Fact_Sheet Pandas

KI_4

2020/2021

Tobias Steiner

Inhalt

Install and Import:

`conda install pandas` → **mit diesem Command installiert man Panda**

`import pandas as pd` →**mit dem Command wird Pandas in Python Datei importiert**

Series and DataFrames:

Series sind Spalten und **DataFrames** sind multi-dimensionale Tabellen.

Creating DataFrames from scratch:

Ein einfacher weg, ein DataFrame zu erstellen ist es, ein einfaches „dict“ zu verwenden.
Bsp:

```
data = {
     ‘apples‘ : [3, 2 , 0 ,1]
     ‘orange‘ : [3, 2 , 0 ,1]
     }
```

Nachdem man es erstellt hat, muss man es in den Konstruktor des pandas DataFrames geben
Bsp:

`purchase = pd.DateFrame(data)`
`pruchase` → mit dem Befehl gibst du es dann aus

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Mit dem nächsten Befehl, können wir den Index umändern:

```
purchases = pd.DataFrame(data, index= ['June', 'Robert', 'Lily',
'David'])
purchases
```

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

Daten von einer CSV Datei lessen

`df = pd.read_csv(‘purchuases.csv’)`

`df` → **mit dem command gibt man es wieder aus**

	Unnamed: 0	apples	oranges
0	June	3	0
1	Robert	2	3
2	Lily	0	7
3	David	1	2

Als nächstes können wir unseren Index ändern:

```
df = pd.read_csv('purchuases.csv', index_col = 0)

df
```

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

Daten einlesen von JSON

```
df = pd.read_json('purchases.json')

df
```

	apples	oranges
David	1	2
June	3	0
Lily	0	7
Robert	2	3

Manchmal funktioniert das einlesen der JSON Datei nicht und dazu muss man dann das Keyword „orient" benutzen.

Link für eine Erklärung: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_json.html

Daten einlesen von SQL

Zuerst müssen wir den Command: `pip install pysqlite3`, **in unserem Terminal eingeben.**

Oder wir lassen den Command in einer Zelle unseres Notebooks laufen: `!pip install pysqlite3`

Um das dann in Python verwenden zu können muss man folgendes eingeben:

```
import sqlite3
```

`con = sqlite3.connect(„database.db")` → **Eine Verbindung mit der Datenbank wird hergestellt.**

```
df = pd.read_sql_query("SELECT * FROM pruchase", con)

df
```

	index	apples	oranges
0	June	3	0
1	Robert	2	3
2	Lily	0	7
3	David	1	2

Mit dem folgenden command kann man den Index auf die Spalte festlegen:

```
df = df.set_index('index')

df
```

	apples	oranges
index		
June	3	0
Robert	2	3
Lily	0	7
David	1	2

Zurück zu CSV, JSON oder SQL konvertieren

```
df.to_csv('new_purchases.csv')
df.to_json('new_purchases.json')
df.to_sql('new_purchases', con)
```

Wichtigsten DataFrame Operationen

```
movies_df = pd.read_csv("IMDB-Movie-Data.csv", index_col="Title")
```

Daten ansehen

Der folgende Command gibt die ersten 5 Zeilen per default aus und als Parameter kann man auch andere Mengen festlegen, das funktioniert auch mit tail(das Ende)

```
movies_df.head()
movies_df.tail()
```

Info über die Datei bekommen

```
mavies_df.info()
```

Mit dem Befehl sieht man wie viele Spalten es gibt und ihre Eigenschaften (Name, Anzahl der Einträge, Eigenschaft, Datentyp)

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, Guardians of the Galaxy to Nine Lives
Data columns (total 11 columns):
Rank                    1000 non-null int64
Genre                   1000 non-null object
Description             1000 non-null object
Director                1000 non-null object
Actors                  1000 non-null object
Year                    1000 non-null int64
Runtime (Minutes)       1000 non-null int64
Rating                  1000 non-null float64
Votes                   1000 non-null int64
Revenue (Millions)      872 non-null float64
Metascore               936 non-null float64
dtypes: float64(3), int64(4), object(4)
memory usage: 93.8+ KB
```

Mit dem shape Command gibt man aus wie viele Reihen und Spalten es gibt.

