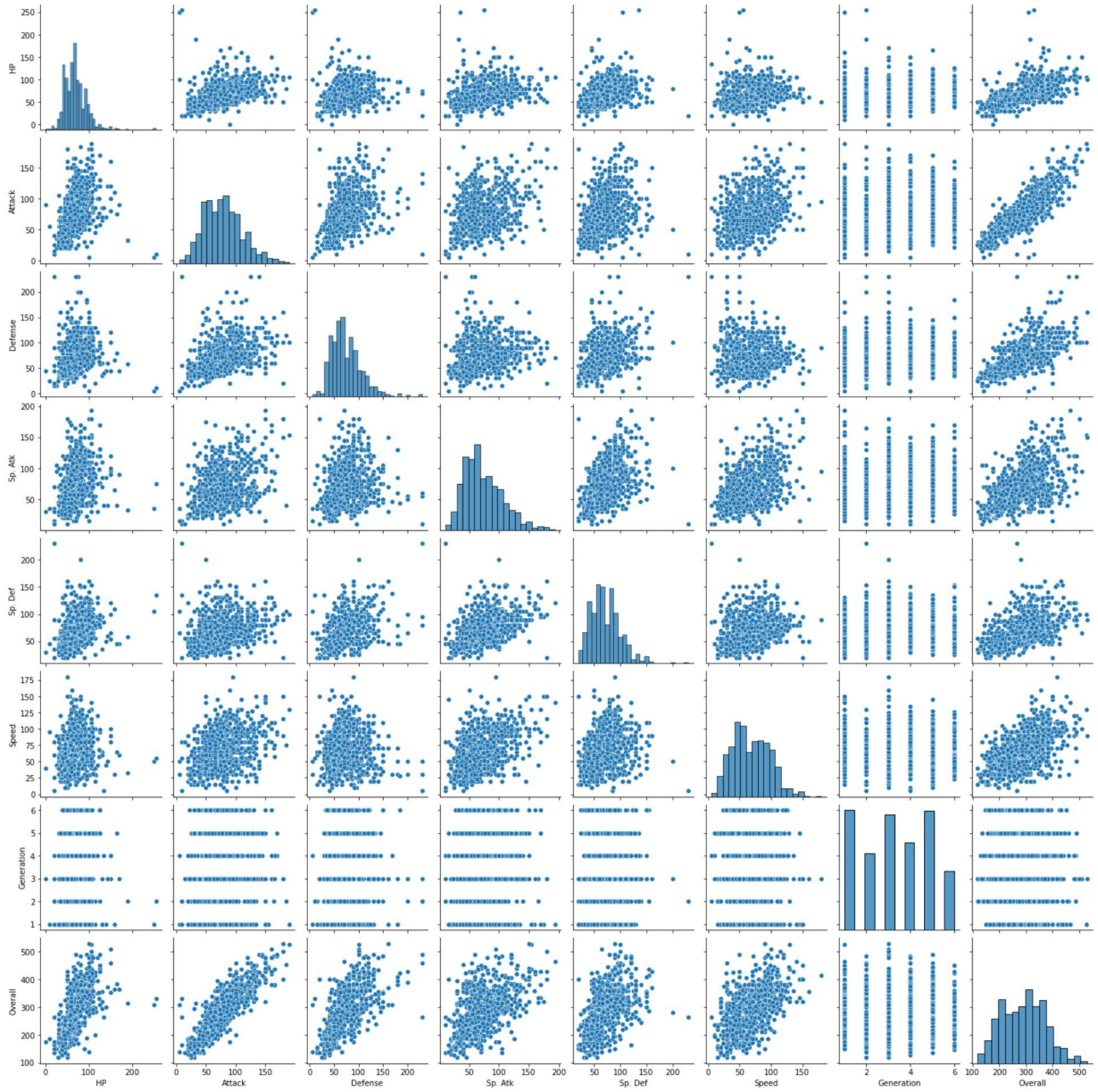
Aufgabe 1: Boxplot über alle Daten

In [40]:
```
_, axes = plt.subplots()
axes.set_title("Boxplot über die Attribute aller Pokemon")
axes.set_ylabel("Werte des jeweiligen Attributs")
axes.set_xlabel("Attribute der Pokemon")
_ = pokemon_data.drop(columns=["#"]).plot(kind="box", figsize=(20, 5), ax=axes)
```



Aufgabe 2: Scatterplotmatrix

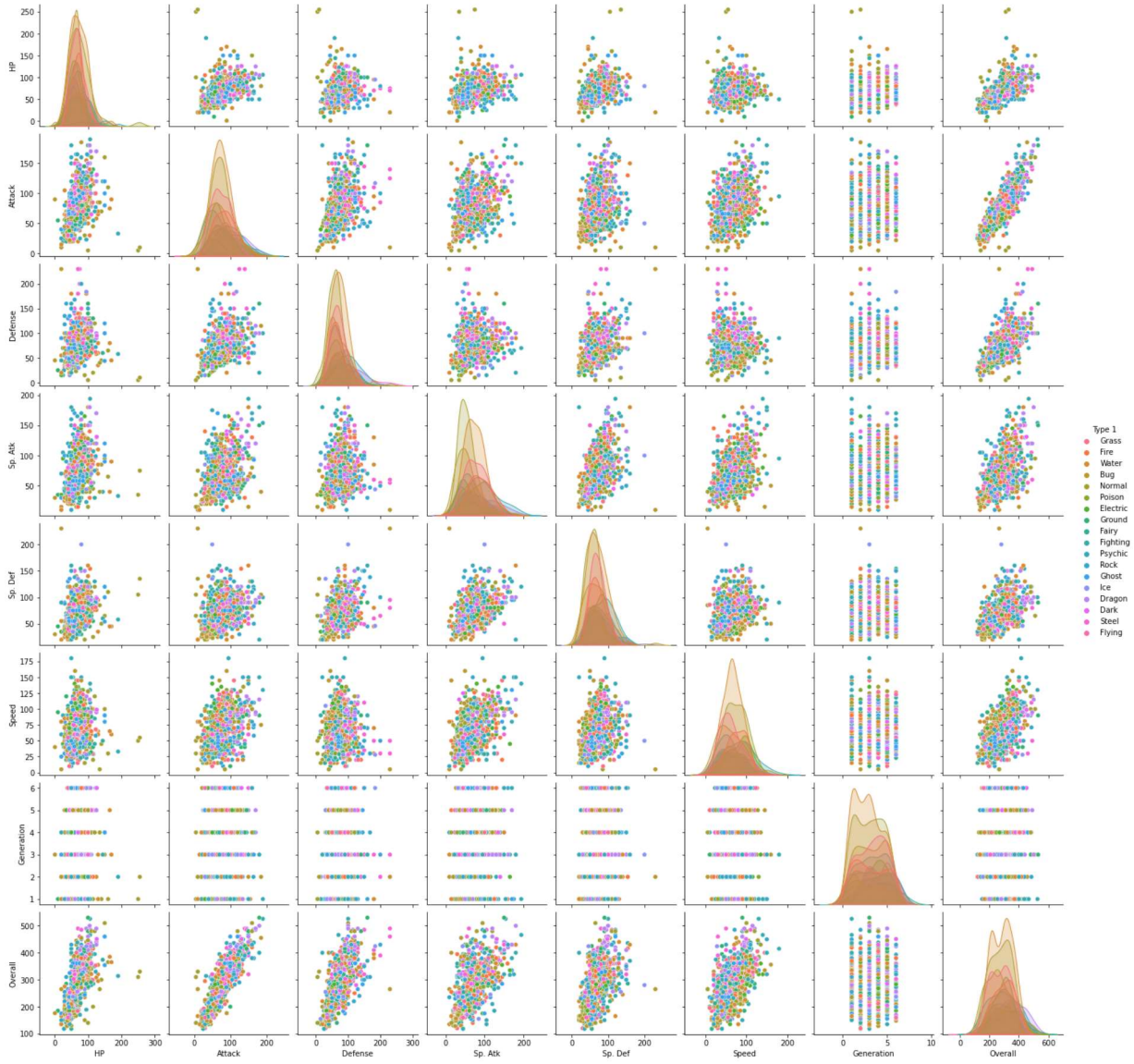
In [41]:
```
import seaborn as sns
_ = sns.pairplot(pokemon_data.drop(columns=["Legendary", "#"]))
```



