In [ ]:

```
import seaborn as sns
```

Portfolio assignment 3

15 min: Perform a univariate analysis on all the categorical data of the penguins dataset. Commit the notebook to your portfolio when you're finished. Optional: Start working on portfolio assignment 4

In [ ]:

```
penguins = sns.load_dataset("penguins")
```

In [ ]:

```
penguins.head()
```

Out[ ]:

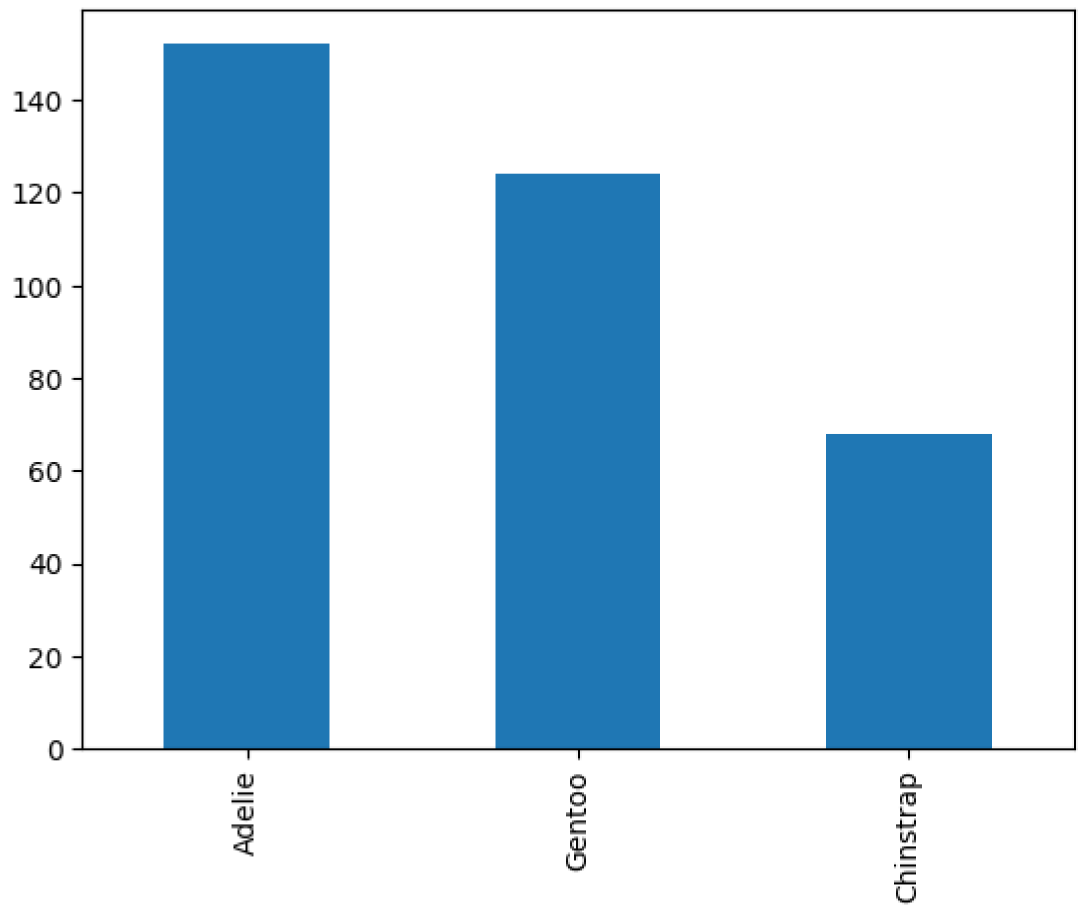
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

