SIRE504 Introduction to Python - part 3

08.11.2018

Harald Grove

harald.gro@mahidol.ac.th

Topics

- Basic topics
  - Error handling
  - Passing parameters to functions
- Pandas
  - Data analysis
- Plotting with python

Syntax errors

- Part of the code is not “Python”.
- The parser encounters syntax that is not understood.

Common causes:

- Missing parenthesis ({ [
- Missing colons :
- Missing quotes ‘ ”

```
$ python read_fasta.py count dengue1.fna dengue1_count.txt
  File "read_fasta.py", line 5
    outputfile.write('{}\n'.format(name))
             ^
SyntaxError: invalid syntax
```

```
def countbases(name, sequence, outputfile):
    bases = ['A', 'C', 'G', 'T', 'N'
    outputfile.write('{}\n'.format(name))
    for base in bases:
        count = sequence.count(base)
        outputfile.write('{}: {}\n'.format(base, count))
```

Exceptions

- The code can be legal Python syntax and still have errors
    - Exceptions
- Exceptions are found when the parser tries to execute the code
    - Important to test every part of your code.
- These exceptions are classified into classes
    - Naming scheme: <class name>Error
    - e.g. NameError, TypeError

Errors/ Exceptions

```
$ python read_fasta.py count dengue1.fna
Traceback (most recent call last):
  File "read_fasta.py", line 38, in <module>
    action = sys.argv[1]
NameError: name 'sys' is not defined

$ python read_fasta.py count dengue1.fna
Traceback (most recent call last):
  File "read_fasta.py", line 40, in <module>
    outputfilename = sys.argv[3]
IndexError: list index out of range

$ python read_fasta.py count dengue1.fna dengue1_count.txt
Traceback (most recent call last):
  File "read_fasta.py", line 54, in <module>
    print(database['sequence4'])
KeyError: 'sequence4'

$ python read_fasta.py count dengue1.fna dengue1_count.txt
Traceback (most recent call last):
  File "read_fasta.py", line 65, in <module>
    action.close()
AttributeError: 'str' object has no attribute 'close'

$ python read_fasta.py count dengue1.fna dengue1_count.txt
Traceback (most recent call last):
  File "read_fasta.py", line 55, in <module>
    countbases(name, sequence, outputfile, 100)
TypeError: countbases() takes 3 positional arguments but 4 were given
```

https://docs.python.org/3/library/exceptions.html#bltin-exceptions

Exceptions can be useful

- The Python parser does not stop immediately an exception is found.
- It exits the current execution and sends a report about the incident.
    - Similar to the way the “return “ statement exits a function.
- This report can be caught before it reaches the top level.
- Syntax:

```
try:
      <some code>
except <name of exception>:
      <code to deal with the error>
```

- Errors not belonging to the chosen exception class will not be affected.

Handling the exception

- Provide a more informative feedback to the user
- Instead of this:

```
$ python read_fasta.py count dengue1.fna
Traceback (most recent call last):
  File "read_fasta.py", line 40, in <module>
    outputfilename = sys.argv[3]
IndexError: list index out of range
```

- You could display this:

```
$ python read_fasta.py count dengue1.fna
ERROR: Not enough command line parameters.
```

- Using this code:

```
try:
    outputfilename = sys.argv[3]
except IndexError:
    print('ERROR: Not enough command line parameters.')
    sys.exit(1)
```

This is also an exception, we use it to exit the program because we've told the user what the problem is and now we need the user to take some action.

Handling the exception

- Dealing with the problem to stop the program from crashing
- Instead of this:

```
$ python read_fasta.py count dengue1.fna
Traceback (most recent call last):
  File "read_fasta.py", line 40, in <module>
    outputfilename = sys.argv[3]
IndexError: list index out of range
```

- You could display this:

```
$ python read_fasta.py count dengue1.fna
WARNING: Output file not specified, using default “file.out”.
```

- Using this code:

```
try:
    outputfilename = sys.argv[3]
except IndexError:
    print('WARNING: Output file not specified, using default “file.out”.')
    outputfilename = ‘file.out’
```

This time we don’t exit the program because we have solved the problem of the missing file.

Warning

- Be careful when letting the program continue after catching an exception
- You are assuming that you know all the reasons for why the exception was triggered.

```
try:
    outputfilename = sys.argv[3]
except IndexError:
    print('WARNING: Output file not specified, using default “file.out”.')
    outputfilename = ‘file.out’
```

- You only know that there are too few parameters
    - The user could have forgotten to enter the input file or the action to take.
- Always be specific about what exceptions you want to handle

Namespaces

Namespaces

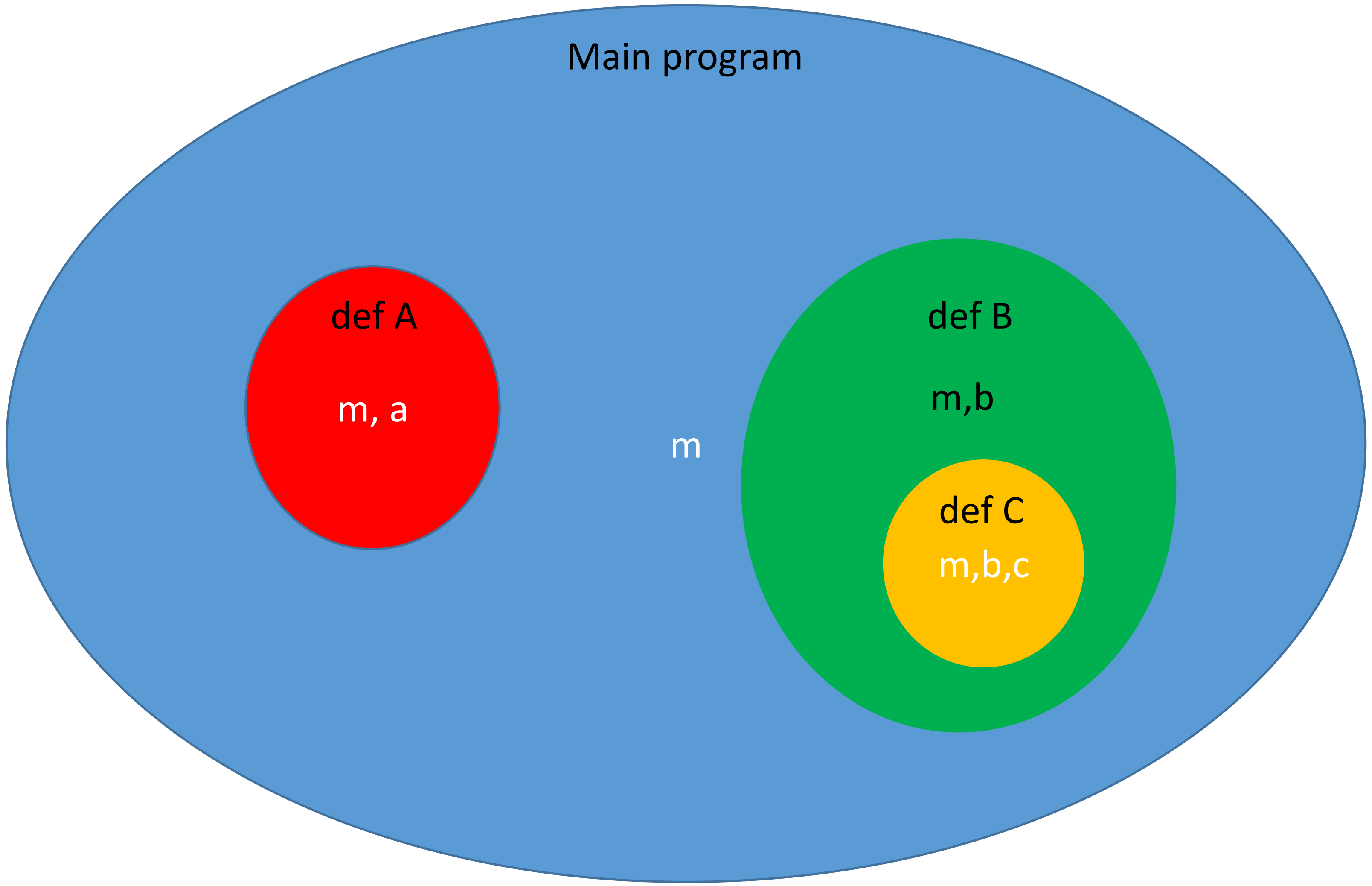
- Variables have to be assigned a value (even empty) before being used
- Variables in python have different visibility
    - a visible variable is one that has been assigned a value and can be used in the code
- Variables created at top level can be seen from everywhere (global).
- Variables created inside functions cannot be seen from outside (local).
- It is possible that a local and a global variable can have the same name, while still being different.