Aufgabe 3: Scatterplot um "Type 1" erweitern

Recht einfach durch hue Parameter möglich

In [42]:
```
_ = sns.pairplot(pokemon_data.drop(columns=["Legendary", "#"]), hue="Type 1")
```

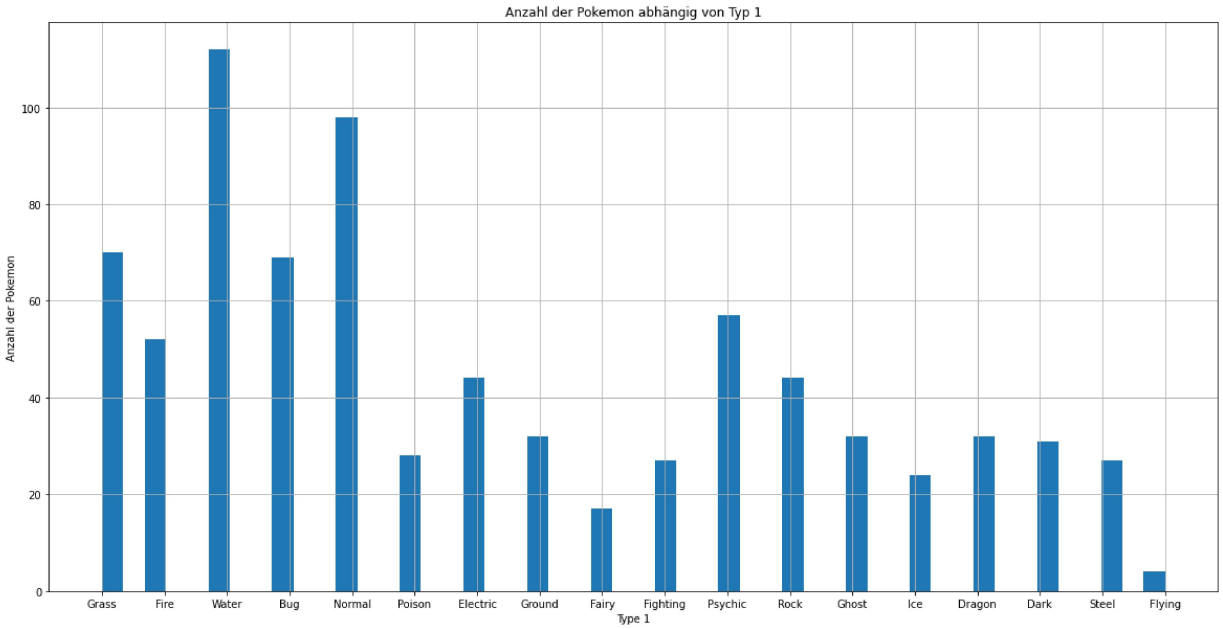


Aufgabe 4: Type 1 mit Anzahl der Pokemons mit Typ

-> Nutzen der `hist` Funktion

In [24]:

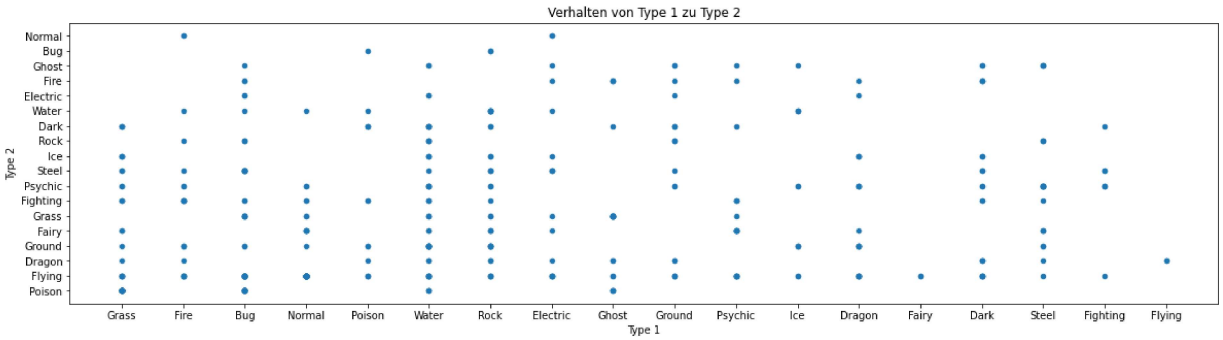
```
from matplotlib import pyplot as plt

_, axes = plt.subplots()
axes.set_xlabel("Type 1")
axes.set_ylabel("Anzahl der Pokemon")
axes.set_title("Anzahl der Pokemon abhängig von Typ 1")
_ = pokemon_data["Type 1"].hist(figsize=(20, 10), ax=axes, bins=50)
```



Aufgabe 5: Verhalten von Type 1 zu Type 2

In [137…

```
_, axes = plt.subplots()
axes.set_xlabel("Type 1")
axes.set_ylabel("Type 2")
axes.set_title("Verhalten von Type 1 zu Type 2")

_ = pokemon_data.dropna()[["Type 1", "Type 2"]].plot(kind="scatter", x="Type 1", y="
```



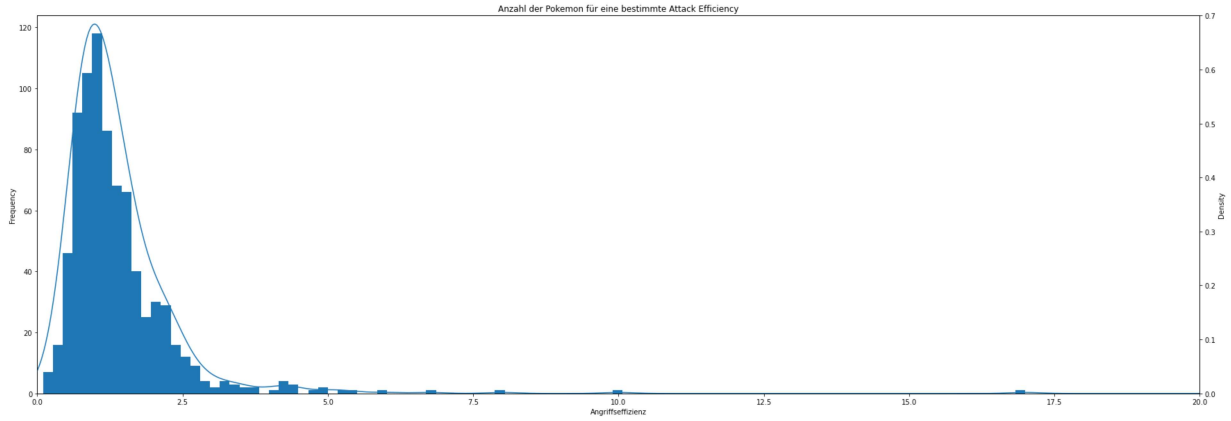
Aufgabe 6: Grafik für Attack Efficiency, gebildet aus Angriffsgeschwindigkeit und Angriffsstärke

Histogramm zeigt, dass viele Pokemon eine ähnliche Effizienz haben

In [150…

```
# Idee: Wie effizient ist ein einzelner Angriff --> Was geben die Werte genau an?
# https://bulbapedia.bulbagarden.net/wiki/Stat#Permanent_stats
_, axes = plt.subplots()
axes.set_title("Anzahl der Pokemon für eine bestimmte Attack Efficiency")
axes.set_xlabel("Angriffseffizienz")
_ = (pokemon_data["Attack"] / pokemon_data["Speed"]).plot(kind="hist", figsize=(30,

axes2 = axes.twinx()
axes2.set_ylabel("Density")
```

```
_ = (pokemon_data["Attack"] / pokemon_data["Speed"]).plot(kind="kde", figsize=(30, 1
```