In [ ]:

```
penguins['species'].value_counts(dropna=False).plot.bar()
```

Out[ ]:

```
<AxesSubplot: >
```

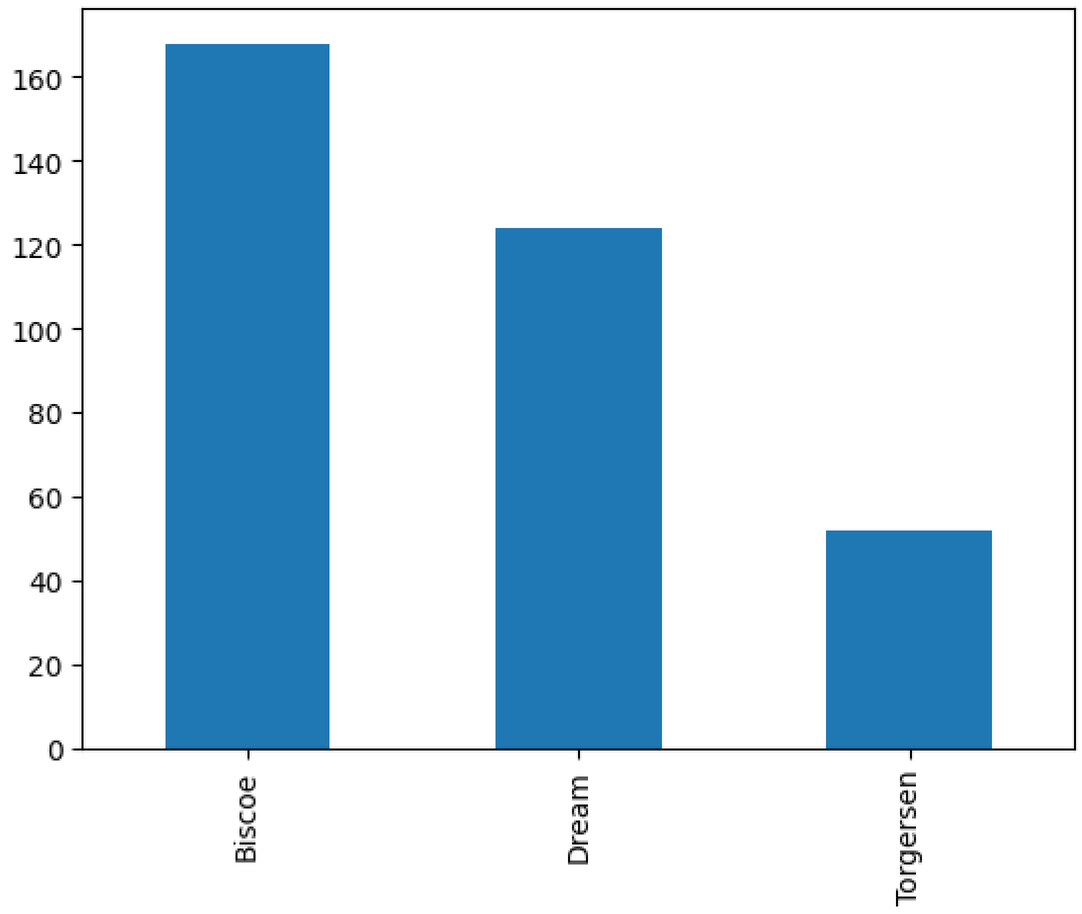


In [ ]:

```
penguins['island'].value_counts(dropna=False).plot.bar()
```

Out[ ]:

```
<AxesSubplot: >
```

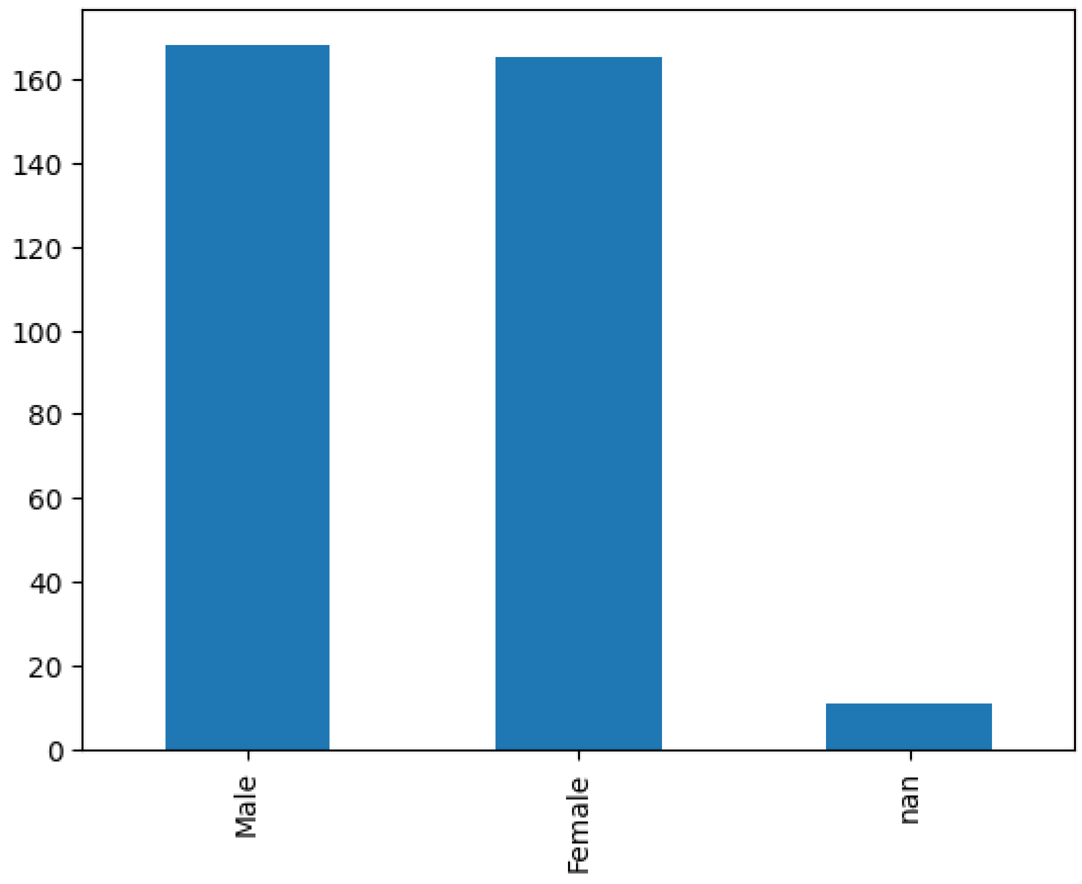


In [ ]:

```
penguins['sex'].value_counts(dropna=False).plot.bar()
```

Out[ ]:

```
<AxesSubplot: >
```



Portfolio assignment 4

15 min: Look online for a datset that you personally find interesting to explore. It can be about any topic that you find interesting: sports, games, software development, etc. Commit the dataset to your portfolio. You will be analysing the dataset in future portfolio assignments.

Required characteristics of the dataset:

- Must be in a tabular format: Contains rows and columns
- Contains at least 100 rows
- Contains at least 2 columns with categorical data and at least 2 columns with numerical data
- Is less than 200 MB

Dataset:

https://github.com/nytimes/covid-19-data/tree/master/colleges

In [ ]:

```
import pandas as pd
```

In [ ]:

```
pokemon = pd.read_csv("../pokemon.csv", sep=",")
pokemon
```

Out[ ]:

	abilities	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_fight	ag
0	['Overgrow', 'Chlorophyll']	1.00	1.0	1.0	0.5	0.5	0.5	2.
1	['Overgrow', 'Chlorophyll']	1.00	1.0	1.0	0.5	0.5	0.5	2.
2	['Overgrow', 'Chlorophyll']	1.00	1.0	1.0	0.5	0.5	0.5	2.
3	['Blaze', 'Solar Power']	0.50	1.0	1.0	1.0	0.5	1.0	0.
4	['Blaze', 'Solar Power']	0.50	1.0	1.0	1.0	0.5	1.0	0.
...	...	...	...	...	...	...	...	...
796	['Beast Boost']	0.25	1.0	0.5	2.0	0.5	1.0	2.
797	['Beast Boost']	1.00	1.0	0.5	0.5	0.5	2.0	4.
798	['Beast Boost']	2.00	0.5	2.0	0.5	4.0	2.0	0.
799	['Prism Armor']	2.00	2.0	1.0	1.0	1.0	0.5	1.
800	['Soul-Heart']	0.25	0.5	0.0	1.0	0.5	1.0	2.

801 rows × 41 columns

In [ ]:

```
import pandas as pd
```

Portfolio assignment 5

20 min:

- Download lifeExpectancyAtBirth.csv from Brightspace (original source).
- Move the file to the same folder as the Notebook that you will be working in.
- Load the dataset in your Notebook with the following code: lifeExpectancy = pd.read_csv('lifeExpectancyAtBirth.csv', sep=',')
- Look at the dataset with the .head() function.
- Filter the dataframe: We only want the life expectancy data about 2019 and 'Both sexes'
- Use this dataframe to perform a univariate analysis on the life expectancy in 2019.
- Which five countries have the highest life expectancy? Which five the lowest?

Commit the notebook and dataset to your portfolio when you're finished.

In [ ]:

```
# Load dataframe
lifeExpectancy = pd.read_csv('lifeExpectancyAtBirth.csv', sep=',')

# Quick look at dataframe
lifeExpectancy.head()
```

Out[ ]:

	Location	Period	Indicator	Dim1	First Tooltip
0	Afghanistan	2019	Life expectancy at birth (years)	Both sexes	63.21
1	Afghanistan	2019	Life expectancy at birth (years)	Male	63.29
2	Afghanistan	2019	Life expectancy at birth (years)	Female	63.16
3	Afghanistan	2015	Life expectancy at birth (years)	Both sexes	61.65
4	Afghanistan	2015	Life expectancy at birth (years)	Male	61.04

In [ ]:

```
# Filter dataframe
filtered = lifeExpectancy[(lifeExpectancy["Period"] == 2019) & (lifeExpectancy["Dim1"] == "Bo

# Quick look at filtered dataframe
filtered.head()
```

Out[ ]:

	Location	Period	Indicator	Dim1	First Tooltip
0	Afghanistan	2019	Life expectancy at birth (years)	Both sexes	63.21
12	Albania	2019	Life expectancy at birth (years)	Both sexes	78.00
24	Algeria	2019	Life expectancy at birth (years)	Both sexes	77.13
36	Angola	2019	Life expectancy at birth (years)	Both sexes	63.06
48	Antigua and Barbuda	2019	Life expectancy at birth (years)	Both sexes	76.45

In [ ]:

```
# Univariate analysis
filtered.describe()
```

Out[ ]:

	Period	First Tooltip
count	183.0	183.000000
mean	2019.0	72.540492
std	0.0	7.129956
min	2019.0	50.750000
25%	2019.0	66.550000
50%	2019.0	73.740000
75%	2019.0	77.730000
max	2019.0	84.260000

In [ ]:

```
# Highest life expectancy
filtered.sort_values(by=['First Tooltip']).tail(5)
```

Out[ ]:

	Location	Period	Indicator	Dim1	First Tooltip
1837	Spain	2019	Life expectancy at birth (years)	Both sexes	83.22
1753	Singapore	2019	Life expectancy at birth (years)	Both sexes	83.22
1573	Republic of Korea	2019	Life expectancy at birth (years)	Both sexes	83.30
1897	Switzerland	2019	Life expectancy at birth (years)	Both sexes	83.45
997	Japan	2019	Life expectancy at birth (years)	Both sexes	84.26

In [ ]:

```
# Lowest life expectancy
filtered.sort_values(by=['First Tooltip']).head(5)
```

Out[ ]:

	Location	Period	Indicator	Dim1	First Tooltip
1117	Lesotho	2019	Life expectancy at birth (years)	Both sexes	50.75
373	Central African Republic	2019	Life expectancy at birth (years)	Both sexes	53.10
1801	Somalia	2019	Life expectancy at birth (years)	Both sexes	56.47
661	Eswatini	2019	Life expectancy at birth (years)	Both sexes	57.73
1333	Mozambique	2019	Life expectancy at birth (years)	Both sexes	58.14