Not recommended!

Namespaces

```
def A():
    a = 1
def B():
    def C():
        c = 3
    b = 2

m = 4
```



Functions and input parameters

```
def function(mylist):
    mylist[4] = ‘A’
alphabet = [‘A’,’B’,’C’,’D’,’E’]
function(alphabet)
print(alphabet)
```

Functions and input parameters

- Variables with mutable data types are passed by reference
  - The function is seeing the exact same object as the main program.
  - Changes to the variable inside the function will be kept after the function ends.
- This also applies when assigning one variable to another

```
>>> a = [1, 2, 3]
>>> b = a
>>> a[0] = 3
>>> print(b)
[3, 2, 3]
```

Copying mutable objects

- Lists and dictionaries have a method called copy().
  - mylist.copy()
- This creates a new object with the same content

```
>>> a = [1, 2, 3]
>>> b = a.copy()
>>> a[0] = 3
>>> print(b)
[1, 2, 3]
```

Copying mutable objects

- The method “copy” is only copying the first level
  - Elements of a list can also be lists

```
>>> a = [[1,2,3], 2, 3]
>>> b = a.copy()
>>> a[0][0] = 3
>>> print(b)
[[3,2,3], 2, 3]
```

Shallow vs. deep copy

- Shallow copy: Only copies the first level
- Deep copy: Makes a complete copy of all items.

```
>>> import copy
>>> a = [[1,2,3],2,3]
>>> b = copy.deepcopy(a)
>>> a[0][0] = 3
>>> print(b)
[[1,2,3],2,3]
>>> print(a)
[[3,2,3],2,3]
```

Data analysis

Pandas and Numpy

Pandas

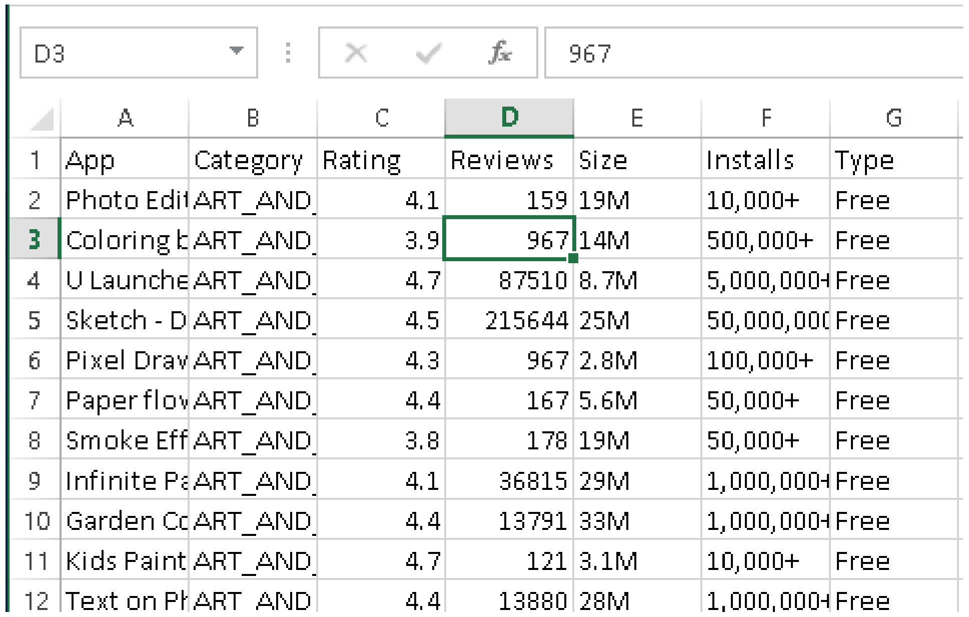
- Python module for working with numerical data
  - Data tables/matrixes
- Data analysis
- Machine learning

Example data set

- Kaggle (https://www.kaggle.com/)
    - A website for practicing machine learning and data analysis
    - Freely available data sets
    - Tutorials
    - Competitions
- Example: googleplaystore
    - Data scraped from the Google play store
    - A selection of apps with various data stored as a csv

Excel vs. Pandas

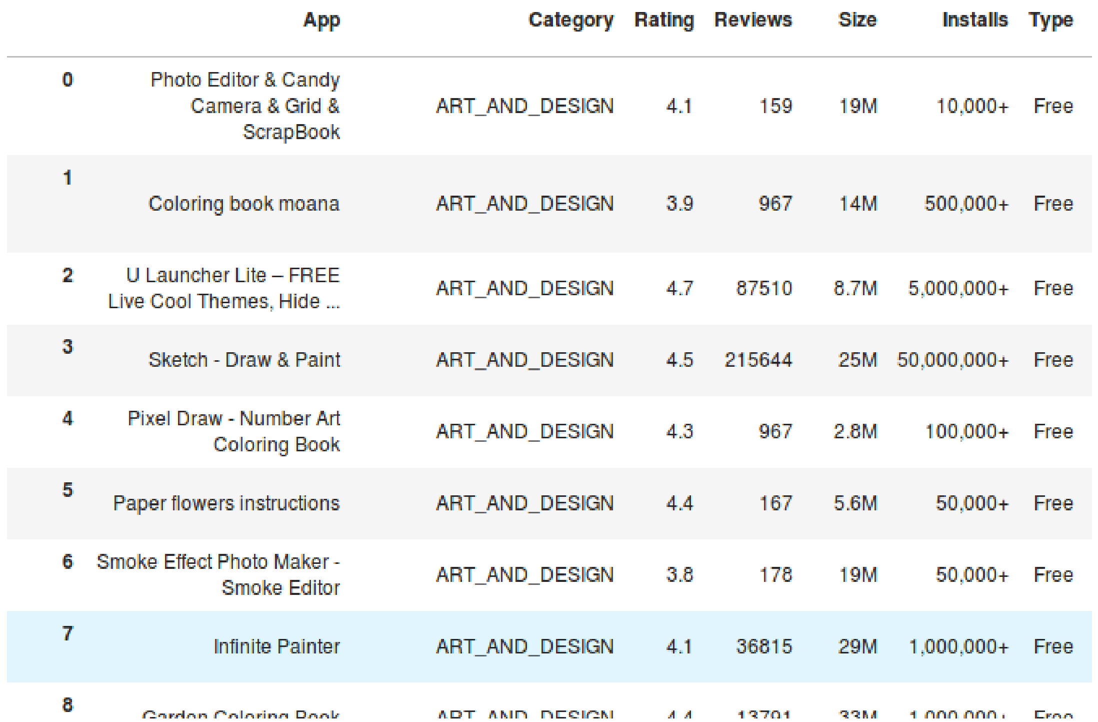
googleplaystores.csv



D3 | 967

	A	B	C	D	E	F	G
1	App	Category	Rating	Reviews	Size	Installs	Type
2	Photo Edit	ART_AND_	4.1	159	19M	10,000+	Free
3	Coloring b	ART_AND_	3.9	967	14M	500,000+	Free
4	U Launche	ART_AND_	4.7	87510	8.7M	5,000,000+	Free
5	Sketch - D	ART_AND_	4.5	215644	25M	50,000,000	Free
6	Pixel Draw	ART_AND_	4.3	967	2.8M	100,000+	Free
7	Paper flow	ART_AND_	4.4	167	5.6M	50,000+	Free
8	Smoke Eff	ART_AND_	3.8	178	19M	50,000+	Free
9	Infinite Pa	ART_AND_	4.1	36815	29M	1,000,000+	Free
10	Garden Co	ART_AND_	4.4	13791	33M	1,000,000+	Free
11	Kids Paint	ART_AND_	4.7	121	3.1M	10,000+	Free
12	Text on Ph	ART_AND_	4.4	13880	28M	1,000,000+	Free