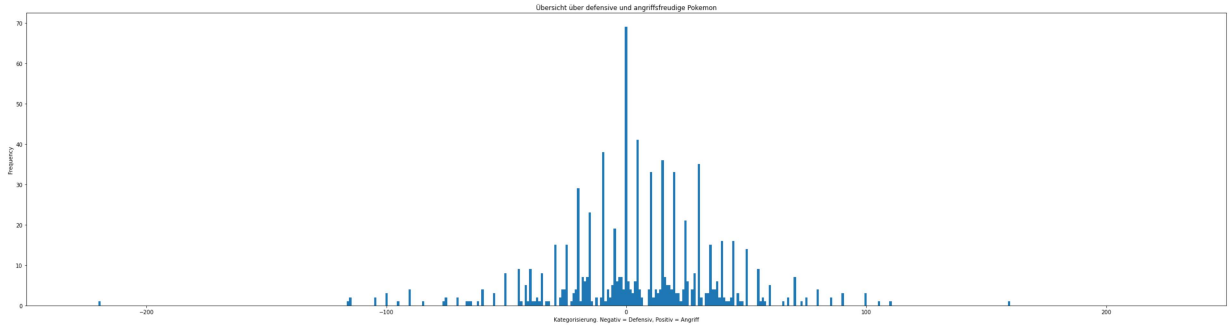


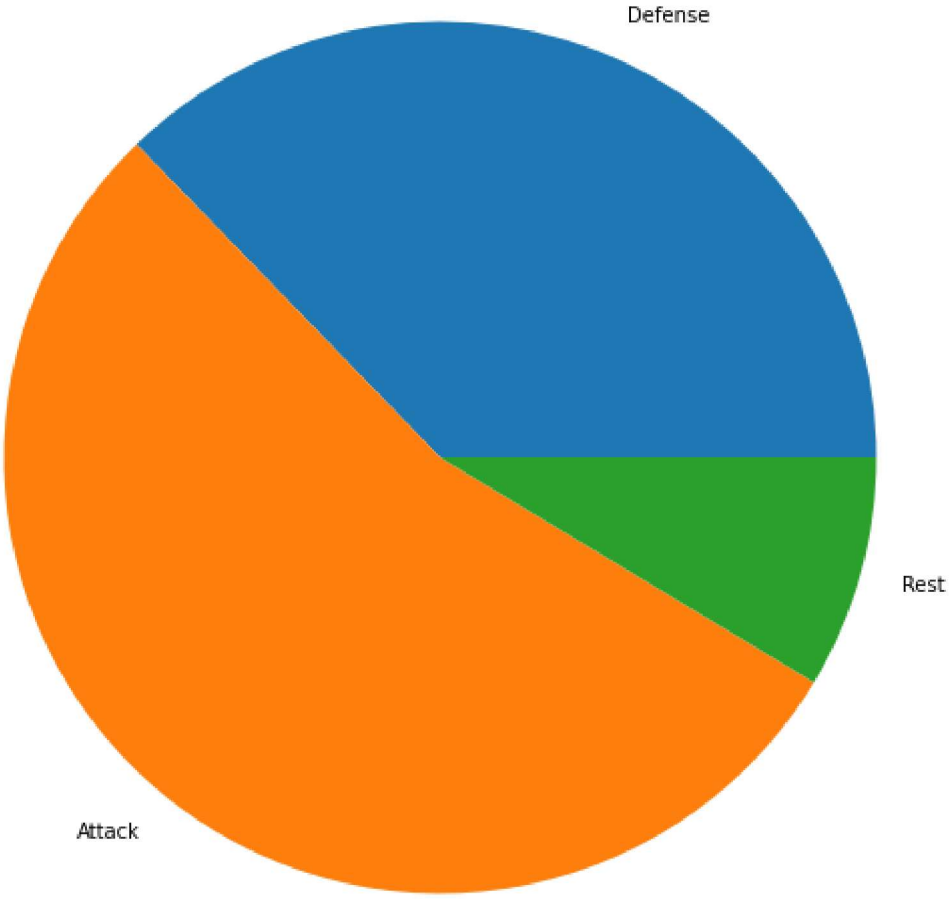
Aufgabe 7: Übersicht über defensive und angriffsfreudige Pokemon

Nutzen eines Histogramms, um die Verteilung der Kategorien abzuschätzen. Relativ gleich verteilt.

In [36]:

```
_, axes = plt.subplots()
axes.set_title("Übersicht über defensive und angriffsfreudige Pokemon")
axes.set_xlabel("Kategorisierung. Negativ = Defensiv, Positiv = Angriff")
axes.set_ylabel("Anzahl der Pokemon")
_ = (pokemon_data["Attack"] - pokemon_data["Defense"]).plot(kind="hist", ax=axes, fi

# Pie Chart
defense = pokemon_data[pokemon_data["Defense"] > pokemon_data["Attack"]]
attack = pokemon_data[pokemon_data["Defense"] < pokemon_data["Attack"]]

plt.figure(figsize=(20,10))
defense_count = defense.shape[0]
attack_count = attack.shape[0]
rest = pokemon_data.shape[0] - defense_count - attack_count
values = [defense_count, attack_count, rest]
labels = ["Defense", "Attack", "Rest"]
plt.pie(values, labels=labels)
plt.show()
```

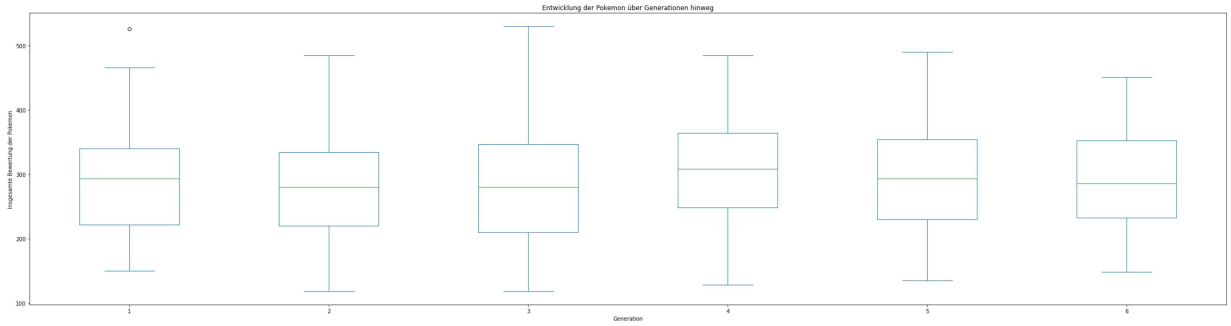




Aufgabe 8: Entwicklung der Pokemon über Generationen hinweg. Besser / Schlechter? Trends?

In [29]:

```
_, axes = plt.subplots()
axes.set_title("Entwicklung der Pokemon über Generationen hinweg")
axes.set_ylabel("Insgesamte Bewertung der Pokemon")
axes.set_xlabel("Generation")

# Generate new dataframe for each generation
generationFrames = [pd.DataFrame(gen["Overall"]).reset_index(drop=True) for _, gen i

# concat Frames and rename axes
_ = pd.concat(generationFrames, axis=1, ignore_index=True).set_axis(range(1, 7), axi
```



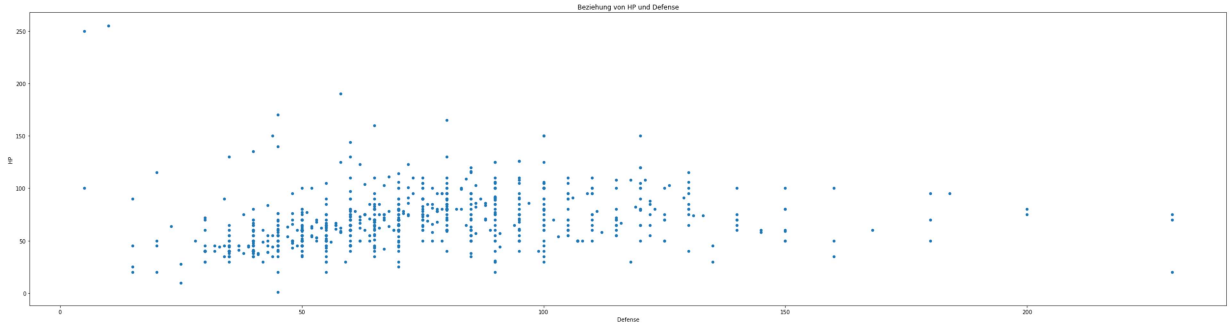
Aufgabe 9: Mindestens 3 weitere sinnvolle Plots

Beziehung von HP und Defense

Der Plot zeigt, dass Pokemon mit vielen HP eine eher geringere Defense haben. Je mehr Defense ein Pokemon hat, desto geringer sind seine HP. Es sind zwar viele Außreißer zu erkennen, alles in allem ist aber ein klares Schema erkennbar.