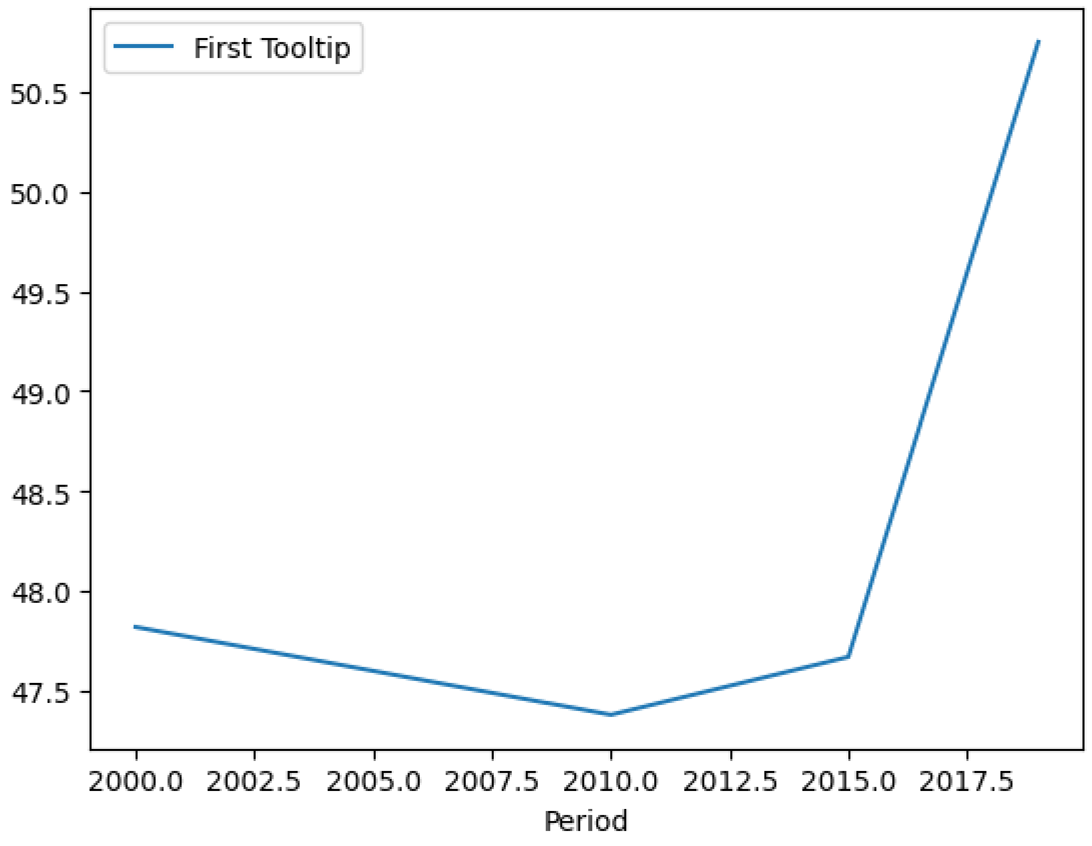
In [ ]:

```
# Analysis only on Lesotho because I went there ;)
lesothoOnly = lifeExpectancy[(lifeExpectancy["Location"] == "Lesotho") & (lifeExpectancy["Dim

lesothoOnly.plot(x='Period', y='First Tooltip', kind='line')
```

Out[ ]:

```
<AxesSubplot: xlabel='Period'>
```


First Tooltip
50.5
50.0
49.5
49.0
48.5
48.0
47.5
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
Period

Portfolio assignment 6

60 min: Perform a univariate analysis on at least 2 columns with categorical data and on at least 2 columns with numerical data in the dataset that you chose in portfolio assignment 4. Commit the Notebook to your portfolio when you're finished.

In [ ]:

```
import pandas as pd
import seaborn as sns
```

In [ ]:

```
pokemon = pd.read_csv("../pokemon.csv", sep=",")
pokemon.head()
```

Out[ ]:

	abilities	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_fight	agai
0	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
1	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
2	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
3	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5
4	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5