```
movies_df.sjape()
```

Duplikate

Mit dem nächsten Command duplizieren wir die Reihen:

```
temp_df = movies_df.append(movies_df)
```

```
temp_df.shape
```

Und mit dem Command dropen wir sie wieder:

```
temp_df = temp_df.drop_duplicates()
```

```
temp_df.shape
```

Jedoch geht es mit dem folgenden Command wieder leichter:

```
temp_df.drop_duplicates(inplace = True)
```

Es gibt dann für die drop Methode 3 verschiedene Zustände bei dem Keyword „keep“

- First
  - Es werden alle Duplikate bis auf das erste gedropped
- last
  - Das letzte Duplikat wird gedropped
- False
  - alle Duplikate werden gedropped

Column Cleanup

Mit dem Command: `movies_df.columns` kann man die einzelnen Spalten anzeigen.

Mit dem Keyword „rename“ kann man es dann umbenennen.

Bsp:
```
movies_df.rename(columns={
            'Runtime (Minutes)': 'Runtime',
            'Revenue (Millions)': 'Revenue_millions'
            }, inplace=True)
movies_df.columns
```

Man kann es auch anders hinzufügen:

```
movies_df.columns = ['rank', 'genre', 'description', 'director', 'actors',
'year', 'runtime', 'rating', 'votes', 'revenue_millions', 'metascore']
```

Mit dem Command ändert man die Namen der Spalten zu Kleinbuchstaben:

```
movies_df.columns = [col.lower() for col in movies_df]
```

Mit dem Command: `movie_df.isnull()` **kann man sich ausgeben lassen welche Zeilen keinen Wert beinhalten.**

Das Gleiche kann man auch mit Spalten machen: `movies_df.isnull().sum()`

Wenn man die NullWerte droppen möchte benutzt man folgenden Command:
`movies_df.dropna()`

Mit `inplace = true` **verändert man dann wirklich den DataFrame.**

Wenn man eine ganze Spalte löschen will benutz man folgenden Command:

```
movies_df.dropna(axis = 1)
```

Imputation

Mit dem folgenden Command erstellen wir einen neuen DataFrame mit dem Inhalt von der Revenue_millions Spalte + den Index revenue = movies_df['revenue_millions']

Mit dem Command: revenue_mean = revenue.mean() können wir uns den Mittelwert anzeigen lassen.

Mit dem Befehl revenue.fillna(revenue_mean, inplace=True) kann man fehlende Daten durch den Mittelwert ersetzen.

Seine Variablen Verstehen

Man kann die Werte in einem DataFrame mit folgendem Command beschreiben:

movies_df.describe()

	rank	year	runtime	rating	votes	revenue_millions	metascore
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	936.000000
mean	500.500000	2012.783000	113.172000	6.723200	1.698083e+05	82.956376	58.985043
std	288.819436	3.205962	18.810908	0.945429	1.887626e+05	96.412043	17.194757
min	1.000000	2006.000000	66.000000	1.900000	6.100000e+01	0.000000	11.000000
25%	250.750000	2010.000000	100.000000	6.200000	3.630900e+04	17.442500	47.000000
50%	500.500000	2014.000000	111.000000	6.800000	1.107990e+05	60.375000	59.500000
75%	750.250000	2016.000000	123.000000	7.400000	2.399098e+05	99.177500	72.000000
max	1000.000000	2016.000000	191.000000	9.000000	1.791916e+06	936.630000	100.000000

Man kann auch eine bestimmte Spalte beschreiben: movies_df['genre'].describe()

```
count                                1000
unique                                207
top       Action,Adventure,Sci-Fi
freq                                   50
Name: genre, dtype: object
```

Mit dem Command movies_df['genre'].value_counts().head(10) gibt man an, wie oft welche Genres vorkommen.