In [152…

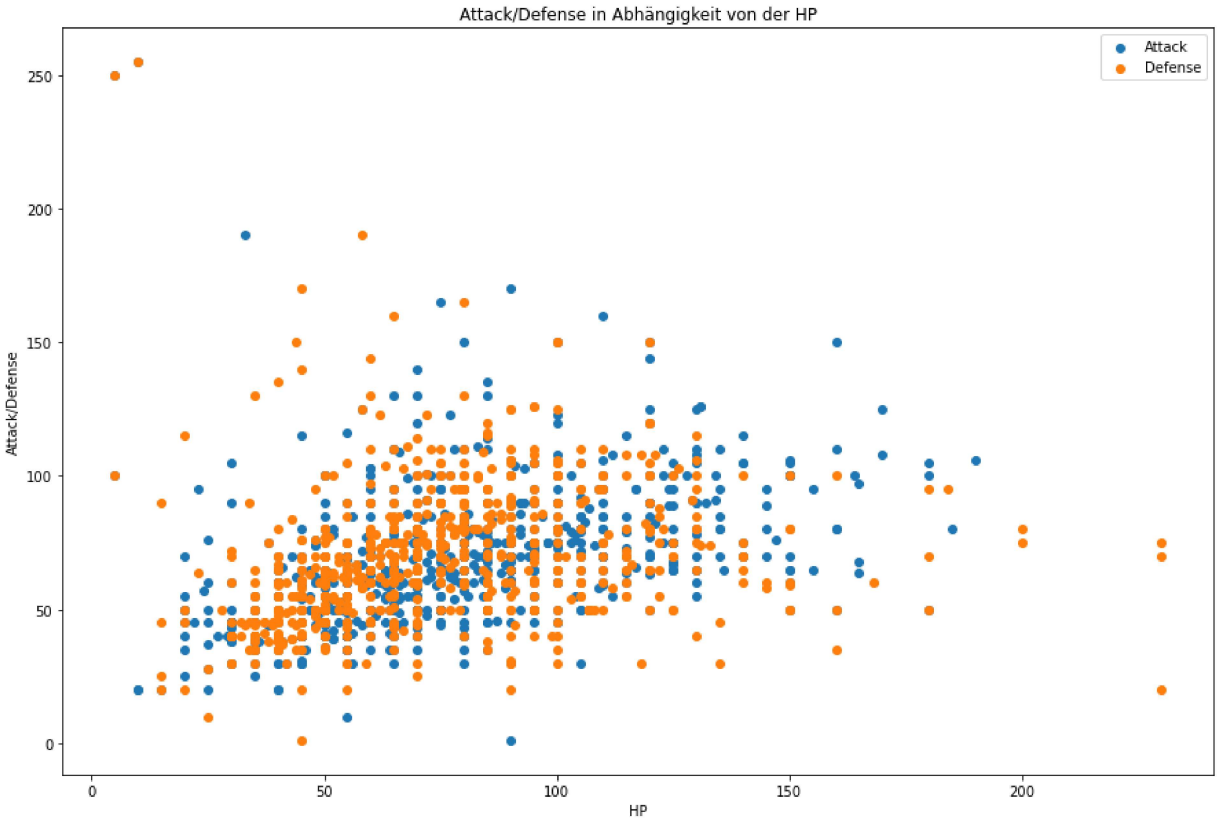
```
_, axes = plt.subplots()
axes.set_title("Beziehung von HP und Defense")
axes.set_xlabel("Defense")
axes.set_ylabel("HP")
_ = pokemon_data[["Defense", "HP"]].sort_values("Defense").plot(kind="scatter", x="D
```



Beziehung von Attack/Defense und HP

In [32]:

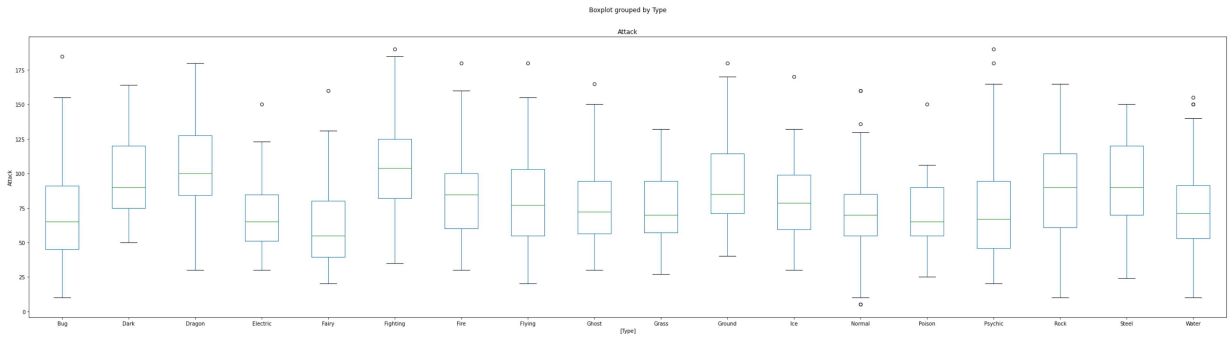
```
plt.figure(figsize=(15,10))
plt.scatter(pokemon_data["Attack"],pokemon_data["HP"],label="Attack")
plt.scatter(pokemon_data["Defense"],pokemon_data["HP"], label="Defense")
plt.legend()
plt.xlabel("HP")
plt.ylabel("Attack/Defense")
plt.title("Attack/Defense in Abhängigkeit von der HP")
plt.show()
```



Durchschnittlicher Angriff für jeden Pokemon Type

Dem Boxplot kann entnommen werden, dass Pokemon der Kategorie `Fighting` durchschnittlich am meisten Schaden verursachen, während `Fairy` am wenigsten Angriffsstärke haben