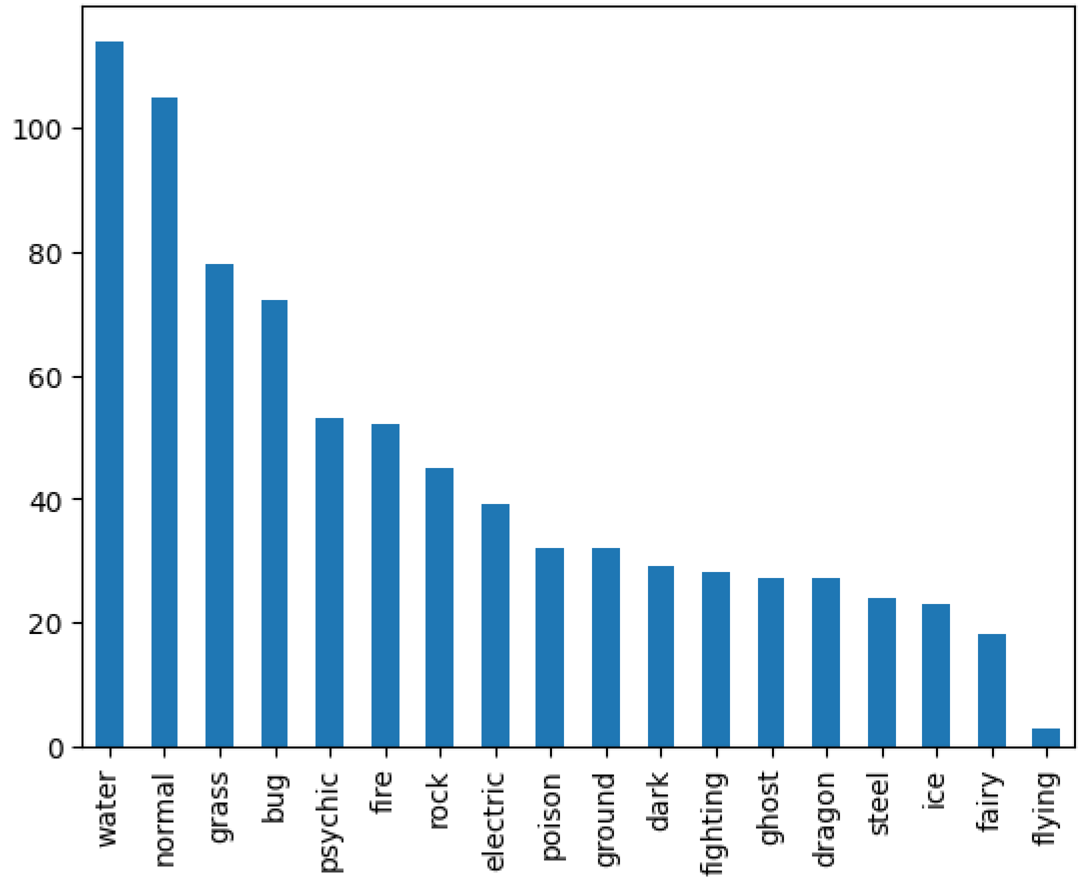
5 rows × 41 columns

In [ ]:

```
pokemon['type1'].value_counts(dropna=False).plot(kind='bar')
```

Out[ ]:

```
<AxesSubplot: >
```



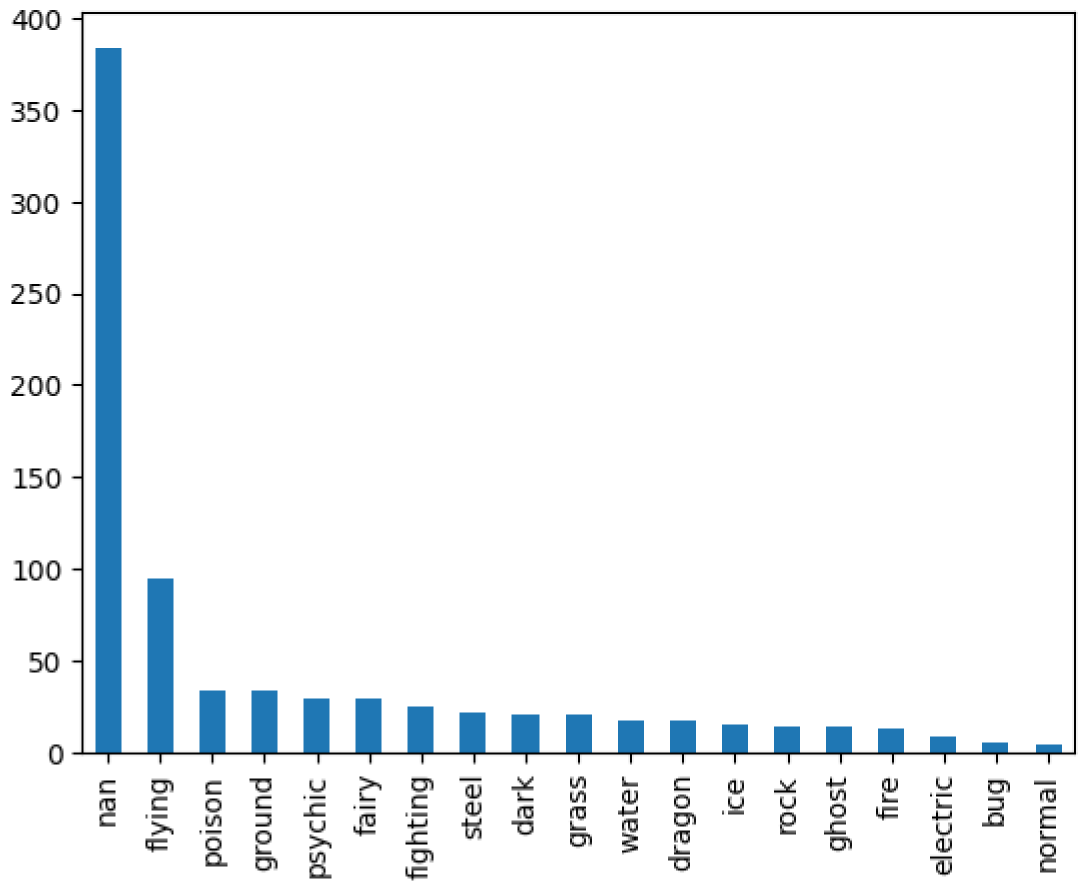
Het type water komt het meeste voor.

In [ ]:

```
pokemon['type2'].value_counts(dropna=False).plot(kind='bar')
```

Out[ ]:

```
<AxesSubplot: >
```



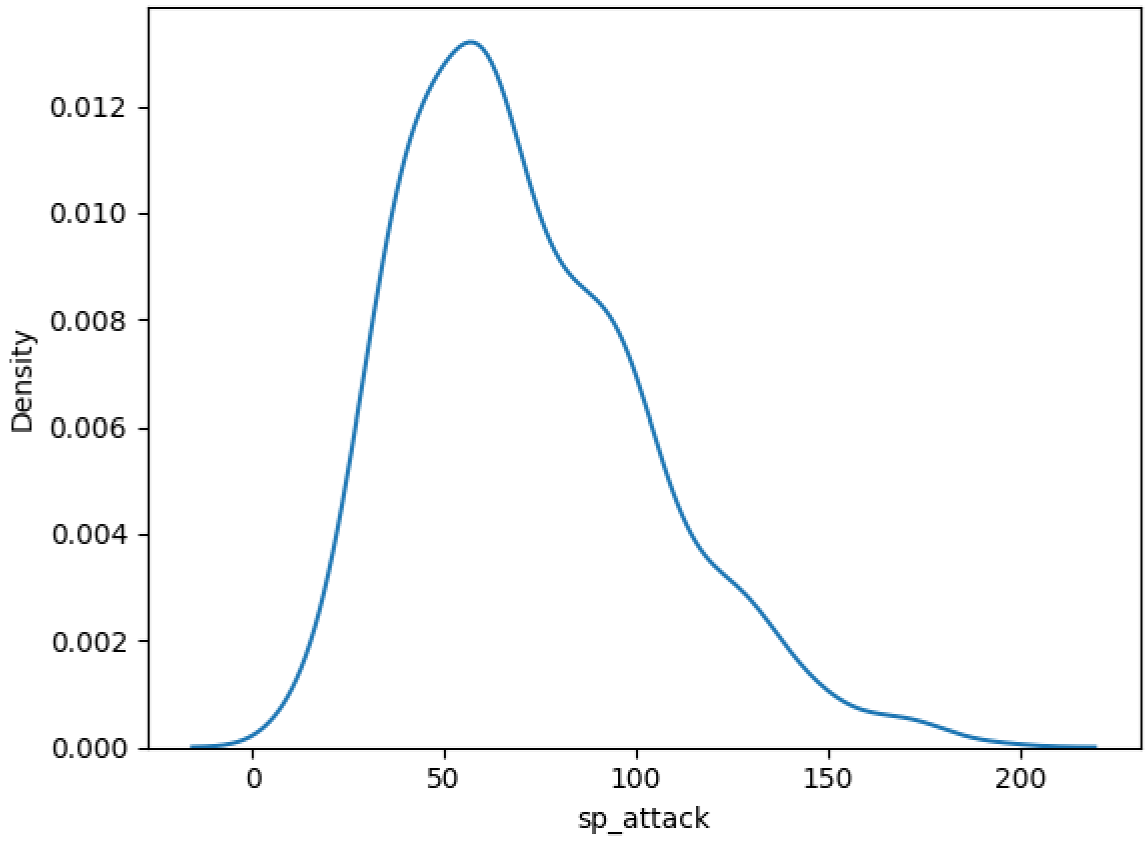
Het type flying komt naast de niet ingevulde waarde het meeste voor.

In [ ]:

```
sns.kdeplot(pokemon['sp_attack'])
```

Out[ ]:

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



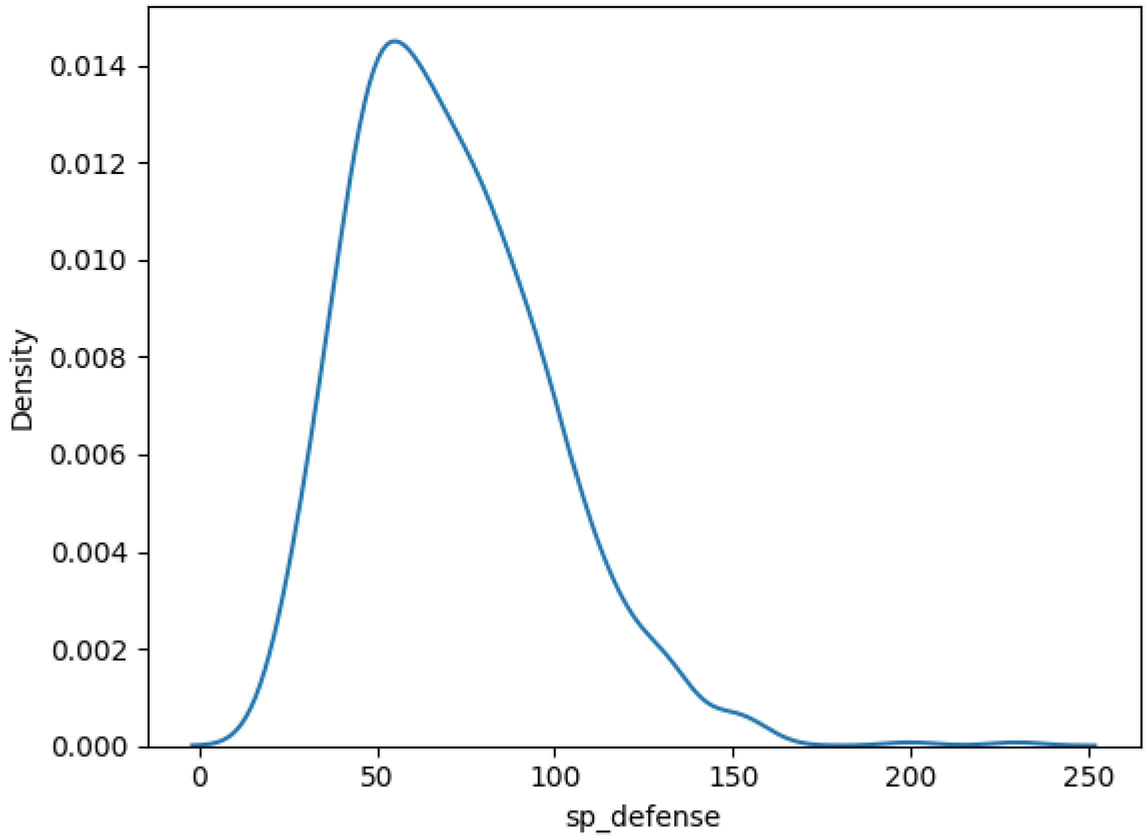
Rond de 60 a 65 sp_attack komt het meeste voor.