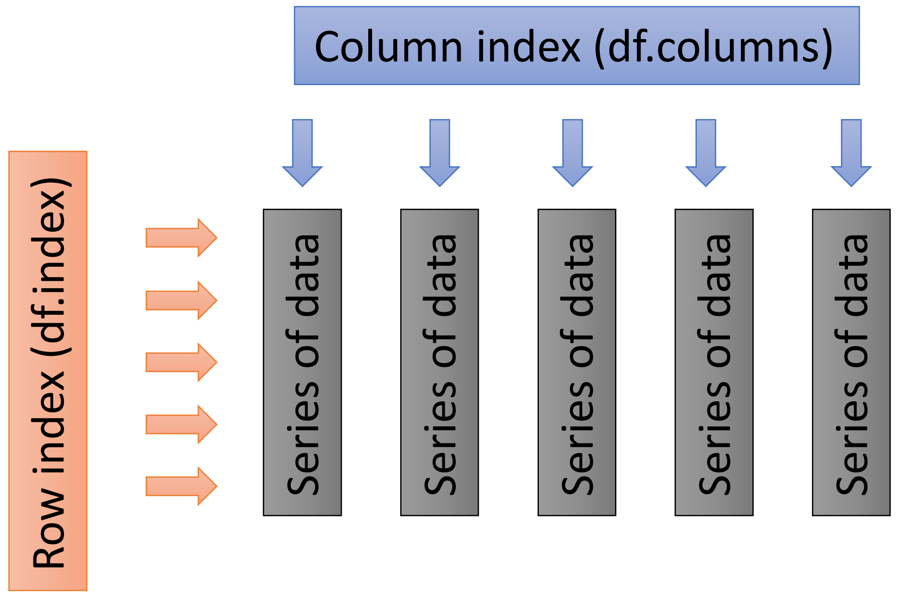
```
import pandas
df = pandas.read_csv("googleplaystores.csv")
```



	App	Category	Rating	Reviews	Size	Installs	Type
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free
6	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178	19M	50,000+	Free
7	Infinite Painter	ART_AND_DESIGN	4.1	36815	29M	1,000,000+	Free
8	Garden Coloring Book	ART_AND_DESIGN	4.4	13791	33M	1,000,000+	Free

Conceptual model

- Data is stored in pandas as:
  - Series (1D)
  - DataFrames (2D)



Importing data from files

```
import pandas as pd
```

The alias "pd" is commonly used as an abbreviation of "pandas".

From csv-files:

```
>>> df = pd.read_csv('googleplaystore.csv')
```

Parameters that are useful to know:

- sep: column separator, default =","
- header: Row number to use as column header, default=0

From excel-files (default is to import the first sheet):

```
>>> df = pd.read_excel('googleplaystore.xlsx')
```

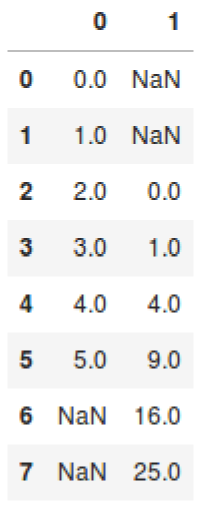
Parameters that are useful to know:

- sheet_name: Sheet to import, can be names or 0-indexed, default =0
- header: Row number to use as column header, default=0

Creating data frames from python objects

Python lists / Pandas Series:

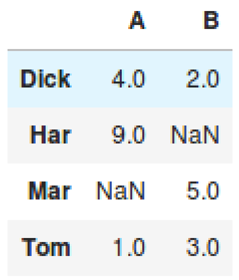
```
>>> a = [1, 2, 3, 4, 5]
>>> s1 = pd.Series(a)
>>> s2 = s1 * s1
>>> s2.index = s2.index + 2
>>> df = pd.concat([s1,s2], axis=1)
```



	0	1
0	0.0	NaN
1	1.0	NaN
2	2.0	0.0
3	3.0	1.0
4	4.0	4.0
5	5.0	9.0
6	NaN	16.0
7	NaN	25.0

Python dicts / Pandas Series:

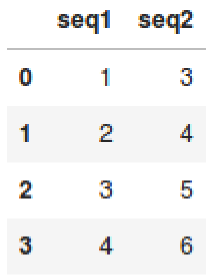
```
>>> s3 = pd.Series({‘Tom’:1, ‘Dick’:4, ‘Har’:9})
>>> s4 = pd.Series({‘Tom’:3, ‘Dick’:2, ‘Mar’:5})
>>> df = pd.concat({‘A’:s3, ‘B’:s4}, axis = 1)
```



	A	B
Dick	4.0	2.0
Har	9.0	NaN
Mar	NaN	5.0
Tom	1.0	3.0

Python dictionaries:

```
>>> d1 = {‘seq1’:[1,2,3,4], ‘seq2’:[3,4,5,6]}
>>> df = pd.DataFrame(d1)
```



	seq1	seq2
0	1	3
1	2	4
2	3	5
3	4	6

Overview

```
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10840 entries, 0 to 10839
Data columns (total 13 columns):
App                  10840 non-null object
Category             10840 non-null object
Rating               9366 non-null float64
Reviews              10840 non-null int64
Size                 10840 non-null object
Installs             10840 non-null object
Type                 10839 non-null object
Price                10840 non-null object
Content Rating       10840 non-null object
Genres               10840 non-null object
Last Updated         10840 non-null object
Current Ver          10832 non-null object
Android Ver          10838 non-null object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

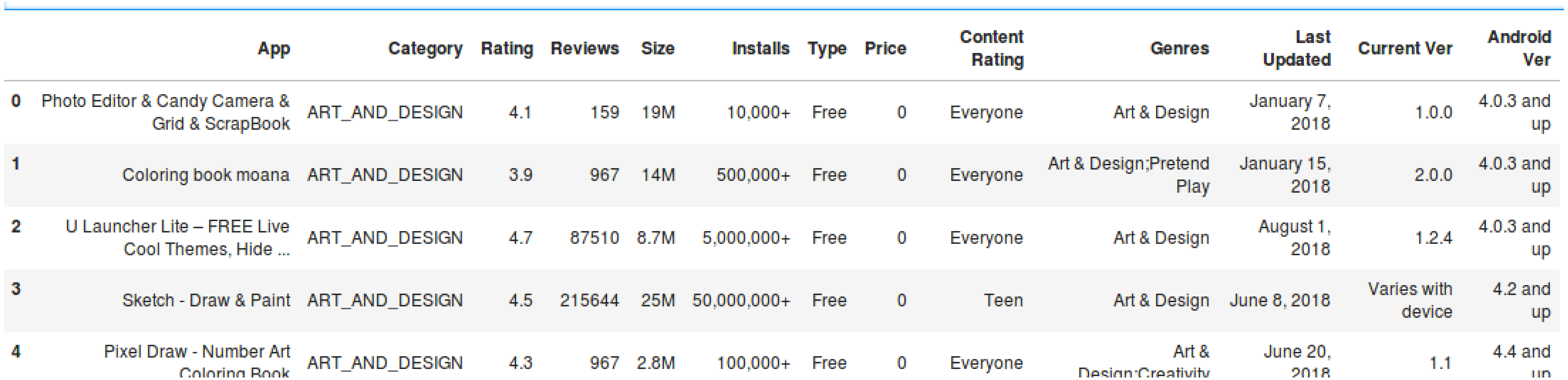
- Total number of lines/entries
- Remember, index starts at 0!

- Content of the column
  - float64 = only floating point numbers
  - int64 = only integer numbers
  - object = anything else

- Number of elements
- NaN is reffered to as ‘null’ and is not counted

Quick peak