```
Action,Adventure,Sci-Fi      50
Drama                        48
Comedy,Drama,Romance         35
Comedy                       32
Drama,Romance                31
Action,Adventure,Fantasy     27
Comedy,Drama                 27
Animation,Adventure,Comedy   27
Comedy,Romance               26
Crime,Drama,Thriller         24
Name: genre, dtype: int64
```

Man kann auch die Beziehungen zwischen den verschiedenen Tupeln aufzeigen

movies_df.corr()

	rank	year	runtime	rating	votes	revenue_millions	metascore
rank	1.000000	-0.261605	-0.221739	-0.219555	-0.283876	-0.252996	-0.191869
year	-0.261605	1.000000	-0.164900	-0.211219	-0.411904	-0.117562	-0.079305
runtime	-0.221739	-0.164900	1.000000	0.392214	0.407062	0.247834	0.211978
rating	-0.219555	-0.211219	0.392214	1.000000	0.511537	0.189527	0.631897
votes	-0.283876	-0.411904	0.407062	0.511537	1.000000	0.607941	0.325684
revenue_millions	-0.252996	-0.117562	0.247834	0.189527	0.607941	1.000000	0.133328
metascore	-0.191869	-0.079305	0.211978	0.631897	0.325684	0.133328	1.000000

Man kann mit folgendem Command eine Series erstellen: genre_col = movies_df['genre']

```
pandas.core.series.Series
```

genre_col = movies_df[['genre']] mit diesem Command erstellt man ein DataFrame

```
pandas.core.frame.DataFrame
```

So erstellt man eine Dataframe mit mehreren Spalten: subset = movies_df[['genre', 'rating']]

Title	genre	rating
Guardians of the Galaxy	Action,Adventure,Sci-Fi	8.1
Prometheus	Adventure,Mystery,Sci-Fi	7.0
Split	Horror,Thriller	7.3
Sing	Animation,Comedy,Family	7.2
Suicide Squad	Action,Adventure,Fantasy	6.2

.loc – locates by name

.iloc – locates by numerical index

Mit diesem Command gibt man Tupel von Prometheus bis Sing aus: movie_subset = movies_df.loc['Prometheus':'Sing']

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
Split	3	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7.3	157606	138.12	62.0
Sing	4	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2016	108	7.2	60545	270.32	59.0

Mit dem nächsten Command geben wir Tupel mit dem numerischen Index von 1-3 aus: movie_subset = movies_df.iloc[1:4]

Der nächste Command zeigt mit True oder False bei welchem Film „Ridley Scott“ Direktor ist. condition = (movies_df['director'] == "Ridley Scott")

```
Title
Guardians of the Galaxy    False
Prometheus                  True
Split                      False
Sing                       False
Suicide Squad              False
Name: director, dtype: bool
```

Um zum Beispiel alle trues in einem DataFrame zu speichern benutzt man folgenden Command: movies_df[movies_df['director'] == "Ridley Scott"]

Mit diesem Command können wir die Filme ausgeben die ein höheres Rating als 8.6 haben movies_df[movies_df['rating'] >= 8.6]

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore
Interstellar	37	Adventure,Drama,Sci-Fi	A team of explorers travel through a wormhole ...	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...	2014	169	8.6	1047747	187.99	74.0
The Dark Knight	55	Action,Crime,Drama	When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...	2008	152	9.0	1791916	533.32	82.0
Inception	81	Action,Adventure,Sci-Fi	A thief, who steals corporate secrets through ...	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...	2010	148	8.8	1583625	292.57	74.0

movies_df[(movies_df['director'] == 'Christopher Nolan') | (movies_df['director'] == 'Ridley Scott')].head()

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
Interstellar	37	Adventure,Drama,Sci-Fi	A team of explorers travel through a wormhole ...	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...	2014	169	8.6	1047747	187.99	74.0
The Dark Knight	55	Action,Crime,Drama	When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...	2008	152	9.0	1791916	533.32	82.0
The Prestige	65	Drama,Mystery,Sci-Fi	Two stage magicians engage in competitive one-...	Christopher Nolan	Christian Bale, Hugh Jackman, Scarlett Johanss...	2006	130	8.5	913152	53.08	66.0
Inception	81	Action,Adventure,Sci-Fi	A thief, who steals corporate secrets through ...	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...	2010	148	8.8	1583625	292.57	74.0