In [ ]:

```
sns.kdeplot(pokemon['sp_defense'])
```

Out[ ]:

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



Rond de 60 sp_defense komt het meeste voor.

Portfolio assignment 7

15 min: Look at the histogram of at least 2 columns with numerical data in the dataset that you chose in portfolio assignment 4. Do you recognise the distribution? Does it look like a uniform or normal distribution or something else? If it doesn't look like a uniform or normal distribution, take a quick look here to see if you can find the distribution shape: https://www.itl.nist.gov/div898/handbook/eda/section3/eda366.htm

In [ ]:

```
import pandas as pd
import seaborn as sns
```

In [ ]:

```
pokemon = pd.read_csv("../pokemon.csv", sep=",")
pokemon.head()
```

Out[ ]:

	abilities	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_fight	agai
0	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
1	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
2	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
3	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5
4	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5

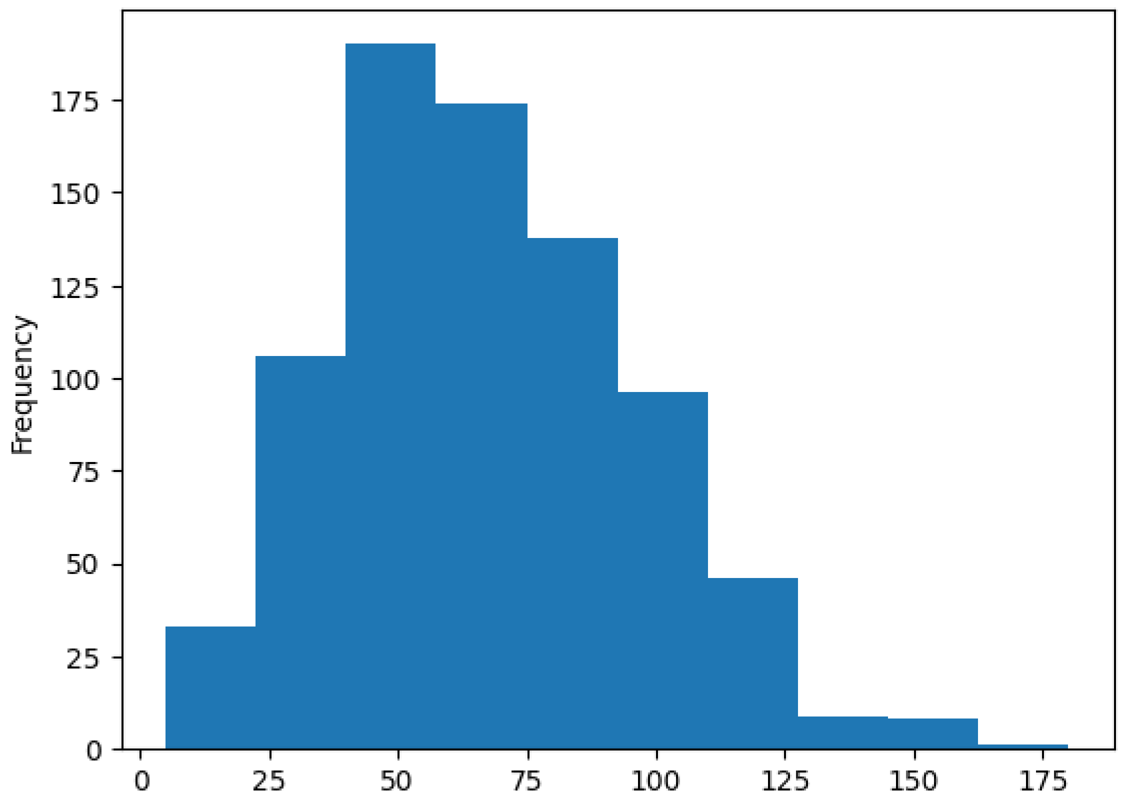
5 rows × 41 columns

In [ ]:

```
pokemon["speed"].plot.hist()
```

Out[ ]:

```
<AxesSubplot: ylabel='Frequency'>
```



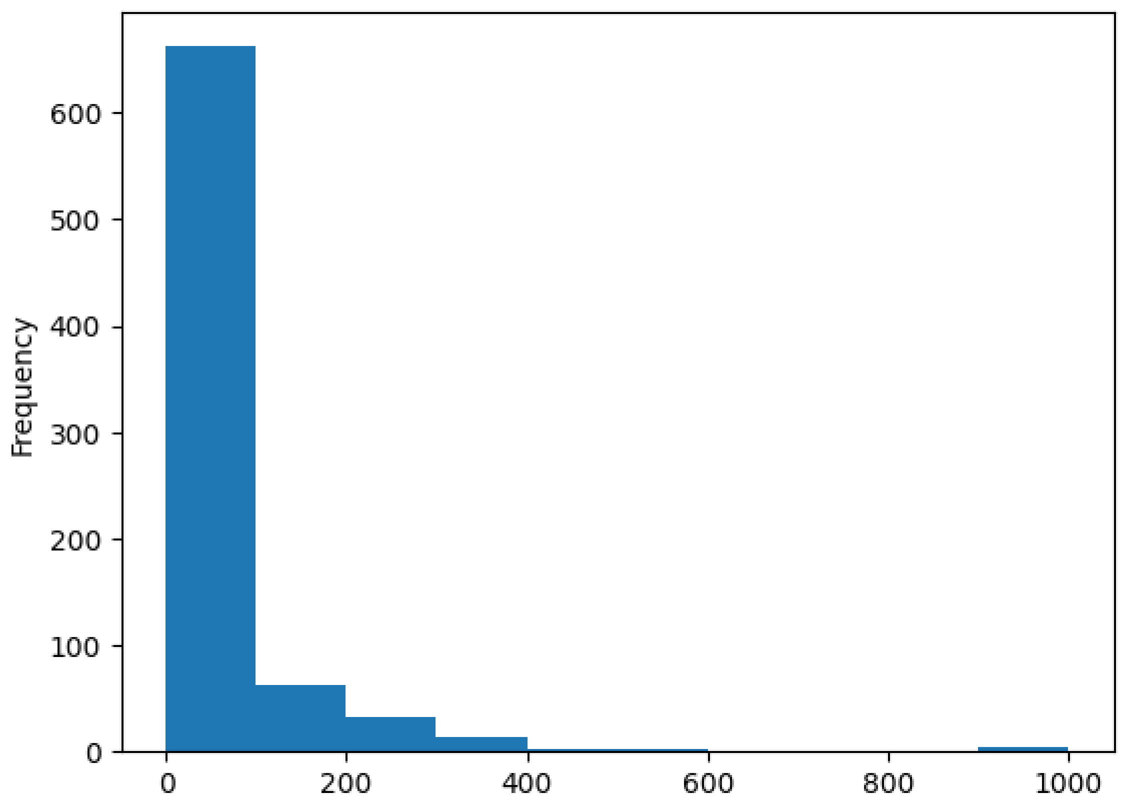
Weibull Distribution

In [ ]:

```
pokemon["weight_kg"].plot.hist()
```

Out[ ]:

```
<AxesSubplot: ylabel='Frequency'>
```



Lognormal Distribution

Portfolio assignment 8

15 min:

- Calculate the 90%, 95%, 99% and 99.99% confidence interval for at least 2 columns with numerical data in the dataset that you chose in portfolio assignment 4. Do you see the impact the confidence has on the interval?
- Now calculate the 95% confidence interval again but use only the first 10% of your rows. Compare this interval to the previous 95% confidence interval you calculated. Do you see the impact of having less data?