```
>>> df.head()
```



	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up

Summarize

Default is to only show numerical columns, this will force all columns to be included

```
>>> df.describe(include = 'all')
```

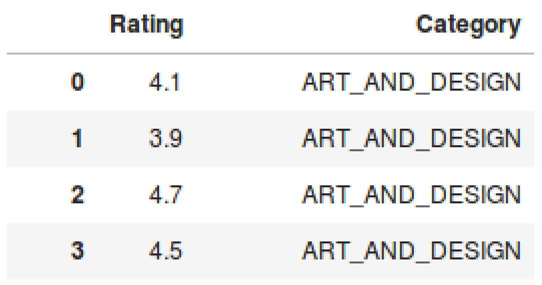
	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
count	10840	10840	9366.000000	1.084000e+04	10840	10840	10839	10840	10840	10840	10840	10832	10838
unique	9659	33	NaN	NaN	461	21	2	92	6	119	1377	2831	33
top	ROBLOX	FAMILY	NaN	NaN	Varies with device	1,000,000+	Free	0	Everyone	Tools	August 3, 2018	Varies with device	4.1 and up
freq	9	1972	NaN	NaN	1695	1579	10039	10040	8714	842	326	1459	2451
mean	NaN	NaN	4.191757	4.441529e+05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	0.515219	2.927761e+06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	1.000000	0.000000e+00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	4.000000	3.800000e+01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	4.300000	2.094000e+03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	4.500000	5.477550e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	5.000000	7.815831e+07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Two numerical columns

1D selections

Columns:

```
>>> df[[‘Rating’,’Category’]]
```

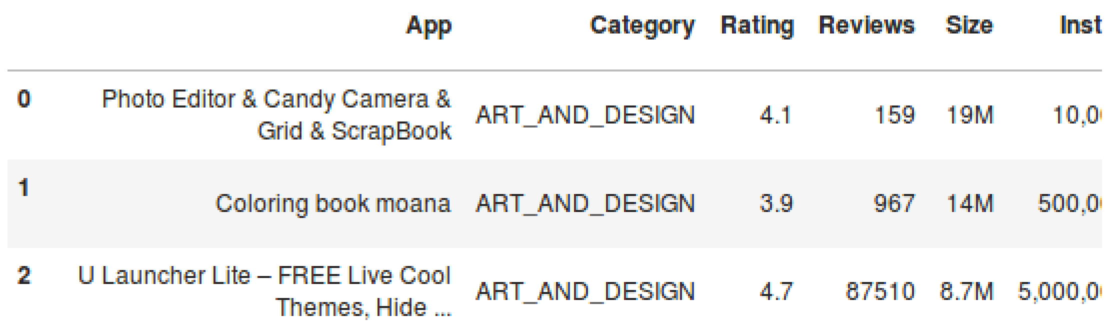


	Rating	Category
0	4.1	ART_AND_DESIGN
1	3.9	ART_AND_DESIGN
2	4.7	ART_AND_DESIGN
3	4.5	ART_AND_DESIGN

Note order of columns is different to how it is in the datafram.

Rows:

```
>>> df[0:3]
```

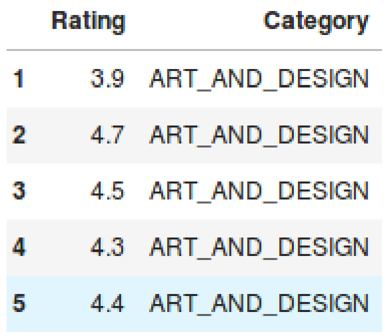


	App	Category	Rating	Reviews	Size	Inst
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,0
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,0
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,0

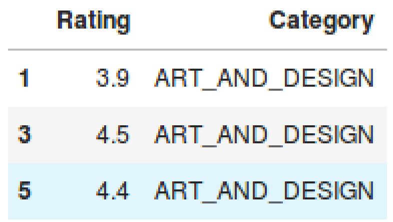
2D selections

Rows and columns by label:

```
>>> df.loc[1:5, [‘Rating’, ‘Category’]]
>>> df.loc[[1,3,5], [‘Rating’, ‘Category’]]
```



	Rating	Category
1	3.9	ART_AND_DESIGN
2	4.7	ART_AND_DESIGN
3	4.5	ART_AND_DESIGN
4	4.3	ART_AND_DESIGN
5	4.4	ART_AND_DESIGN

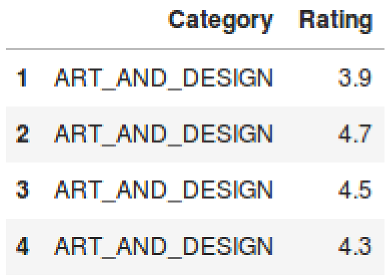


	Rating	Category
1	3.9	ART_AND_DESIGN
3	4.5	ART_AND_DESIGN
5	4.4	ART_AND_DESIGN

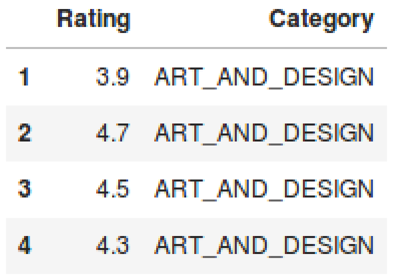
Note, stop is included.

Rows and columns by index:

```
>>> df.iloc[1:5, 1:3]
>>> df.iloc[1:5, [2,1]]
```



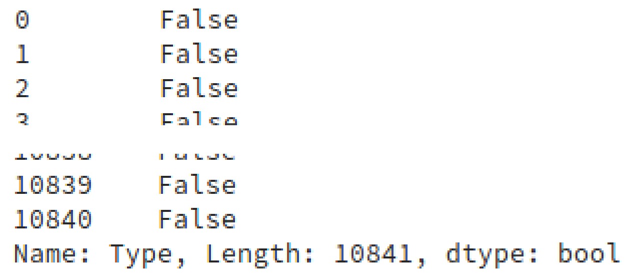
	Category	Rating
1	ART_AND_DESIGN	3.9
2	ART_AND_DESIGN	4.7
3	ART_AND_DESIGN	4.5
4	ART_AND_DESIGN	4.3



	Rating	Category
1	3.9	ART_AND_DESIGN
2	4.7	ART_AND_DESIGN
3	4.5	ART_AND_DESIGN
4	4.3	ART_AND_DESIGN

Note, stop is **not** included.

Filter by criteria



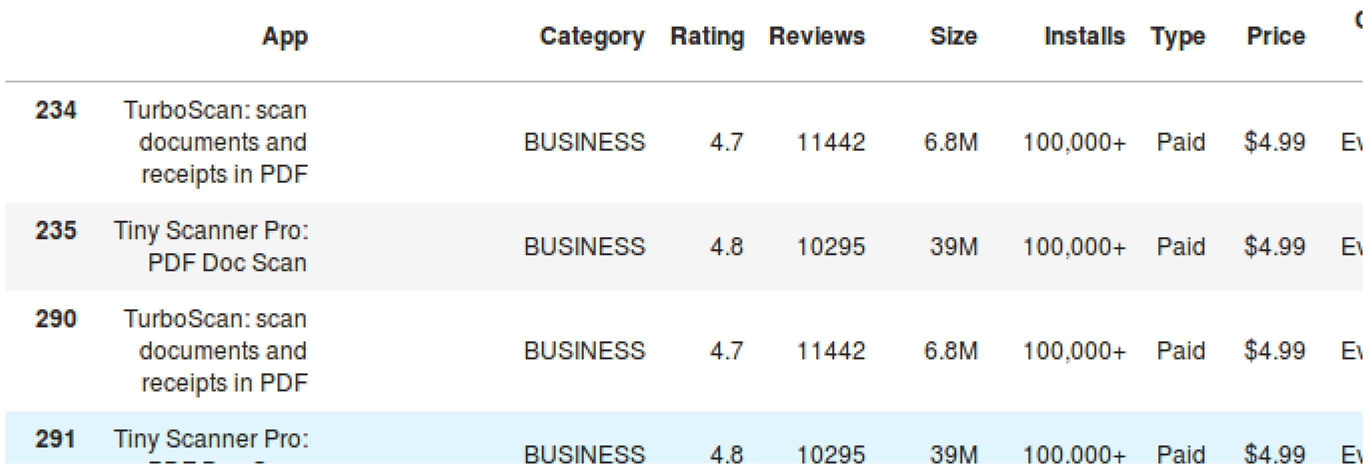
```
0        False
1        False
2        False
3        False
10839    False
10840    False
Name: Type, Length: 10841, dtype: bool
```

We can create an index by comparing a column to a given criteria.