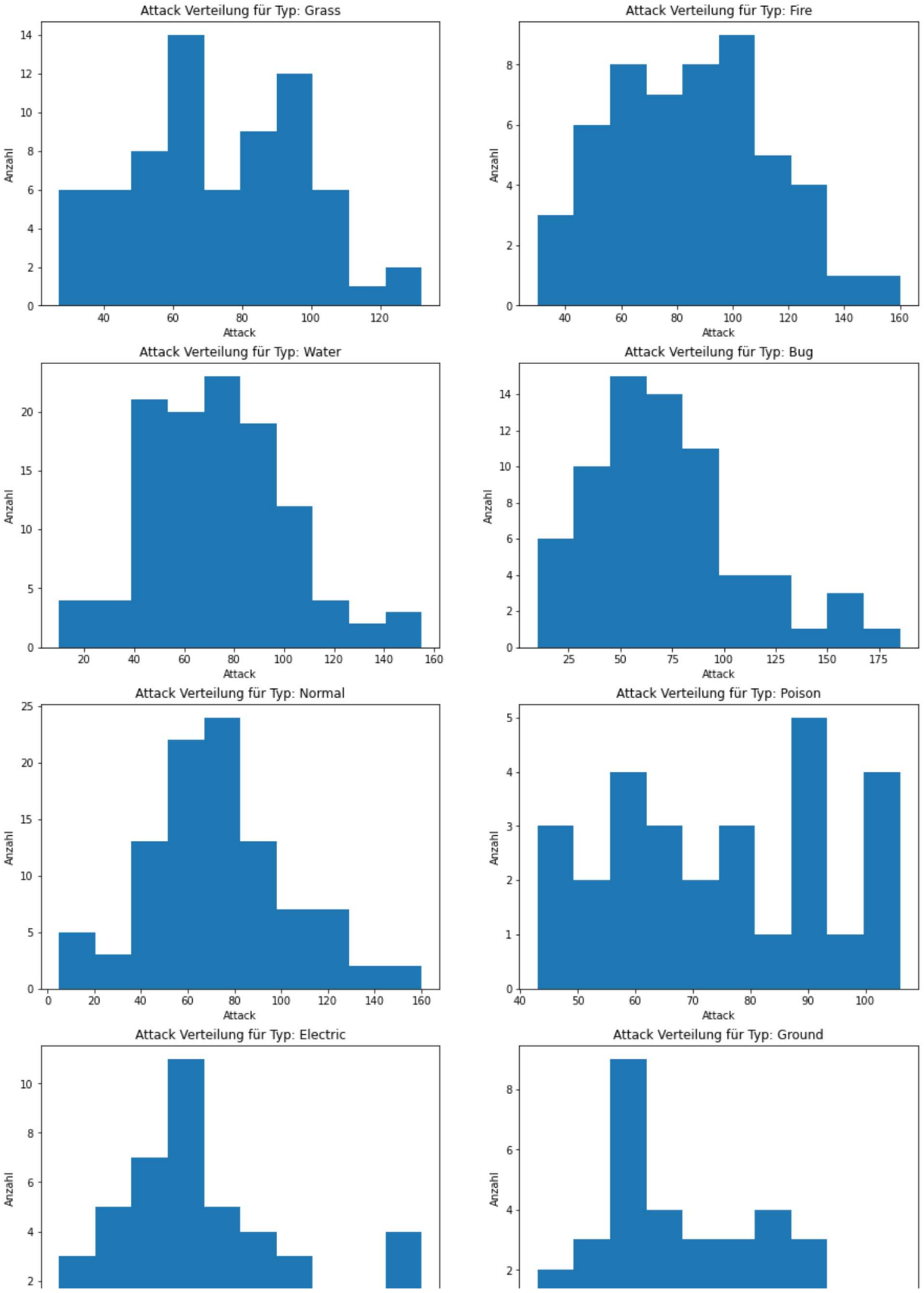
In [31]:
```
_, axes = plt.subplots(figsize=(40, 10))
axes.set_title("Boxplot über den Angriff aller Pokemon kategorisiert durch den Typen
axes.set_ylabel("Attack")
axes.set_xlabel("Pokemon Typ")

# Generate new dataframe for each Type
type1Frames = [pd.DataFrame(pokemon_type[["Type 1", "Attack"]]).reset_index(drop=Tru
type2Frames = [pd.DataFrame(pokemon_type[["Type 2", "Attack"]]).reset_index(drop=Tru

# Combine Type 1 and Type 2
typeFrames = pd.concat(type1Frames).set_axis(["Type", "Attack"], axis=1).append(type
_ = typeFrames.boxplot(by="Type", ax=axes, grid=False)
```

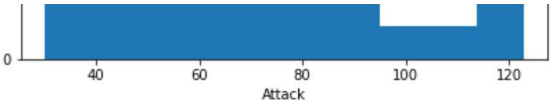


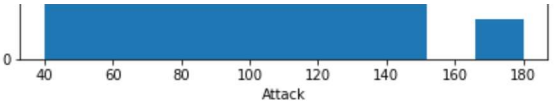
Attack-Verteilung für Type

In [37]:
```
from math import ceil
```

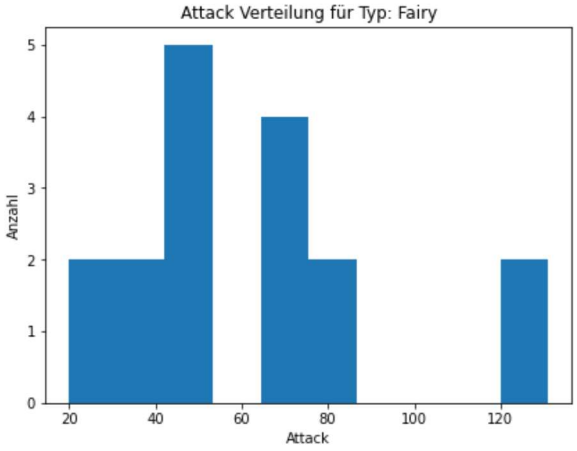
```
type_count = pokemon_data["Type 1"].nunique()

plt.figure(figsize=(15,40))
plt.subplots_adjust(left=None, bottom=2, right=None, top=3, wspace=None, hspace=None
for index,pokemon_type in enumerate(pokemon_data["Type 1"].unique()):
    plt.subplot(ceil(type_count/2),2,index + 1)
    plt.title(f"Attack Verteilung für Typ: {pokemon_type}")
    plt.ylabel("Anzahl")
    plt.xlabel("Attack")
    plt.hist(pokemon_data[pokemon_data["Type 1"] == pokemon_type].loc[:,"Attack"], b
plt.show()
```