In [ ]:

```
import pandas as pd
import seaborn as sns
```

In [ ]:

```
pokemon = pd.read_csv("../pokemon.csv", sep=",")
pokemon.head()
```

Out[ ]:

	abilities	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_fight	agai
0	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
1	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
2	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
3	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5
4	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5

5 rows × 41 columns

In [ ]:

```
import scipy.stats as st

speed = pokemon["speed"]

confidence = 0.90
st.t.interval(confidence, len(speed)-1, loc=speed.mean(), scale=st.sem(speed))

# Confidence speed 90%
```

Out[ ]:

```
(64.65257722723477, 68.01658631833325)
```

In [ ]:

```
confidence = 0.95
st.t.interval(confidence, len(speed)-1, loc=speed.mean(), scale=st.sem(speed))

# Confidence speed 95%
```

Out[ ]:

```
(64.32963730629338, 68.33952623927466)
```

In [ ]:

```
confidence = 0.99
st.t.interval(confidence, len(speed)-1, loc=speed.mean(), scale=st.sem(speed))

# Confidence speed 99%
```

Out[ ]:

```
(63.697333205871644, 68.9718303396964)
```

In [ ]:

```
confidence = 0.9999
st.t.interval(confidence, len(speed)-1, loc=speed.mean(), scale=st.sem(speed))

# Confidence speed 99.99%
```

Out[ ]:

```
(62.3405940846074, 70.32856946096064)
```

Calculate confidence interval over top 10% of rows

In [ ]:

```
rows = len(pokemon)

pokemon10speed = pokemon.head(round(rows / 100 * 10))["speed"]

confidence = 0.95
print("95% ALL ROWS")
print(st.t.interval(confidence, len(speed)-1, loc=speed.mean(), scale=st.sem(speed)))

print("95% 10 percent ROWS")
print(st.t.interval(confidence, len(pokemon10speed)-1, loc=pokemon10speed.mean(), scale=st.se
```

```
95% ALL ROWS
(64.32963730629338, 68.33952623927466)
95% 10 percent ROWS
(61.42797858144729, 74.1470214185527)
```

In [ ]:

```
import pandas as pandas
import seaborn as sns
import numpy as np
```

Portfolio assignment 9

25 min: Perform a bivariate analysis on the columns with numerical data in the penguins dataset.

- Use corr() on the DataFrame to calculate all the correlations. Use the code example above to show the correlation table with colors.
- Look at the corrrelations. Do they match your expectations?
- Show a scatter plot for
  - The strongest positive correlation
  - The strongest negative correlation
  - The weakest correlation

In [ ]:

```
penguins = sns.load_dataset("penguins")
```

In [ ]:

```
penguins.head()
```

Out[ ]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

In [ ]:

```
penguinsCorrelation = penguins.corr()
penguinsCorrelation.style.background_gradient(cmap='coolwarm', axis=None).set_precision(2)
```

```
C:\Users\Jens\AppData\Local\Temp\ipykernel_33040\464541182.py:2: FutureWarning: this method is
  penguinsCorrelation.style.background_gradient(cmap='coolwarm', axis=None).set_precision(2)
```

Out[ ]:

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
bill_length_mm	1.00	-0.24	0.66	0.60
bill_depth_mm	-0.24	1.00	-0.58	-0.47
flipper_length_mm	0.66	-0.58	1.00	0.87
body_mass_g	0.60	-0.47	0.87	1.00

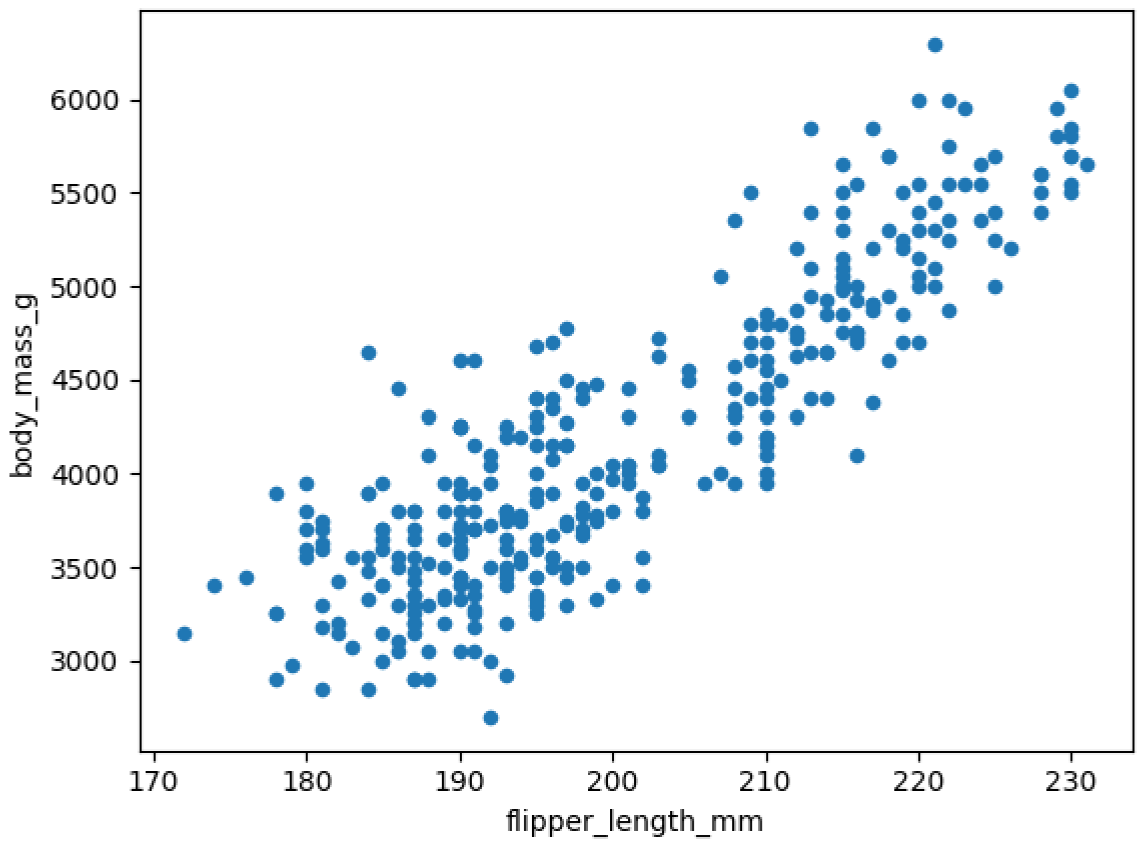
In [ ]:

```
penguins.plot(kind="scatter", x="flipper_length_mm", y="body_mass_g")
```

```
C:\Users\Jens\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalC
  scatter = ax.scatter(
```