```
>>> df['Type'] == 'Paid'
```

This index can be used as a selection criteria:

```
>>> df[df['Type'] == 'Paid']
```



	App	Category	Rating	Reviews	Size	Installs	Type	Price	
234	TurboScan: scan documents and receipts in PDF	BUSINESS	4.7	11442	6.8M	100,000+	Paid	$4.99	Ev
235	Tiny Scanner Pro: PDF Doc Scan	BUSINESS	4.8	10295	39M	100,000+	Paid	$4.99	Ev
290	TurboScan: scan documents and receipts in PDF	BUSINESS	4.7	11442	6.8M	100,000+	Paid	$4.99	Ev
291	Tiny Scanner Pro:	BUSINESS	4.8	10295	39M	100,000+	Paid	$4.99	Ev

Multiple criteria is also possible:

```
>>> df[(df['Type']=='Paid') & (df['Category']=='DATING')]
```

Note: Parenthesis

Note:
- & = and
- | = or

Dealing with missing data

Remove any row with missing values:

```
>>> df.dropna()
```

Fill missing values with a given value

```
>>> df.fillna(0)
```

Index missing values (use for filtering):

```
>>> df[‘Rating’].isnull()
>>> df[‘Rating’].notnull()
```

Search and replace

Use df.replace(from, to) to replace full entry in all cells

```
>>> df.replace(‘TOOLS’, ‘tools’)
```

Price should be numeric, but pandas doesn’t understand currencies (the “$”-sign).
To replace subsets of text in cells, use regex:

```
>>> df['Price'] = df['Price'].replace('\$','', regex=True)
```

To also convert data type:

```
>>> df['Price'] = df['Price'].replace('\$','', regex=True).astype(float)
```

The $-sign has a special meaning, so the “\” character is used to cancel that meaning.

Apply

- Sometimes there is no easy ready-made function to modify cells
- Then we write our own and tell pandas to use that function

```
>>> df.apply(our_own_function)
```

- We want to convert the size column to just numbers

```
def fix_size(s):
    if s.endswith('M'):
        return float(s[:-1])
    if s.endswith('k'):
        return float(s[:-1])/1000
    if s.endswith('G'):
        return float(s[:-1])*1000
    return None
```

The input is going to be the content of one cell.

Pandas will insert a NaN when it sees None.

```
>>> df['Size'] = df['Size'].apply(fix_size)
```

We replace the "Size" column with the new column created by apply.

Loop over rows

Creating loops in Pandas is usually not what we want, but it can help when we don't know any other way.

Removing the $-sign, cell by cell, and creating a new column:

```
>>> a = []
>>> for index, row in df.iterrows():
>>>    a.append(float(row['Price'].strip('$')))
>>> df['newPrice'] = a
```

Access the cells in each row the same way as accessing the column in the whole dataframe.

Cleaning the data

```
>>> df = pd.read_csv('googleplaystore.csv')
>>> df['Price'] = df['Price'].replace('\$','',regex=True).astype(float)
>>> df[‘Size’] = df[‘Size’].apply(fix_size)
>>> df = df.dropna()
```

Explanation:

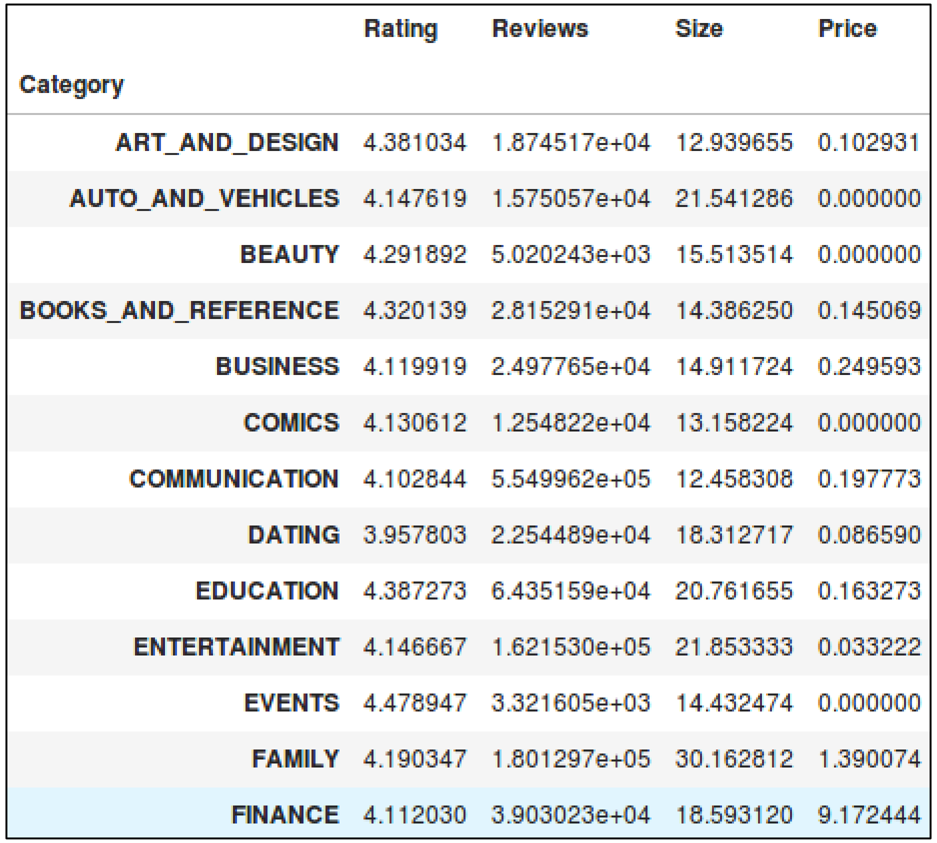
- Read in the data file as csv
- Remove the $-sign from “price” column and re-define as float.
- Use custom made function to convert the “size” column to numbers.
- Remove all rows with missing values

```
def fix_size(s):
    if s.endswith('M'):
        return float(s[:-1])
    if s.endswith('k'):
        return float(s[:-1])/1000
    if s.endswith('G'):
        return float(s[:-1])*1000
    return None
```

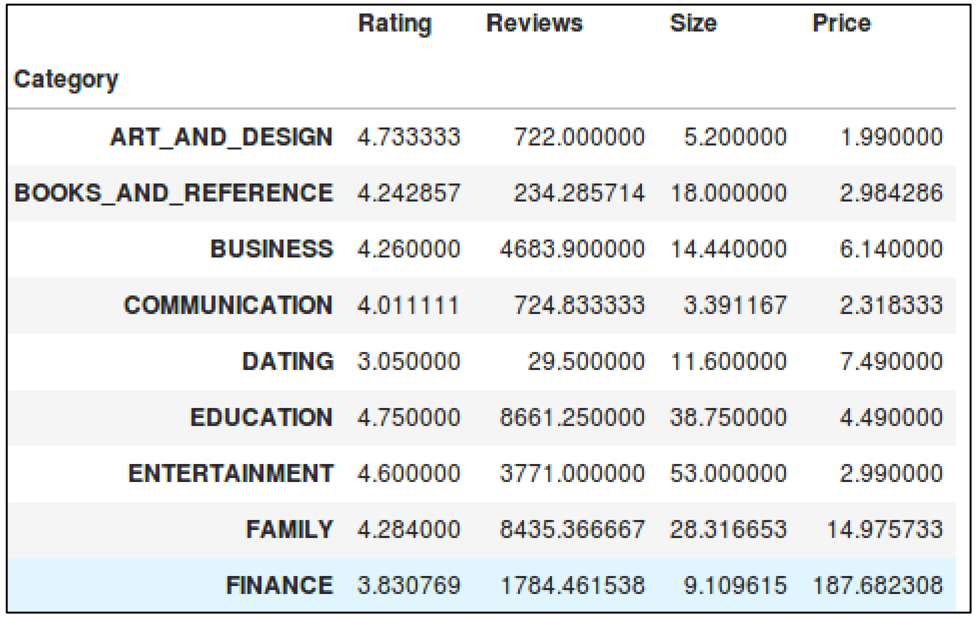
Group by categories

Perform operations within categories:

```
>>> df.groupby(‘Category’).mean()
>>> df[df['Type']=='Paid'].groupby('Category').mean()
>>> df[df['Type']=='Paid'].groupby('Category')[‘Price’].mean()
```



Category	Rating	Reviews	Size	Price
ART_AND_DESIGN	4.381034	1.874517e+04	12.939655	0.102931
AUTO_AND_VEHICLES	4.147619	1.575057e+04	21.541286	0.000000
BEAUTY	4.291892	5.020243e+03	15.513514	0.000000
BOOKS_AND_REFERENCE	4.320139	2.815291e+04	14.386250	0.145069
BUSINESS	4.119919	2.497765e+04	14.911724	0.249593
COMICS	4.130612	1.254822e+04	13.158224	0.000000
COMMUNICATION	4.102844	5.549962e+05	12.458308	0.197773
DATING	3.957803	2.254489e+04	18.312717	0.086590
EDUCATION	4.387273	6.435159e+04	20.761655	0.163273
ENTERTAINMENT	4.146667	1.621530e+05	21.853333	0.033222
EVENTS	4.478947	3.321605e+03	14.432474	0.000000
FAMILY	4.190347	1.801297e+05	30.162812	1.390074
FINANCE	4.112030	3.903023e+04	18.593120	9.172444



Category	Rating	Reviews	Size	Price
ART_AND_DESIGN	4.733333	722.000000	5.200000	1.990000
BOOKS_AND_REFERENCE	4.242857	234.285714	18.000000	2.984286
BUSINESS	4.260000	4683.900000	14.440000	6.140000
COMMUNICATION	4.011111	724.833333	3.391167	2.318333
DATING	3.050000	29.500000	11.600000	7.490000
EDUCATION	4.750000	8661.250000	38.750000	4.490000
ENTERTAINMENT	4.600000	3771.000000	53.000000	2.990000
FAMILY	4.284000	8435.366667	28.316653	14.975733
FINANCE	3.830769	1784.461538	9.109615	187.682308

Operations:

- mean
- sum
- size
- describe
- median
- min
- max
- std
- corr

Sort columns

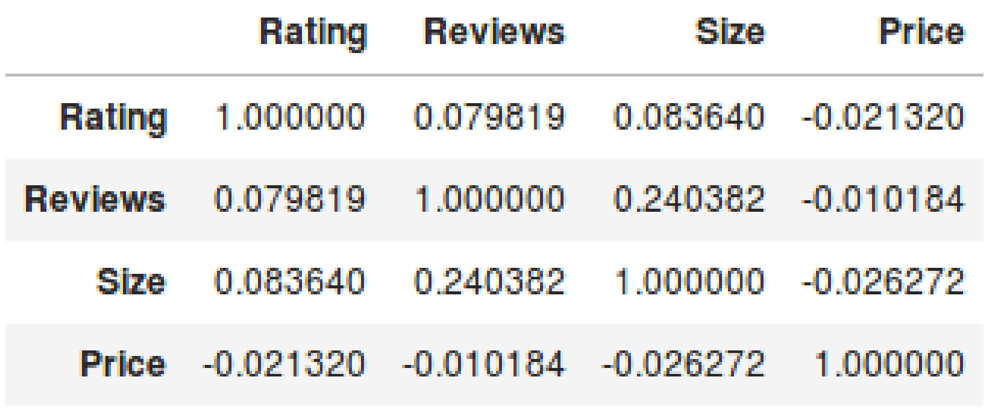
```
>>> df.sort_values(‘Rating’)
>>> df.sort_values(‘Rating’, ascending=False)
>>> df.sort_values([‘Rating’,’Reviews’], ascending=[False, False])
```

Statistics

Correlations

- Calculate correlations between all numerical columns

```
>>> df.corr()
```



	Rating	Reviews	Size	Price
Rating	1.000000	0.079819	0.083640	-0.021320
Reviews	0.079819	1.000000	0.240382	-0.010184
Size	0.083640	0.240382	1.000000	-0.026272
Price	-0.021320	-0.010184	-0.026272	1.000000

Statistics

- Is paid gaming apps more popular than free?
  - Select only gaming apps:

```
>>> df[df['Category']=='GAME']
```

  - Compare Paid vs. Free:

```
>>> df[df['Category']=='GAME'].groupby('Type')
```

  - Define popular as high rating

```
>>> df[df['Category']=='GAME'].groupby('Type')['Rating']
```

  - Calculate the mean and standard deviation:

```
>>> df[df['Category']=='GAME'].groupby('Type')['Rating'].mean()
>>> df[df['Category']=='GAME'].groupby('Type')['Rating'].std()
```

Type	mean Rating	std Rating
Free	4.261513	0.377762
Paid	4.365333	0.358485

T-test in Python

```
>>> from scipy import stats
>>> free = df[(df['Category']=='GAME') & (df['Type']=='Free')]
>>> paid = df[(df['Category']=='GAME') & (df['Type']=='Paid')]
>>> stats.ttest_ind(free['Rating'], paid['Rating'])
Ttest_indResult(statistic=-2.295336499844753, pvalue=0.021926078903232442)
```

Plotting

Plotting setup

- To display plots automatically in Jupyter notebook, execute the following command in a cell