```
movies_df[movies_df['director'].isin(['Christopher Nolan', 'Ridley Scott'])] //Bessere Schreibweise

movies_df[
   ((movies_df['year'] >= 2005) & (movies_df['year'] <= 2010))     & (movies_df['rating'] > 8.0)

   & (movies_df['revenue_millions'] <   movies_df['revenue_millions'].quantile(0.25))
]
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore
3 Idiots	431	Comedy,Drama	Two friends are searching for their long lost ...	Rajkumar Hirani	Aamir Khan, Madhavan, Mona Singh, Sharman Joshi	2009	170	8.4	238789	6.52	67.0
The Lives of Others	477	Drama,Thriller	In 1984 East Berlin, an agent of the secret po...	Florian Henckel von Donnersmarck	Ulrich Mühe, Martina Gedeck,Sebastian Koch, Ul...	2006	137	8.5	278103	11.28	89.0
Incendies	714	Drama,Mystery,War	Twins journey to the Middle East to discover t...	Denis Villeneuve	Lubna Azabal, Mélissa Désormeaux-Poulin, Maxim...	2010	131	8.2	92863	6.86	80.0
Taare Zameen Par	992	Drama,Family,Music	An eight-year-old boy is thought to be a lazy ...	Aamir Khan	Darsheel Safary, Aamir Khan, Tanay Chheda, Sac...	2007	165	8.5	102697	1.20	42.0

Applying functions

Mit folgendem Command können wir abfragen, wenn ein Wert ein bestimmtes Rating hat ob es gut oder schlecht ist.

```
def rating_function(x):
    if x >= 8.0:
        return "good"
    else:
        return "bad"
```

Mit dem .apply wird jeder einzelne Wert durch die Funktion durchiteriert

```
movies_df["rating_category"] = movies_df["rating"].apply(rating_function)

movies_df.head(2)
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore	rating_category
Guardians of the Galaxy	1	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1	757074	333.13	76.0	good
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0	bad

```
movies_df["rating_category"] = movies_df["rating"].apply(lambda x: 'good' if x >= 8.0 else 'bad')

movies_df.head(2)
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore	rating_category
Guardians of the Galaxy	1	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1	757074	333.13	76.0	good
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0	bad