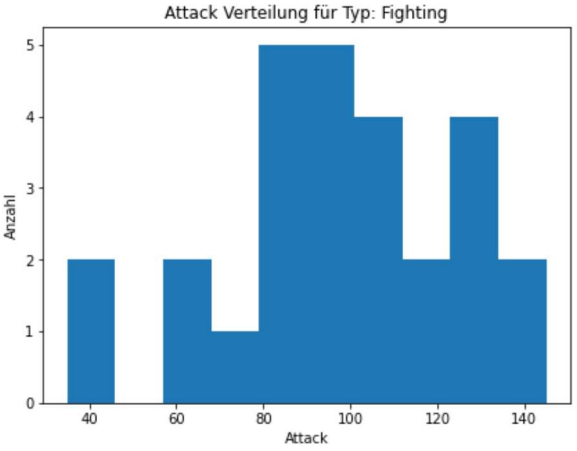




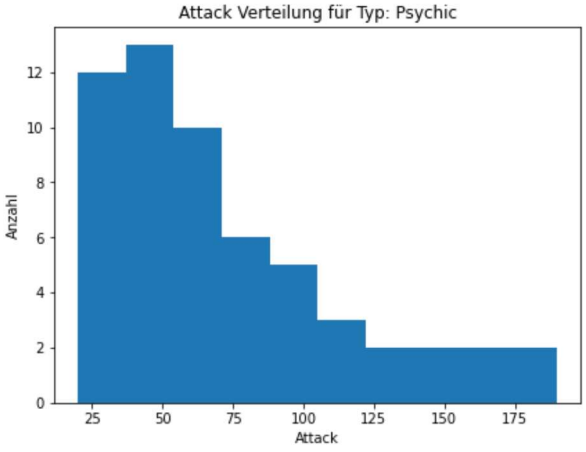
Attack Verteilung für Typ: Fairy



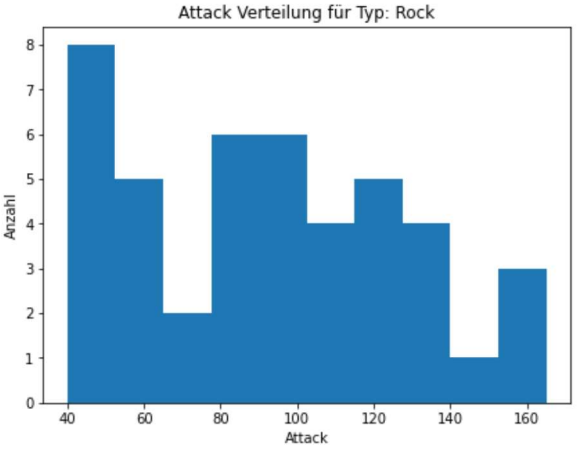
Attack Verteilung für Typ: Fighting



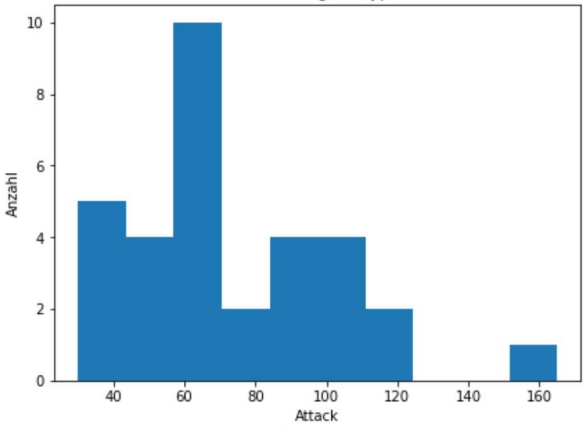
Attack Verteilung für Typ: Psychic



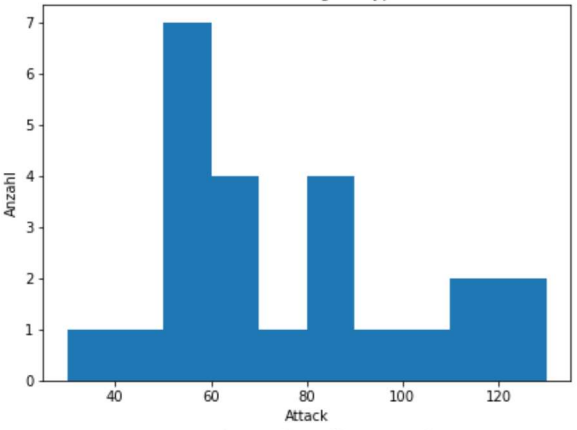
Attack Verteilung für Typ: Rock



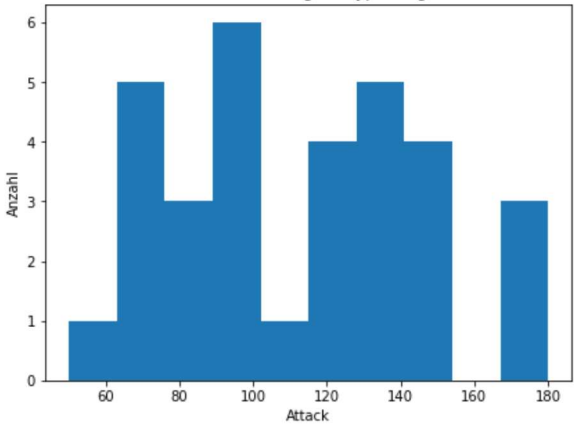
Attack Verteilung für Typ: Ghost



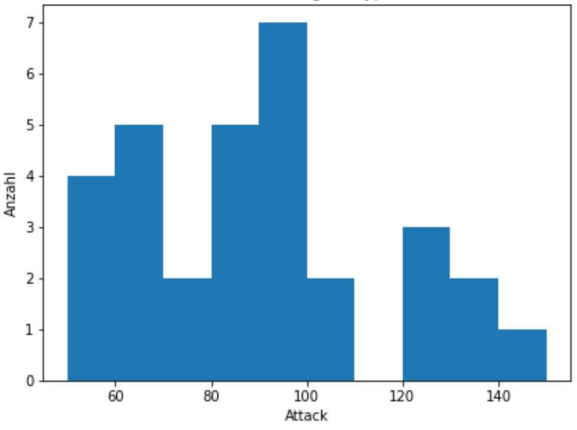
Attack Verteilung für Typ: Ice



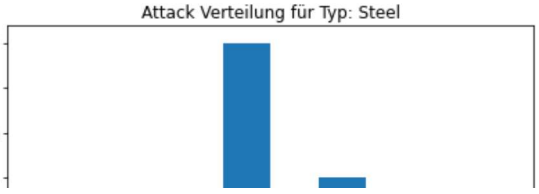
Attack Verteilung für Typ: Dragon



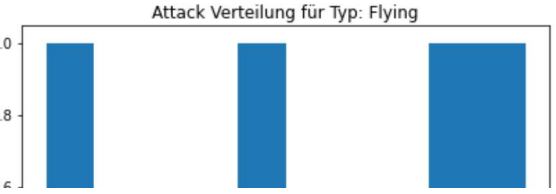
Attack Verteilung für Typ: Dark

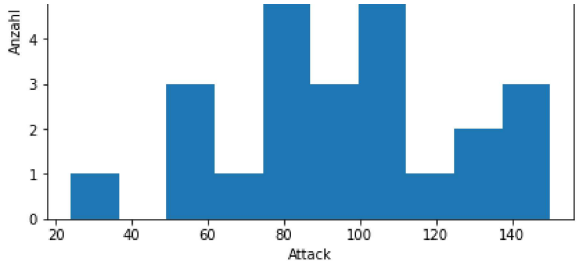


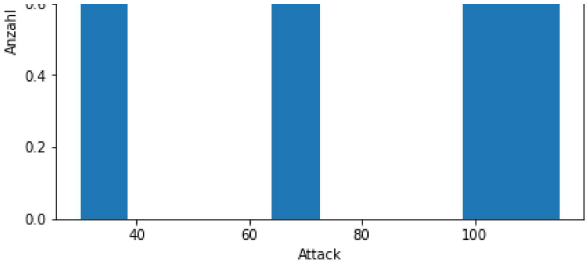
Attack Verteilung für Typ: Steel



Attack Verteilung für Typ: Flying



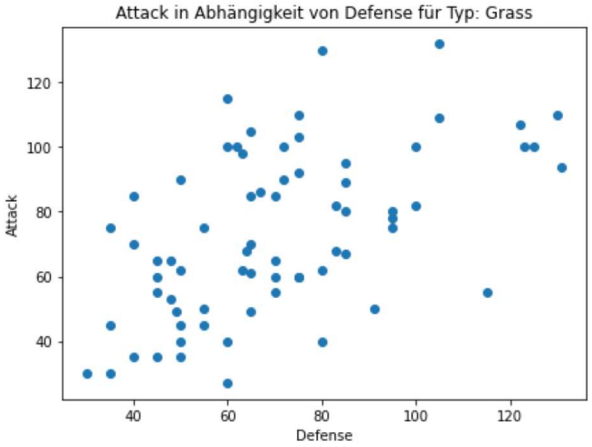


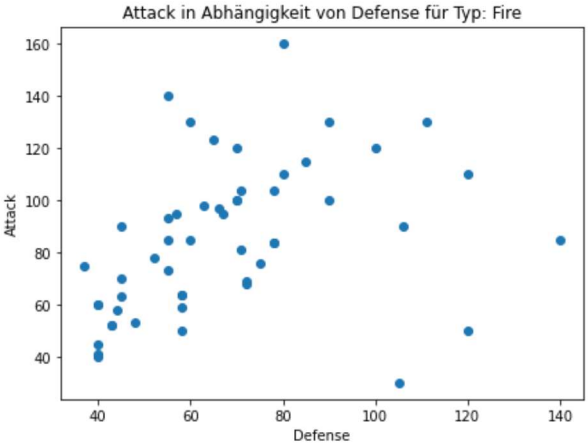


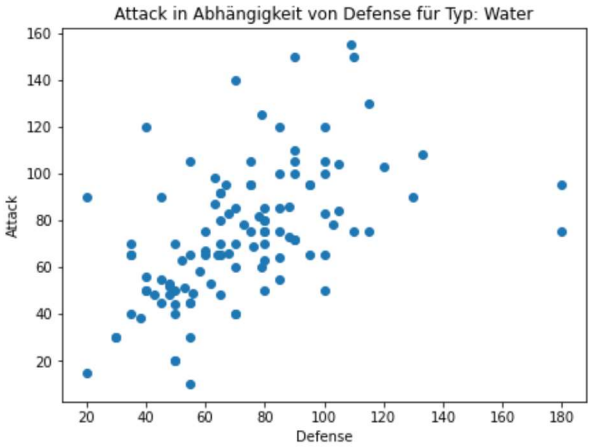
Attack in Beziehung zu Defense für jeden Type