Out[ ]:

```
<AxesSubplot: xlabel='flipper_length_mm', ylabel='body_mass_g'>
```

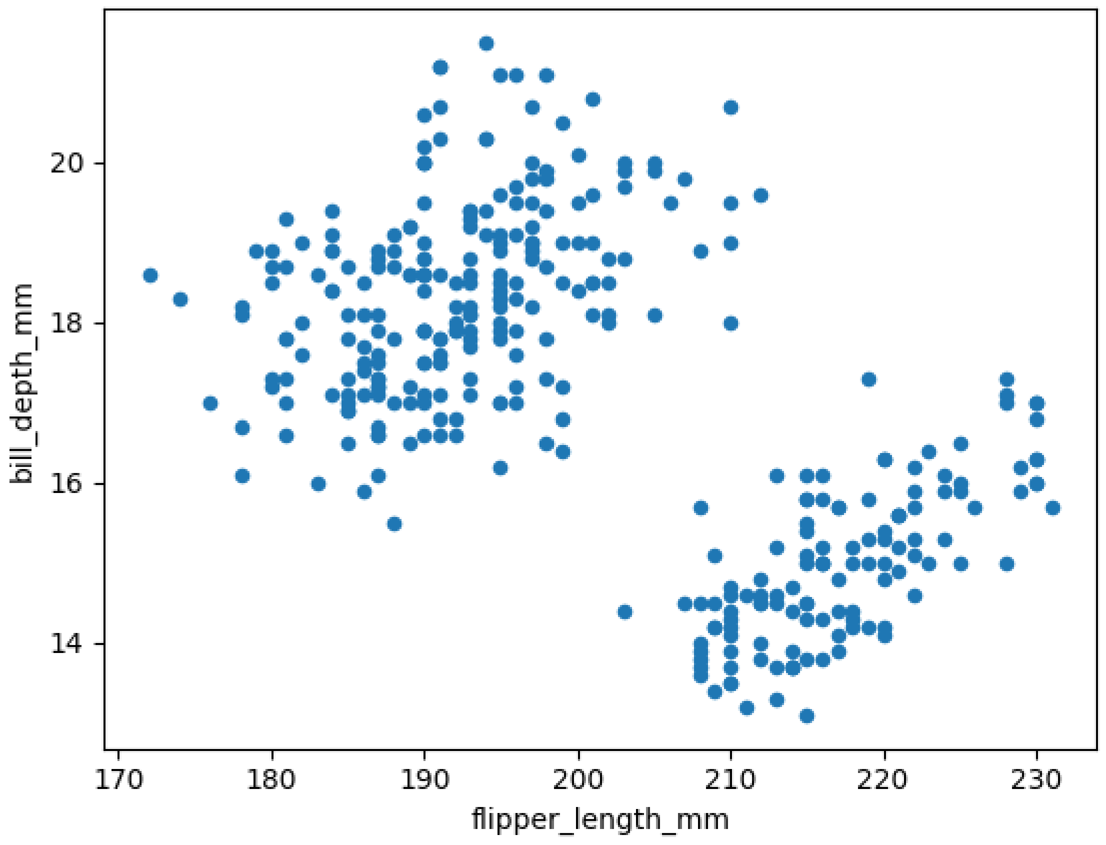


In [ ]:

```
penguins.plot(kind="scatter", x="flipper_length_mm", y="bill_depth_mm")
```

Out[ ]:

```
<AxesSubplot: xlabel='flipper_length_mm', ylabel='bill_depth_mm'>
```

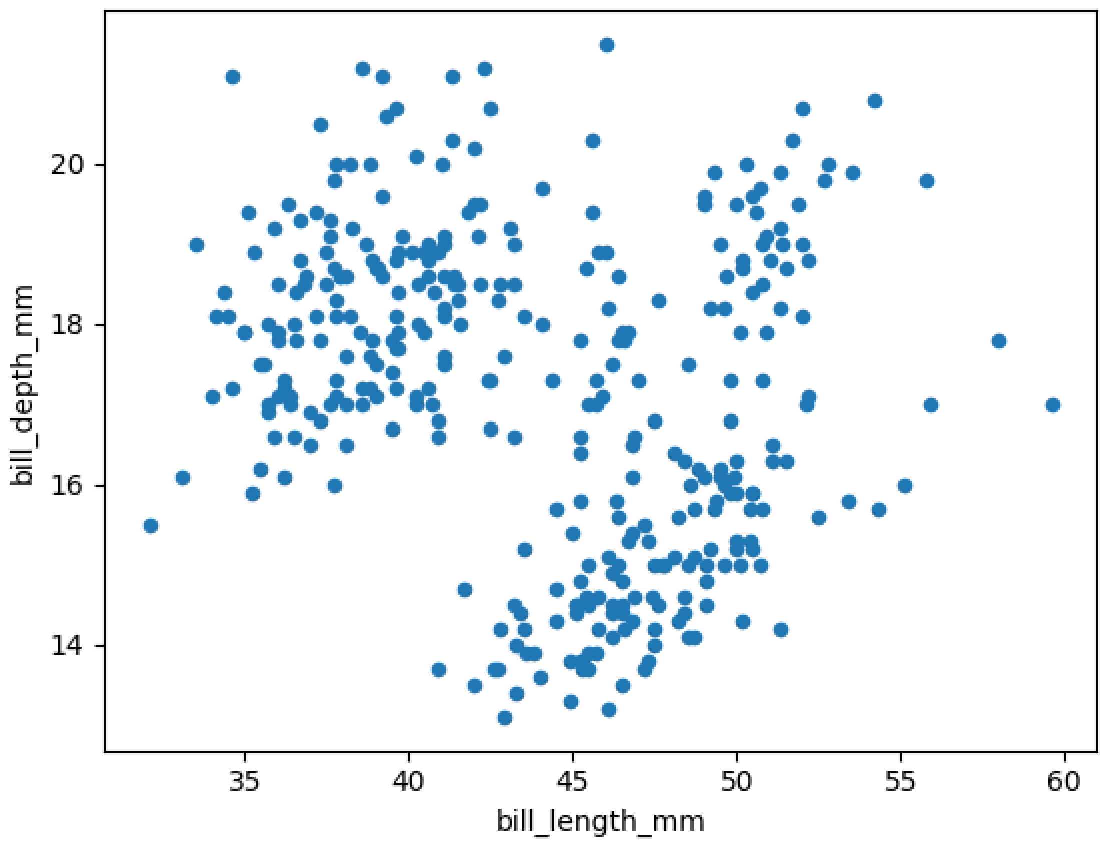


In [ ]:

```
penguins.plot(kind="scatter", x="bill_length_mm", y="bill_depth_mm")
```

Out[ ]:

```
<AxesSubplot: xlabel='bill_length_mm', ylabel='bill_depth_mm'>
```


20
18
16
14
bill_depth_mm
35
40
45
50
55
60
bill_length_mm

Portfolio assignment 10

15 min: Perform a bivariate analysis (Pearson correlation and scatter plot) on at least 1 combination of 2 columns with numeric data in the dataset that you chose in portfolio assignment 4. Does the correlation and scatter plot match your expectations? Add your answer to your notebook. Commit the Notebook to your portfolio when you're finished.

In [ ]:

```
import pandas as pd
```

In [ ]:

```
pokemon = pd.read_csv("../pokemon.csv", sep=",")
pokemon.head()
```

Out[ ]:

	abilities	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_fight	agai
0	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
1	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
2	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2.0
3	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5
4	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	0.5

5 rows × 41 columns

In [ ]:

```
pokemon.corr()
```

Out[ ]:

	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_fight
against_bug	1.000000	0.230107	0.165430	-0.246943	0.239566	0.137902
against_dark	0.230107	1.000000	0.140830	-0.015830	-0.301354	-0.357981
against_dragon	0.165430	0.140830	1.000000	-0.108928	0.439705	0.035237
against_electric	-0.246943	-0.015830	-0.108928	1.000000	-0.089864	-0.102798
against_fairy	0.239566	-0.301354	0.439705	-0.089864	1.000000	0.157712
against_fight	0.137902	-0.357981	0.035237	-0.102798	0.157712	1.000000
against_fire	0.202778	0.010527	-0.261570	-0.279029	-0.169489	-0.076480
against_flying	0.183343	-0.179697	0.064850	-0.111461	0.199862	-0.318941
against_ghost	0.129174	0.672337	-0.049941	-0.073031	-0.120806	-0.546982
against_grass	0.079197	-0.006533	-0.037135	0.056209	0.052899	0.269157
against_ground	-0.186841	-0.007660	-0.120042	-0.269444	-0.256504	0.358793
against_ice	0.148176	-0.010763	0.350048	-0.328531	0.273650	-0.220239
against_normal	0.215589	-0.413632	0.142035	0.076699	0.149488	-0.006997
against_poison	0.354255	-0.236919	-0.210199	-0.015769	0.146464	-0.189798
against_psychic	-0.463272	-0.230415	0.100153	-0.017592	-0.145238	-0.264938
against_rock	-0.210522	0.011963	0.090184	0.417261	-0.205444	-0.240964
against_steel	0.055504	-0.119758	-0.227697	-0.187543	0.130323	0.165066
against_water	-0.254732	-0.001976	-0.096549	-0.297600	-0.218937	0.205249
attack	-0.054175	-0.098849	0.138217	-0.104276	0.207526	0.149123
base_egg_steps	0.062133	0.187220	0.164773	-0.061970	0.120594	-0.006359
base_happiness	0.009994	0.024155	-0.151915	0.030411	-0.209323	-0.088722
base_total	-0.012398	0.065446	0.069766	-0.017137	0.098948	0.048629
defense	-0.036474	0.048039	-0.023794	-0.072433	0.001655	0.150424
experience_growth	0.035717	-0.008391	0.172547	-0.041584	0.146370	0.010407
height_m	-0.060858	0.019219	0.164464	0.003068	0.115360	0.059184
hp	0.034897	0.010589	0.089721	-0.035354	0.129284	0.109425
percentage_male	-0.048373	-0.097547	0.061785	0.051265	0.010527	0.048101
pokedex_number	0.004618	0.009066	0.000872	-0.068552	0.176651	0.018296
sp_attack	0.055352	0.170849	0.039739	0.022305	-0.010296	-0.118481
sp_defense	-0.002342	0.132507	-0.047416	0.019193	0.002754	-0.044460
speed	-0.043802	-0.000326	0.078123	0.111422	0.065401	-0.050495
weight_kg	-0.031909	0.038871	0.126003	-0.102926	0.098523	0.161564
generation	-0.001549	-0.016013	-0.025201	-0.063180	0.150801	0.000681
is_legendary	0.027864	0.136315	0.014844	-0.023151	0.050165	-0.059132