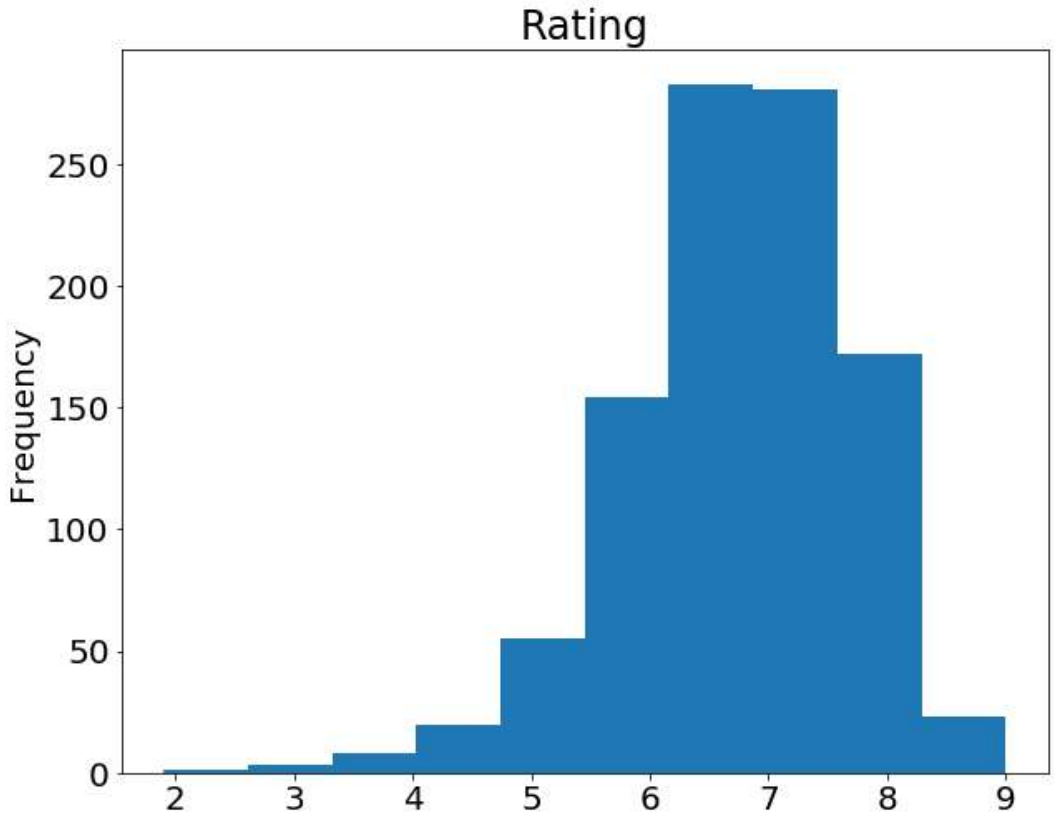
Brief Plotting

Damit wir matplotlib verwenden können müssen wir folgenden Command in der Konsole eingeben „`pip install matplotlib`“

```
import matplotlib.pyplot as plt
plt.rcParams.update({'font.size': 20, 'figure.figsize': (10, 8)}) # set font and plot size to be larger
```

Wenn wir ein Histogram ausgeben wollen müssen wir folgenden Kommand schreiben:

```
movies_df['rating'].plot(kind='hist', title='Rating');
```



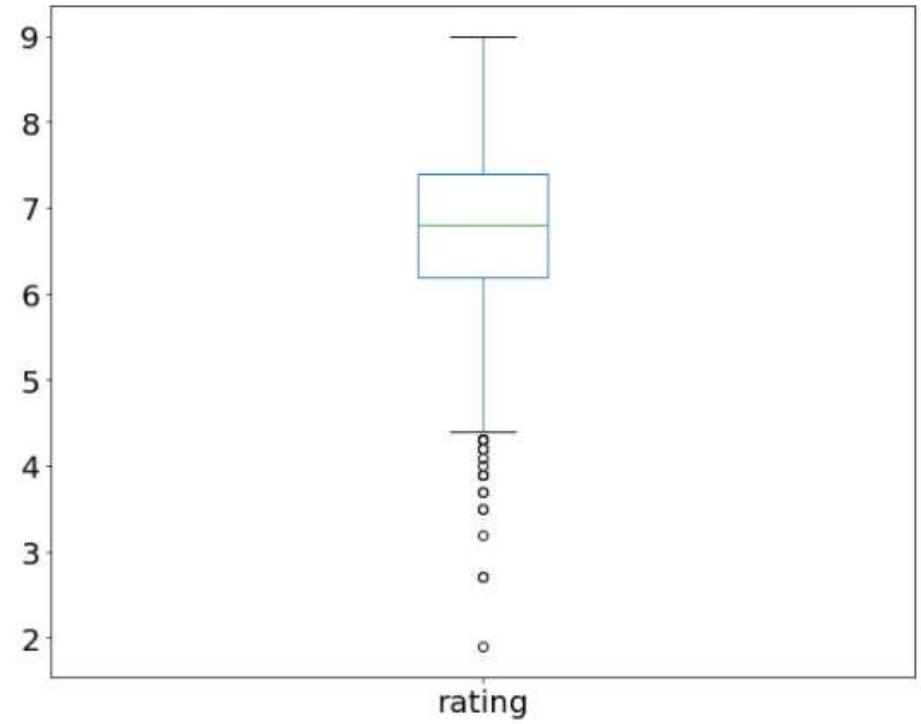
Mit dem folgendem Command können wir uns einen Boxplot ausgeben, jedoch ohne Grafik, sondern nur die Werte

