Aufbauwissen Python

Standardbibliotheken = Erweiterungen für Python

- pandas (Third-Party-Library):
  - kann tabulare Daten lesen
  - DataFrames = Tabellen
  - Series = Spalten

In [1]:
```
#Bibliotheken importieren:
import urllib.request
import pandas as pd
```

In [2]:
```
#Anzahl der Reihen der Ausgabe über pandas festsetzen:
pd.set_option("display.max_rows", 10)
```

1) OpenAPC-Projekt

1.1) Datei einlesen

In [3]:
```
#CSV-Datei per Link von GitHub reinladen:
open_apc_url = "https://raw.githubusercontent.com/OpenAPC/openapc-de/mas
ter/data/fuberlin/APC_FU_Berlin_2015.csv"
```

In [4]:
```
#Datei vor dem Speichern benennen:
apc_file = "openapc.csv"
```

In [5]:
```
#Datei speichern:
#Dateien liegen default im Verzeichnis des Notebooks!
urllib.request.urlretrieve(open_apc_url, apc_file)
```

Out[5]:
```
('openapc.csv', <http.client.HTTPMessage at 0x1e4bf5a54a8>)
```

In [6]:
```
#Variable für die Datei definieren und sie in pandas einlesen:
apcs = pd.read_csv(apc_file)

#Tab (bspw.) als Separator definieren:
#apcs = pd.read_csv(apc_file, sep = "\t")
#(hier nicht nötig, da 'pd.read_csv' bereits einen Standardseparator set
zt)
```

In [7]:
```
#Datei ansehen (normale Ansicht im Jupyter Notebook):
#print(apcs)

#Schöner (Dank pandas!):
apcs
```

Out[7]:

	institution	period	euro	doi	is_hybrid	pul
0	FU Berlin	2015	2000.00	10.1038/npjqi.2015.10	False	Spr Nat
1	FU Berlin	2015	1503.12	10.1103/PhysRevX.5.041008	False	Am Phy Soc (AP
2	FU Berlin	2015	2000.00	10.1038/ncomms9498	False	Spr Nat
3	FU Berlin	2015	2000.00	10.1371/journal.ppat.1005246	False	Pul Lib Sci (PL
4	FU Berlin	2015	1805.00	10.1186/s13099-015-0075-z	False	Spr Nat
...	...	...	...	...	...	...
56	FU Berlin	2015	1731.45	10.1186/s13015-014-0028-y	False	Spr Nat
57	FU Berlin	2015	1600.00	10.3389/fpsyg.2015.00194	False	Fro Me
58	FU Berlin	2015	219.84	10.3389/fevo.2015.00020	False	Fro Me
59	FU Berlin	2015	1386.35	10.1038/srep19416	False	Spr Nat
60	FU Berlin	2015	2000.00	10.1155/2015/569512	False	Hin Pul Co

61 rows × 17 columns

1.2) Spalten anzeigen lassen:

In [8]:
```
#Eine bestimmte Spalte anzeigen:
apcs["euro"]

#Geht auch, ist aber fehleranfälliger:
apcs.euro
```

Out[8]:
```
0     2000.00
1     1503.12
2     2000.00
3     2000.00
4     1805.00
       ...
56    1731.45
57    1600.00
58     219.84
59    1386.35
60    2000.00
Name: euro, Length: 61, dtype: float64
```

In [9]:
```
#Zwei Spalten anzeigen:
apcs[["euro", "doi"]]
```

Out[9]:

	euro	doi
0	2000.00	10.1038/npjqi.2015.10
1	1503.12	10.1103/PhysRevX.5.041008
2	2000.00	10.1038/ncomms9498
3	2000.00	10.1371/journal.ppat.1005246
4	1805.00	10.1186/s13099-015-0075-z
...	...	...
56	1731.45	10.1186/s13015-014-0028-y
57	1600.00	10.3389/fpsyg.2015.00194
58	219.84	10.3389/fevo.2015.00020
59	1386.35	10.1038/srep19416
60	2000.00	10.1155/2015/569512