34 rows × 34 columns

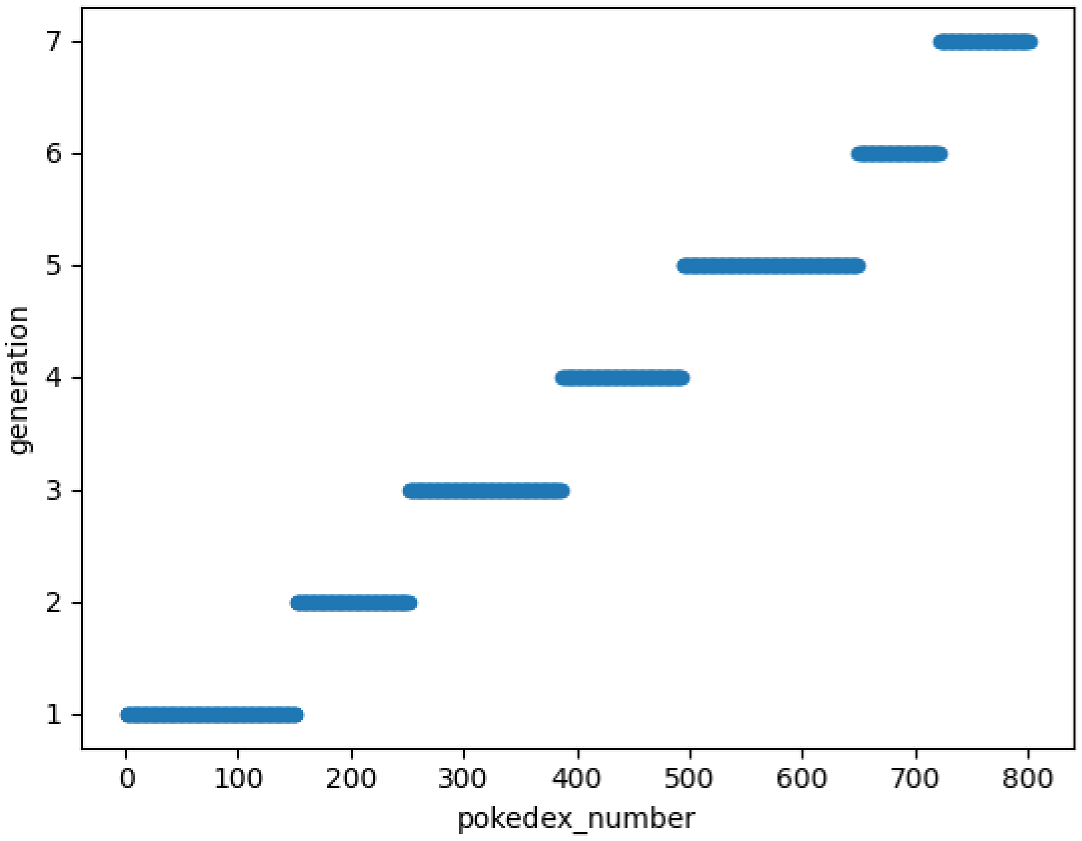
In [ ]:

```
pokemon.plot(kind="scatter", x="pokedex_number", y="generation")
```

```
C:\Users\Jens\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalC
  scatter = ax.scatter(
```

Out[ ]:

```
<AxesSubplot: xlabel='pokedex_number', ylabel='generation'>
```



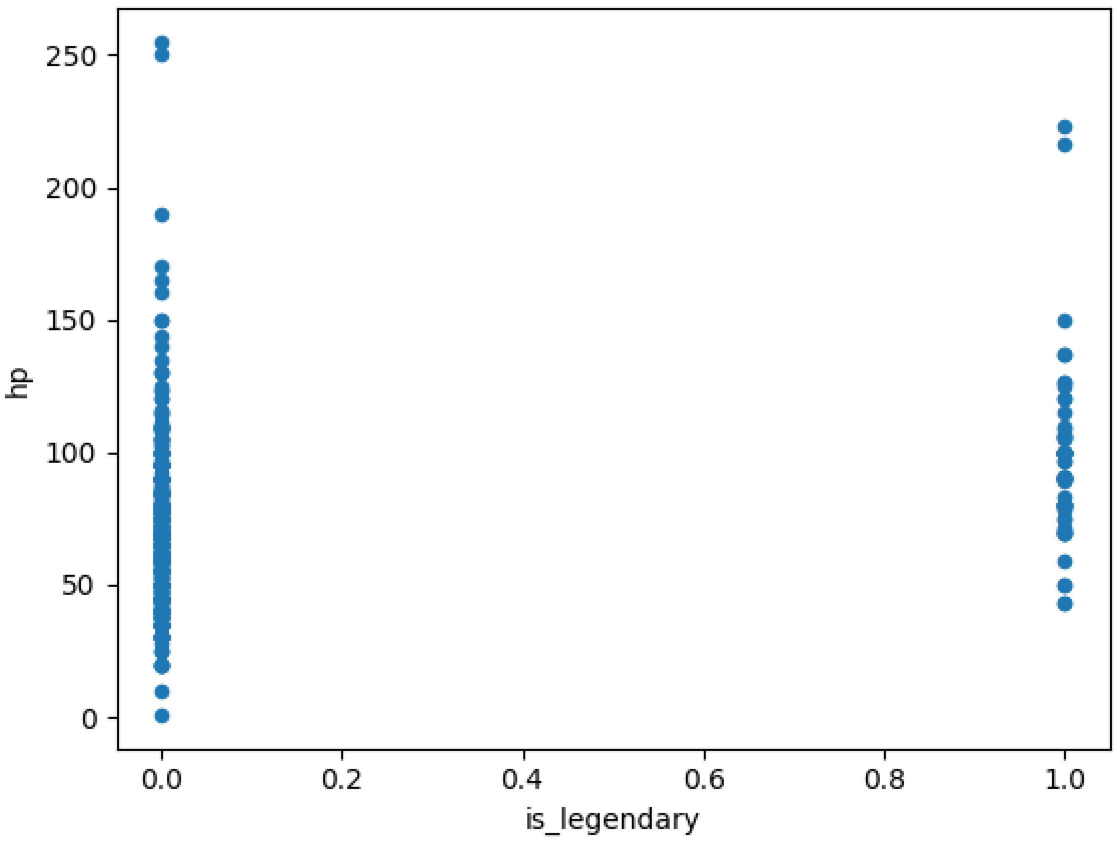
Het pokedex nummer wordt groter naarmaate we generaties hoger gaan. Mijn verwachtingen waren juist. Dit komt ook overeen met de correlation overview.

In [ ]:

```
pokemon.plot(kind="scatter", x="is_legendary", y="hp")
```

Out[ ]:

```
<AxesSubplot: xlabel='is_legendary', ylabel='hp'>
```



Dit ging tegen mijn verwachtingen in. Als niet-pokemon kenner had ik verwacht dat de HP hoger zou zijn bij legendarische pokemons. Maar dat is niet per definitie het geval blijkt uit bovenstaande scatter plot. De scatter plot komt ook overeen met de correlation overview. De correlation is namelijk erg laag en ligt dicht bij de 0.