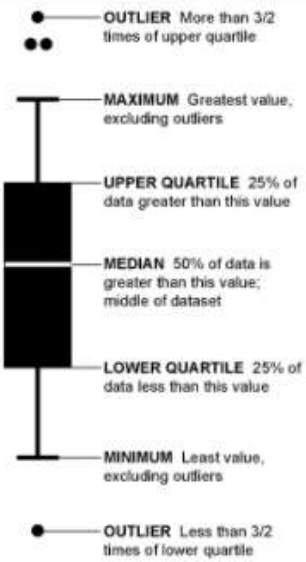
```
movies_df['rating'].describe()
```

```
count    1000.000000
mean        6.723200
std         0.945429
min         1.900000
25%         6.200000
50%         6.800000
75%         7.400000
max         9.000000
Name: rating, dtype: float64
```

Mit diesem Kommand können wir uns die Grafik ausgeben:

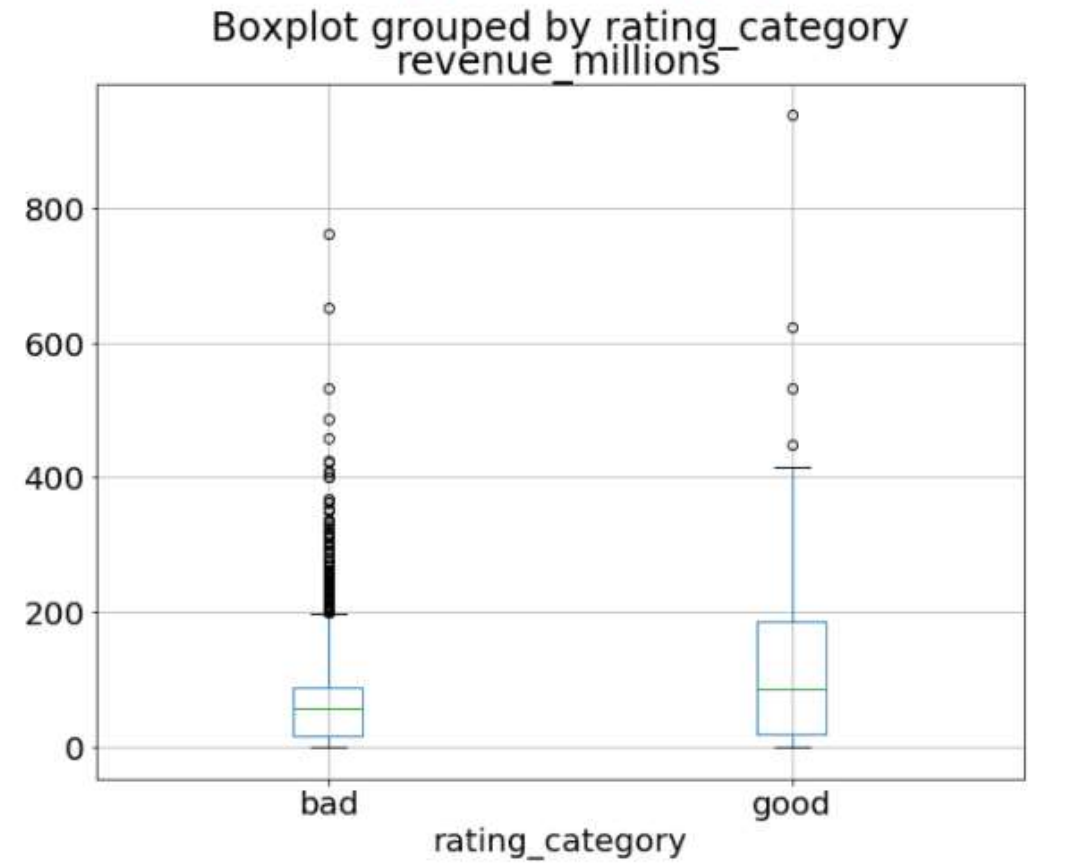
```
movies_df['rating'].plot(kind="box");
```