61 rows × 2 columns

In [10]:
```
#Spalte mit Bedingung ausgeben:
apcs["euro"]>= 2000
```

Out[10]:
```
0      True
1     False
2      True
3      True
4     False
      ...
56    False
57    False
58    False
59    False
60     True
Name: euro, Length: 61, dtype: bool
```

1.3) Zeilen anzeigen lassen:

In [11]:
```
#Zeilen mit Bedingung ausgeben:
apcs[apcs["euro"]>= 2000]
```

Out[11]:

	institution	period	euro	doi	is_hybrid	publ
0	FU Berlin	2015	2000.0	10.1038/npjqi.2015.10	False	Spri Natu
2	FU Berlin	2015	2000.0	10.1038/ncomms9498	False	Spri Natu
3	FU Berlin	2015	2000.0	10.1371/journal.ppat.1005246	False	Publ Libra Scie (PLc
12	FU Berlin	2015	2000.0	10.3389/fpsyg.2015.01675	False	Fron Med
15	FU Berlin	2015	2000.0	10.1371/journal.ppat.1004781	False	Publ Libra Scie (PLc
...	...	...	...	...	...	...
28	FU Berlin	2015	2000.0	10.3389/fnhum.2015.00551	False	Fron Med
29	FU Berlin	2015	2000.0	10.1038/ncomms8606	False	Spri Natu
32	FU Berlin	2015	2000.0	10.1186/s12918-015-0183-x	False	Spri Natu
45	FU Berlin	2015	2000.0	10.1371/journal.pcbi.1004200	False	Publ Libra Scie (PLc
60	FU Berlin	2015	2000.0	10.1155/2015/569512	False	Hind Publ Corp

11 rows × 17 columns

In [12]:
```
#DataFrame für die Abfrage definieren:
filtered_apcs = apcs[apcs["euro"]>= 2000]
```

In [13]:
```
#DataFrame als Excel-Datei speichern:
#Dateien liegen default im Verzeichnis des Notebooks!
filtered_apcs.to_excel("filtered_apcs.xlsx")
```

1.4) Diagramm anzeigen lassen:

In [14]:
```
#Zeige mir das Diagramm innerhalb des Notebooks an:
%matplotlib inline
```