```
>>> %matplotlib inline
```

- Alternatively import matplotlib.pyplot

```
>>> import matplotlib.pyplot as plt
```

- and then display the plots with

```
>>> plt.show()
```

- The last option is how to do it when plotting from Python programs.

Plotting

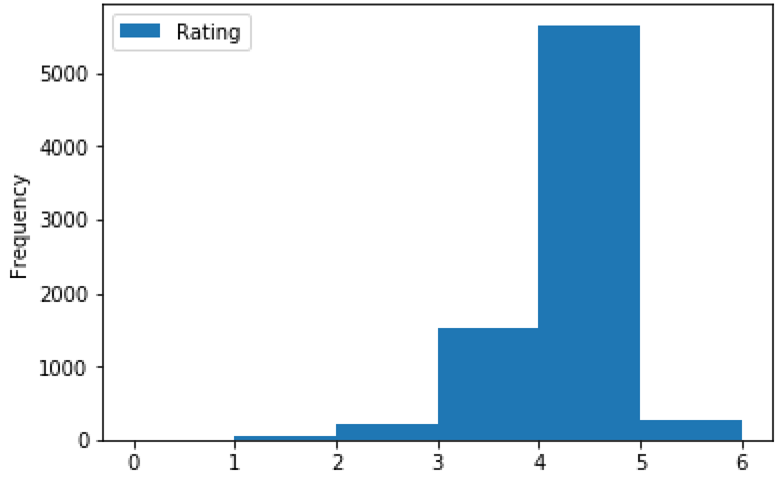
- Dataframes have a method plot that contains several plotting options
  - df.plot.<plot type>
  - line, bar, barh, hist, box, kde, density, area, pie, scatter, hexbin
- Commonly used options
  - x : label or position of column to use as x-values
  - y : label or position of column to use as y-values
  - figsize : Size of plot in inches (width, height)
- Extra options for scatter plots:
  - c : label or position for column to assign colors to points
  - colorbar: adds a colorbar to the plot (True/False)

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html

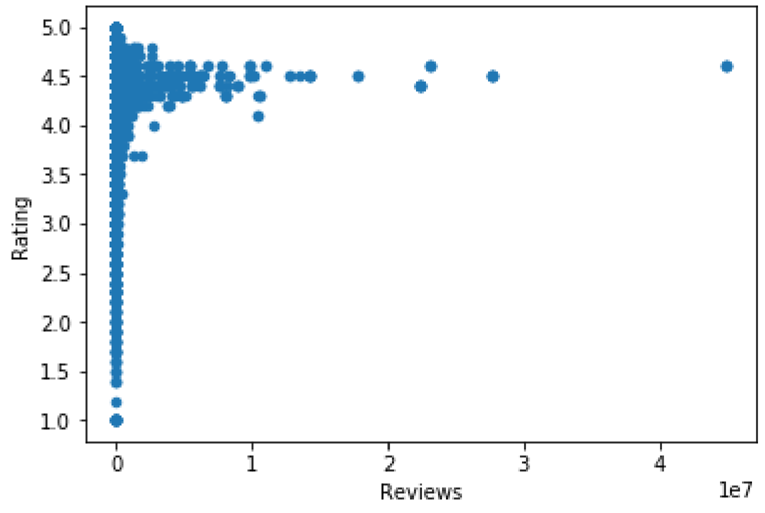
Histograms

Optional, can also be a single number indicating total of evenly spaced bins.

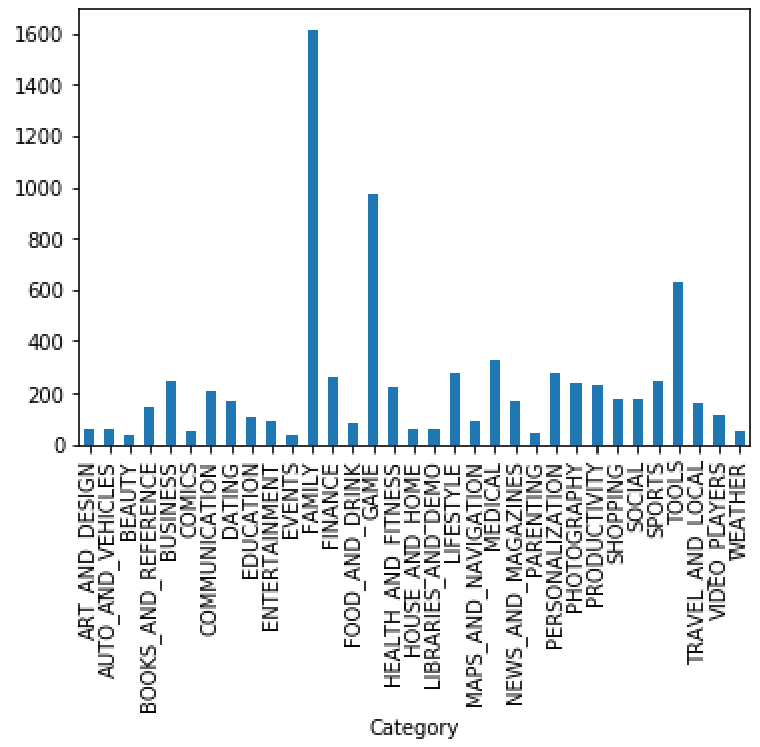
```
>>> df.plot.hist(y='Rating', bins=[0,1,2,3,4,5,6])
```



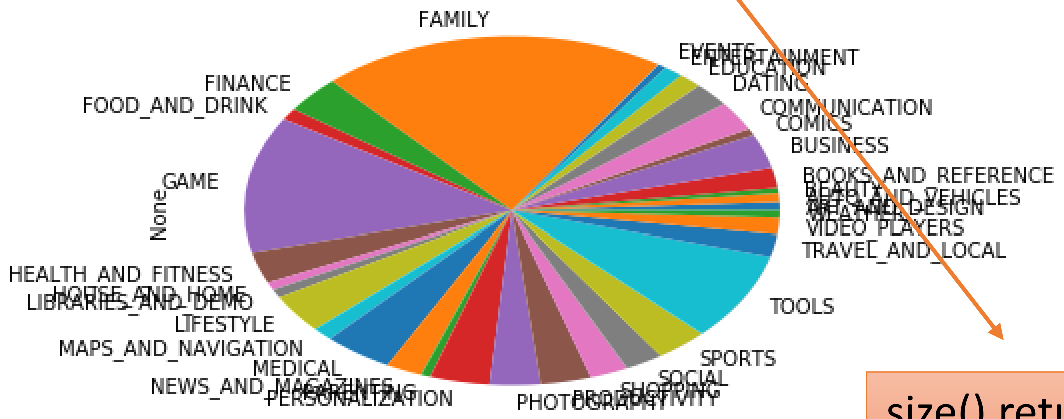
```
>>> df.plot.scatter(x='Reviews', y='Rating')
```



```
>>> df.groupby('Category').size().plot.bar()
```



```
>>> df.groupby('Category').size().plot.pie()
```



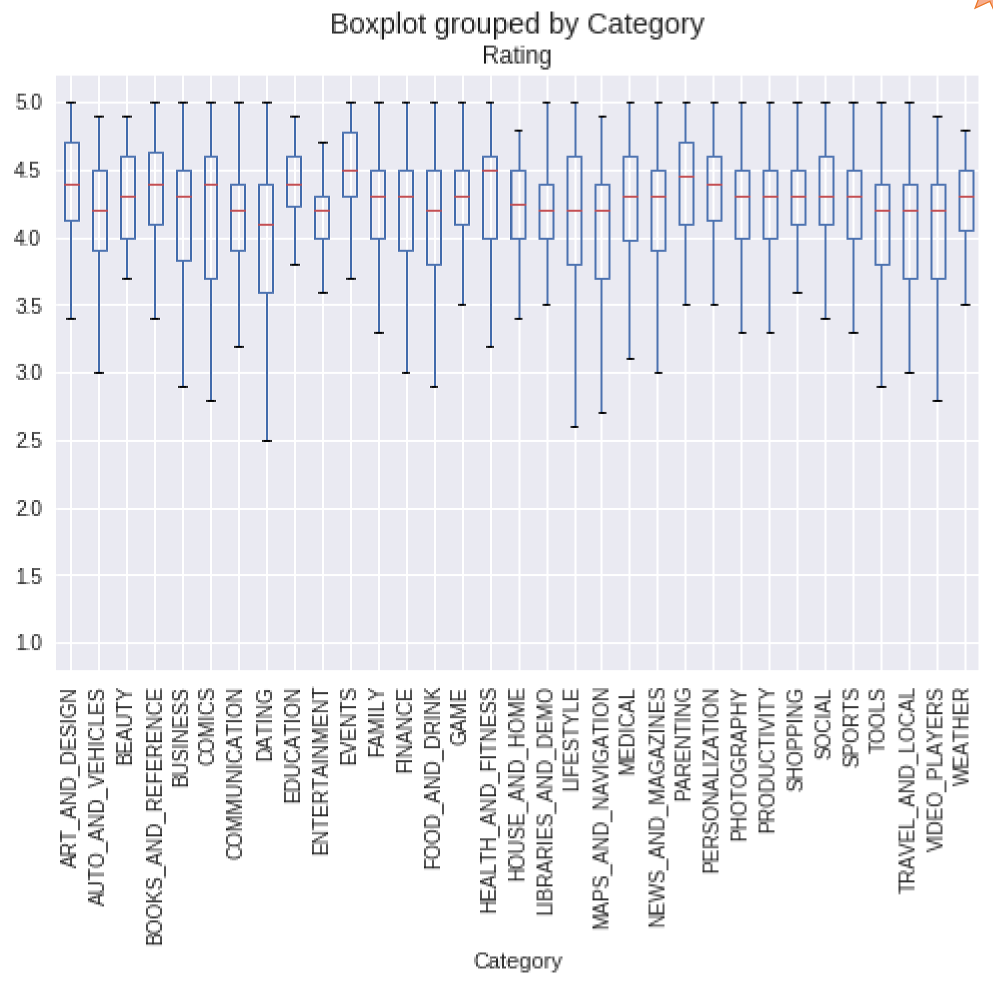
size() returns a pandas Series object, how can we tell?

Boxplots work (slightly) differently