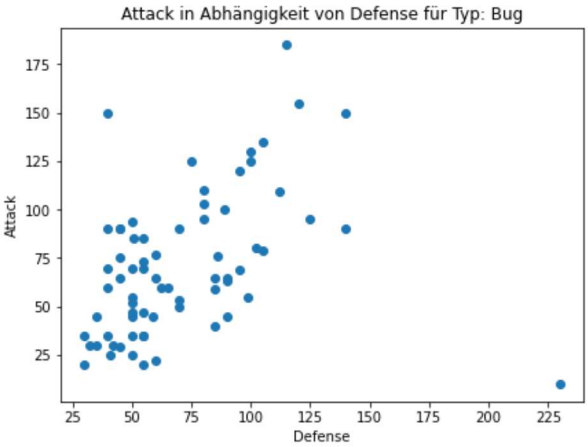
In [160…

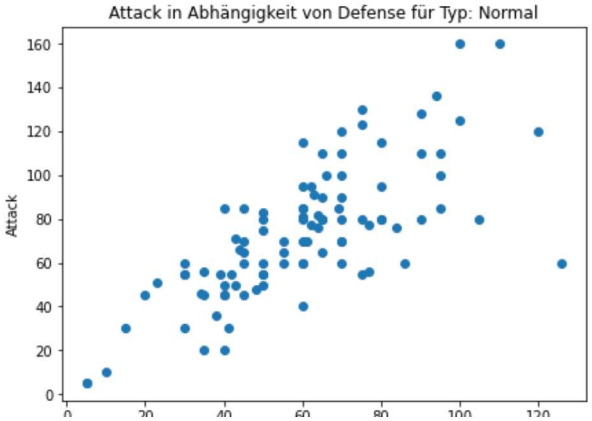
```
plt.figure(figsize=(15,40))
plt.subplots_adjust(left=None, bottom=2, right=None, top=3, wspace=None, hspace=None
for index,pokemon_type in enumerate(pokemon_data["Type 1"].unique()):
    plt.subplot(ceil(type_count/2),2,index + 1)
    plt.title(f"Attack in Abhängigkeit von Defense für Typ: {pokemon_type}")
    plt.ylabel("Attack")
    plt.xlabel("Defense")
    type1 = pokemon_data[pokemon_data["Type 1"] == pokemon_type]
    plt.scatter(x=type1["Defense"], y=type1["Attack"])
plt.show()
```