In [15]:
```
#Diagramm ausgeben:
#.hist = Histogramm
apcs["euro"].hist()
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4bc456cc0>

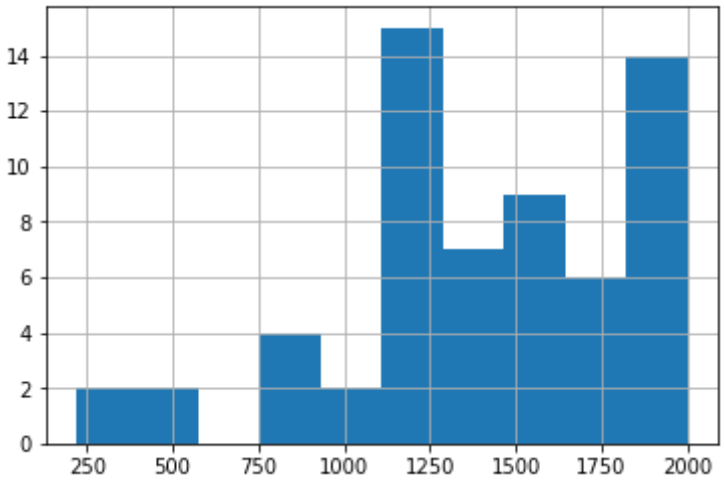
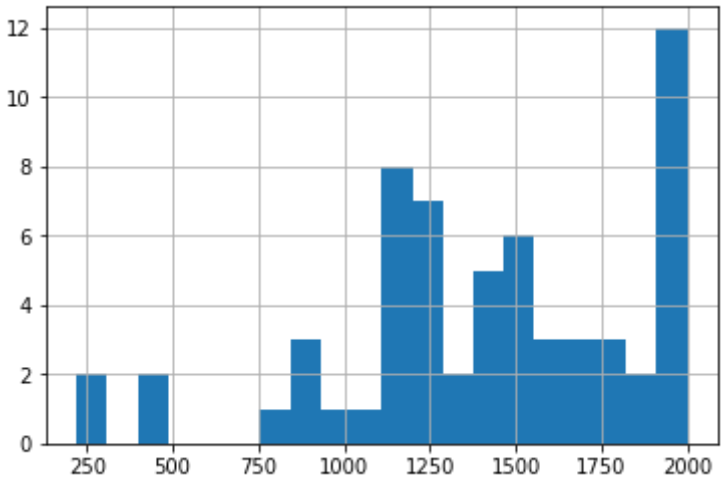


Diagramme modifizieren:

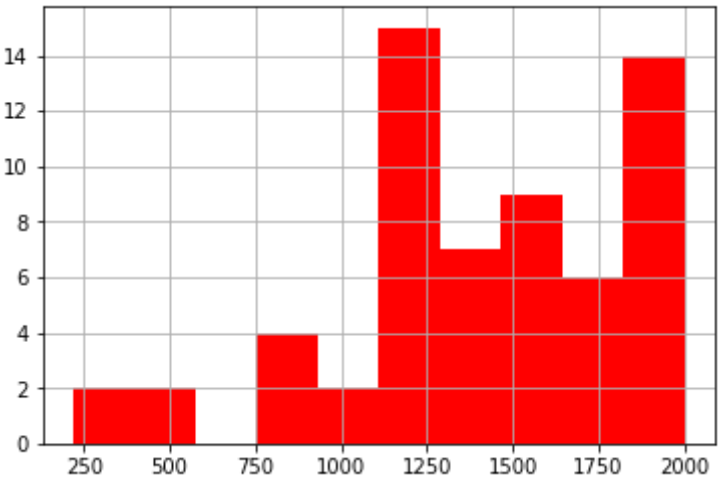
In [16]:
```
#Abstände definieren:
apcs["euro"].hist(bins = 20)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4c00a6080>



In [17]:
```
#Farbe ändern:
apcs["euro"].hist(color = "red")
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e4bc456860>
```



Eigenschaften eines DataFrames herausfinden:

In [18]:
```
#Gibt Typ des DataFrames zurück:
type(filtered_apcs)
```

Out[18]:
```
pandas.core.frame.DataFrame
```

In [19]:
```
#Gibt Umfang des DataFrames zurück:
filtered_apcs.shape
```

Out[19]:
```
(11, 17)
```

In [20]:
```
#Gibt alle Spaltennamen des DataFrames zurück:
filtered_apcs.columns
```

Out[20]:
```
Index(['institution', 'period', 'euro', 'doi', 'is_hybrid', 'publisher',
       'journal_full_title', 'issn', 'issn_print', 'issn_electronic',
       'license_ref', 'indexed_in_crossref', 'pmid', 'pmcid', 'ut', 'ur
l',
       'doaj'],
      dtype='object')
```

Werte von Series modifizieren:

In [21]:
```
#Multiplizieren:
apcs["euro"] * 1000
```

Out[21]:
```
0     2000000.0
1     1503120.0
2     2000000.0
3     2000000.0
4     1805000.0
        ...
56    1731450.0
57    1600000.0
58     219840.0
59    1386350.0
60    2000000.0
Name: euro, Length: 61, dtype: float64
```

In [22]:
```
#Runden:
apcs["euro"].apply(round)
```

Out[22]:
```
0     2000
1     1503
2     2000
3     2000
4     1805
      ...
56    1731
57    1600
58     220
59    1386
60    2000
Name: euro, Length: 61, dtype: int64
```

Sonstiges:

In [23]:
```
#Hilfe zu einem Befehl ausgeben lassen:
?pd.read_csv
```

In [24]:
```
#Prozentzeichen = "Jupyter Magic"
#Befehl richtet sich nicht Python, sondern an das Notebook
%who
```

```
apc_file         apcs    filtered_apcs   open_apc_url    pd      urllib
```