Mit diesem Command können wir uns zum Beispiel für die guten und schlechten Dinge einen Boxplot ausgeben:

```
movies_df.boxplot(column='revenue_millions', by='rating_category');
```



Iris-Dataset laden

```
import sklearn
from sklearn import datasets

iris = datasets.load_iris()
```

```
iris.data   # lasst alle Werte anzeigen
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
```

Mit diesem Methode kann man sich die Keys anzeigen lassen

```
iris.keys() # Zeigt nur keys vom Directory
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

Hiermit kann man einzelne Felder genauer betrachten

```
iris['feature_names']
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

Spalte hinzufügen:

```
iris_df['Species'] = 0
```

Hiermit kann man sich die Anzahl der Zeilen und Spalten anzeigen lassen.

```
iris_df.shape
```

```
(150, 5)
```

Hiermit kann man alle unterschiedlichen Werte der Spalte Spezies anzeigen lassen

```
iris_df['Species'].unique()
```

```
array([0], dtype=int64)
```

Mit unuique kann man sich die Anzahl unterschiedlicher Werte anzeigen lassen

```
iris_df['Species'].nunique()
```

```
1
```

Um Nullwerte anzuzeigen:

```
iris_df.isnull().values.any()
```

```
False
```

Hiermit kann man mögliche Korellationen zwischen den einzelnen Spalten herausfinden

```
iris_df.corr()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
sepal length (cm)	1.000000	-0.117570	0.871754	0.817941	NaN
sepal width (cm)	-0.117570	1.000000	-0.428440	-0.366126	NaN
petal length (cm)	0.871754	-0.428440	1.000000	0.962865	NaN
petal width (cm)	0.817941	-0.366126	0.962865	1.000000	NaN
Species	NaN	NaN	NaN	NaN	NaN

Data preparation with apply, applymap or map

```
def check_species(x):
    if x == 0:
        return 'SET'
    elif x == 1:
        return 'VER'
    elif x == 2:
        return 'VIR'
    return x

iris_df['Species'] = iris_df['Species'].apply(check_species)
iris_df.head(5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	SET
1	4.9	3.0	1.4	0.2	SET
2	4.7	3.2	1.3	0.2	SET
3	4.6	3.1	1.5	0.2	SET
4	5.0	3.6	1.4	0.2	SET

Hier sieht man ein Beispiel wie man eine Spalte mittels apply überschreibt und ein if einbaut

```
iris_df['wide petal'] = iris_df['petal width (cm)'].apply(lambda x: 1 if x >= 1.3 else 0)
iris_df.head(5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species	wide petal
0	5.1	3.5	1.4	0.2	SET	0
1	4.9	3.0	1.4	0.2	SET	0
2	4.7	3.2	1.3	0.2	SET	0
3	4.6	3.1	1.5	0.2	SET	0
4	5.0	3.6	1.4	0.2	SET	0

Man kann auch eine neue Spalte erstellen, die mittels einer Lambda Expression auf eine andre Spalte zugreift und entscheidet welche Werte eingetragen werden sollen

```
iris_df['petal area'] = iris_df.apply(lambda x: x['petal length (cm)'] * x['petal width (cm)'], axis=1)
iris_df.head(5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species	wide petal	petal area
0	5.1	3.5	1.4	0.2	SET	0	0.28
1	4.9	3.0	1.4	0.2	SET	0	0.28
2	4.7	3.2	1.3	0.2	SET	0	0.26
3	4.6	3.1	1.5	0.2	SET	0	0.30
4	5.0	3.6	1.4	0.2	SET	0	0.28

Ein anderes Beispiel

```
iris_df = iris_df.apply(lambda x: np.log(x) if isinstance(x, float) else x)
iris_df.head(5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species	wide petal	petal area
0	5.1	3.5	1.4	0.2	SET	0	0.28
1	4.9	3.0	1.4	0.2	SET	0	0.28
2	4.7	3.2	1.3	0.2	SET	0	0.26