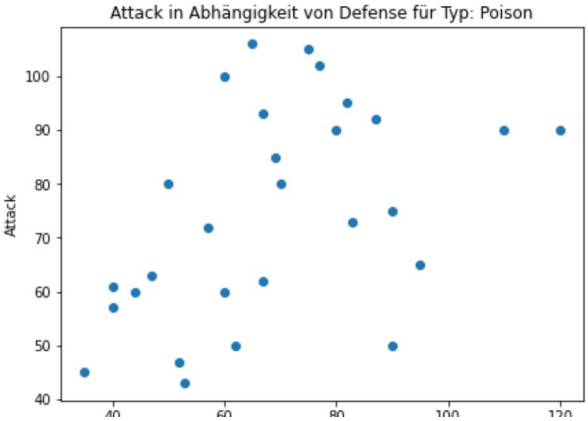


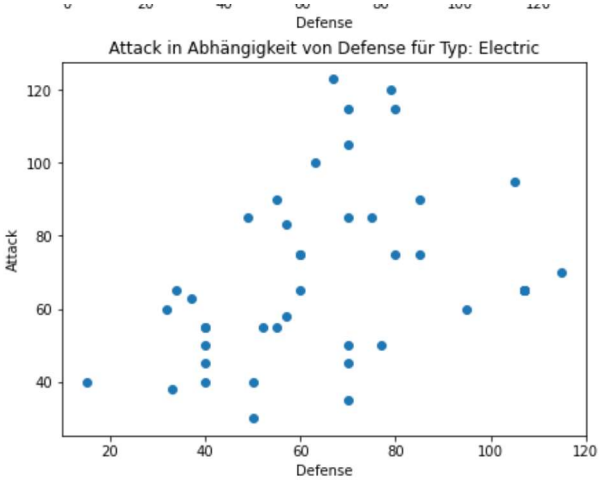


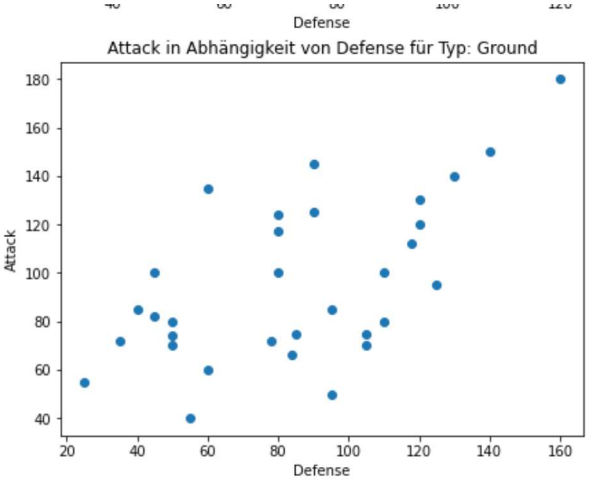


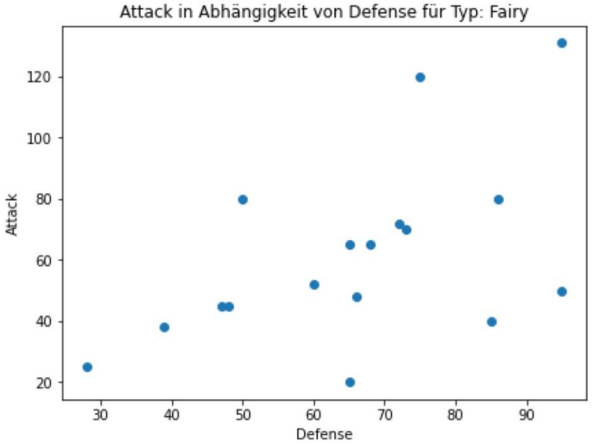


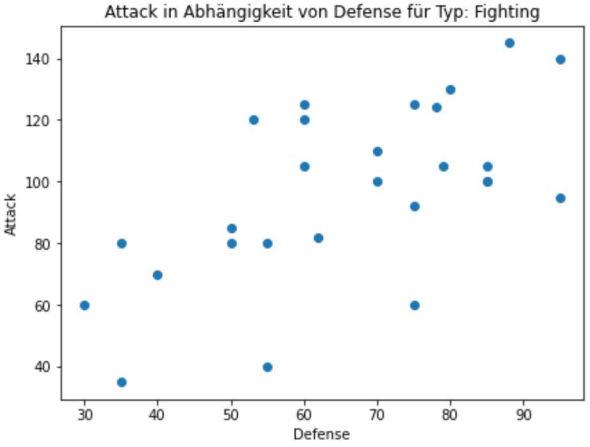


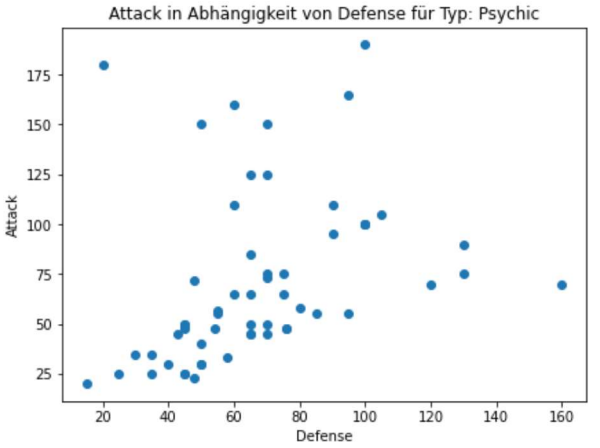


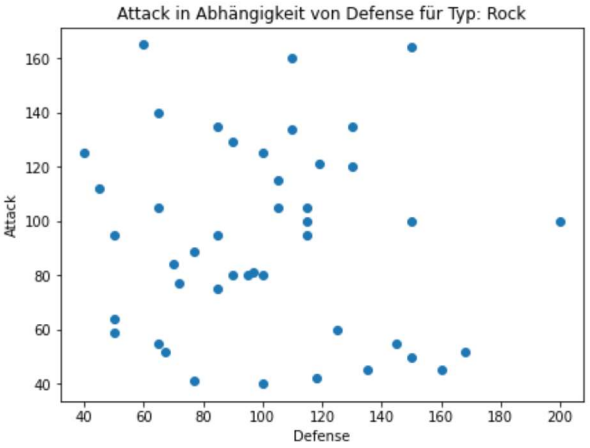

Attack in Abhängigkeit von Defense für Typ: Electric
Attack
Defense

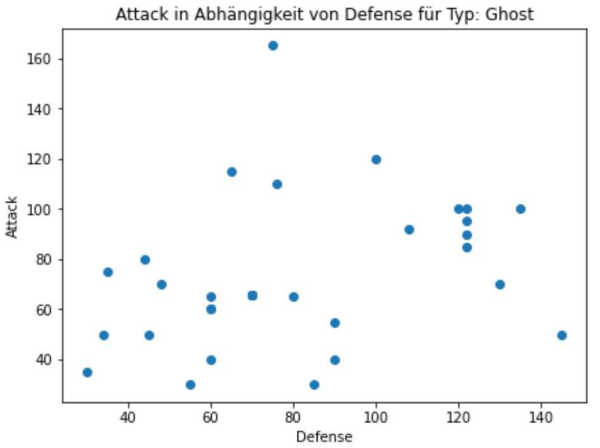

Attack in Abhängigkeit von Defense für Typ: Ground
Attack
Defense

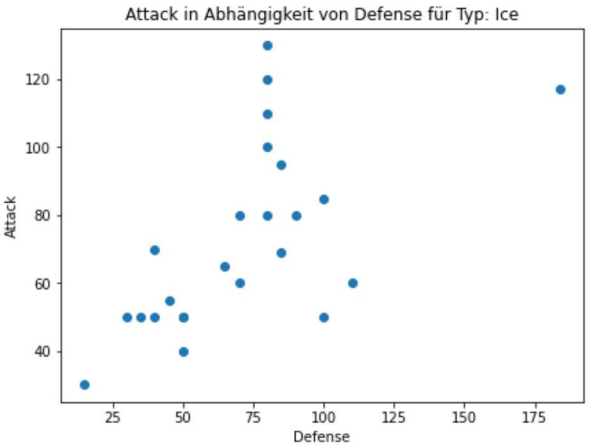

Attack in Abhängigkeit von Defense für Typ: Fairy
Attack
Defense

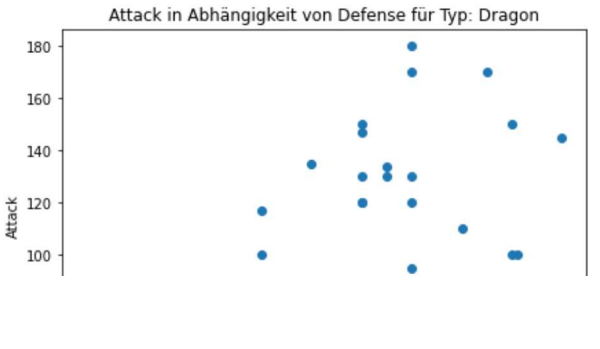

Attack in Abhängigkeit von Defense für Typ: Fighting
Attack
Defense

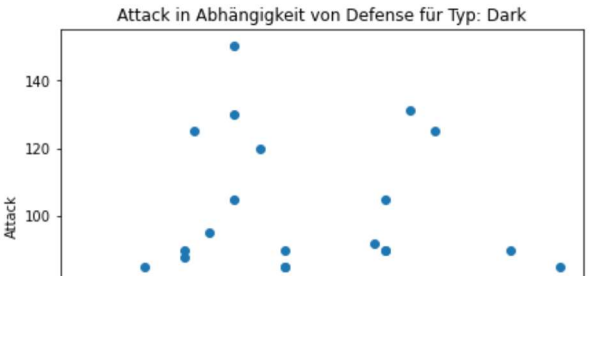

Attack in Abhängigkeit von Defense für Typ: Psychic
Attack
Defense

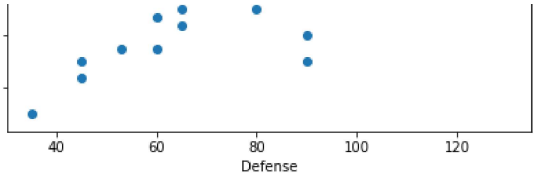

Attack in Abhängigkeit von Defense für Typ: Rock
Attack
Defense

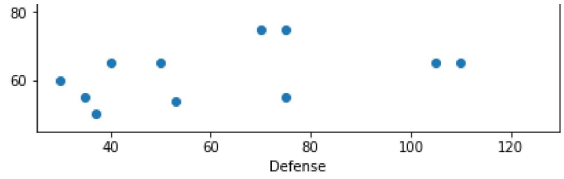

Attack in Abhängigkeit von Defense für Typ: Ghost
Attack
Defense

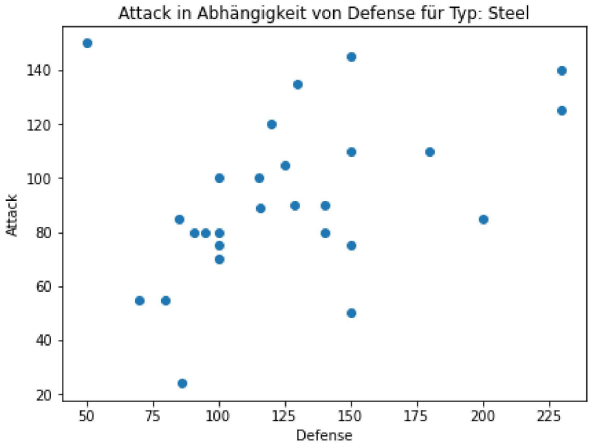

Attack in Abhängigkeit von Defense für Typ: Ice
Attack
Defense

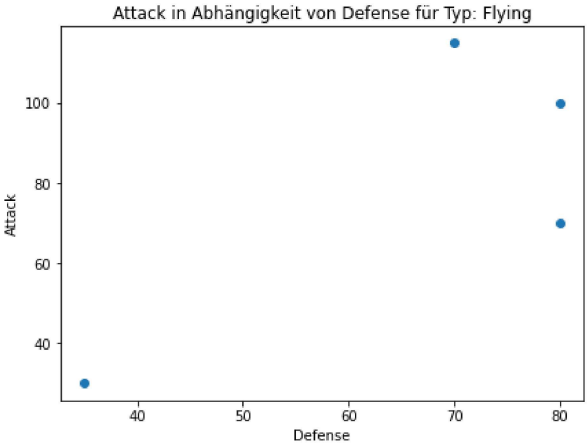

Attack in Abhängigkeit von Defense für Typ: Dragon
Attack


Attack in Abhängigkeit von Defense für Typ: Dark
Attack


80
60
40
60
80
100
120
Defense


80
60
40
60
80
100
120
Defense


Attack in Abhängigkeit von Defense für Typ: Steel
Attack
140
120
100
80
60
40
20
50
75
100
125
150
175
200
225
Defense


Attack in Abhängigkeit von Defense für Typ: Flying
Attack
100
80
60
40
40
50
60
70
80
Defense