```
>>> df.boxplot(column='Rating', by='Category', rot=90)
```

Note different method!

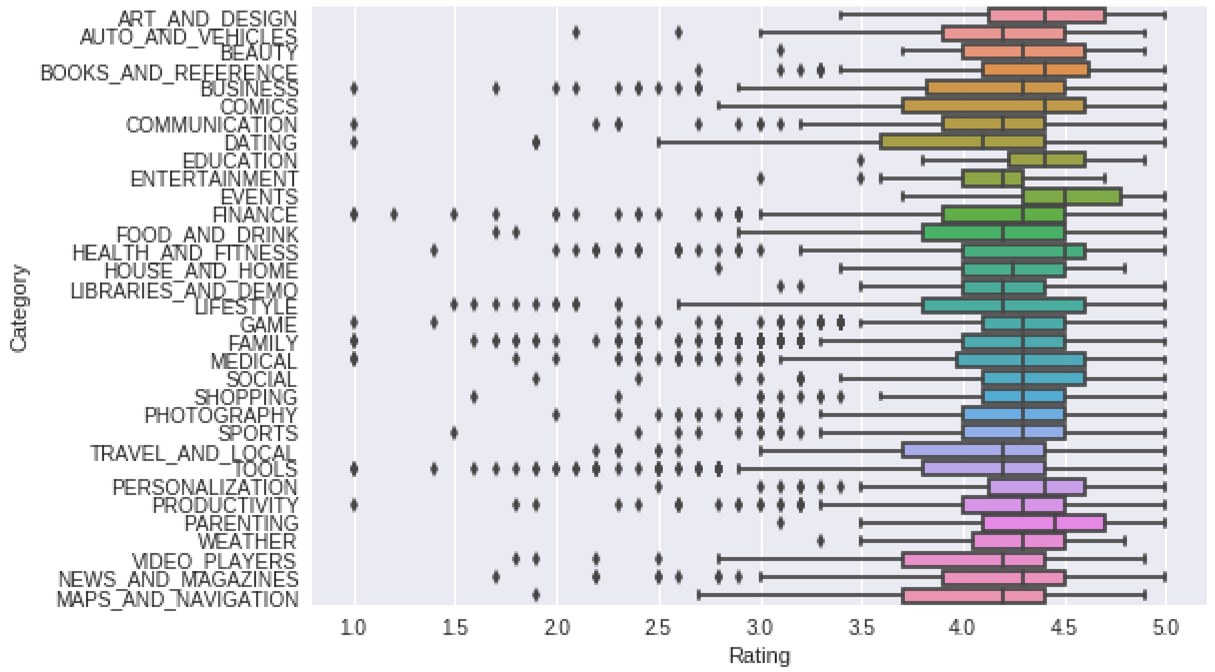
Sets the orientation of the x-labels

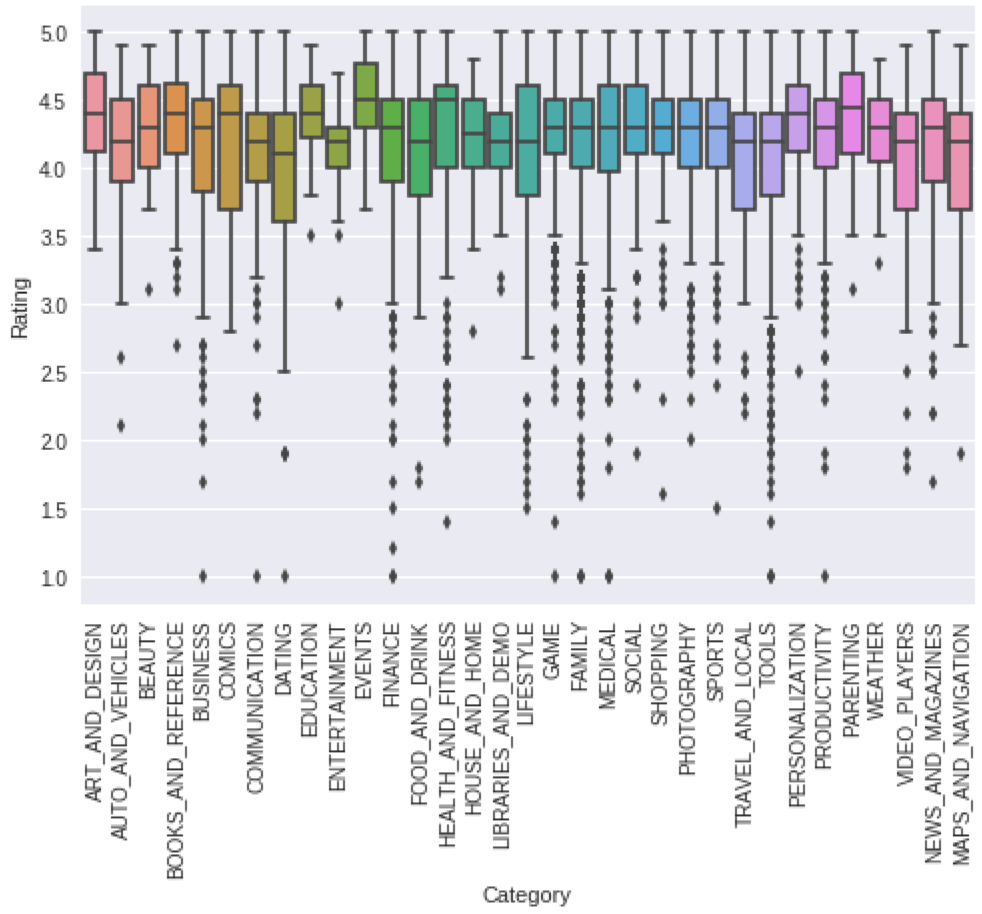


Introducing Seaborn

```
>>> import seaborn as sns
>>> ax = sns.boxplot(x='Category', y='Rating', data=df)
>>> ax.set_xticklabels(ax.get_xticklabels(),rotation=90)
```

```
>>> import seaborn as sns
>>> sns.boxplot(x=‘Rating’, y=‘Category’, orient=‘h, data=df)
```


ART_AND_DESIGN
AUTO_AND_VEHICLES
BEAUTY
BOOKS_AND_REFERENCE
BUSINESS
COMICS
COMMUNICATION
DATING
EDUCATION
ENTERTAINMENT
EVENTS
FINANCE
FOOD_AND_DRINK
HEALTH_AND_FITNESS
HOUSE_AND_HOME
LIBRARIES_AND_DEMO
LIFESTYLE
GAME
FAMILY
MEDICAL
SOCIAL
SHOPPING
PHOTOGRAPHY
SPORTS
TRAVEL_AND_LOCAL
TOOLS
PERSONALIZATION
PRODUCTIVITY
PARENTING
WEATHER
VIDEO_PLAYERS
NEWS_AND_MAGAZINES
MAPS_AND_NAVIGATION
Category
Rating


Rating
ART_AND_DESIGN
AUTO_AND_VEHICLES
BEAUTY
BOOKS_AND_REFERENCE
BUSINESS
COMICS
COMMUNICATION
DATING
EDUCATION
ENTERTAINMENT
EVENTS
FINANCE
FOOD_AND_DRINK
HEALTH_AND_FITNESS
HOUSE_AND_HOME
LIBRARIES_AND_DEMO
LIFESTYLE
GAME
FAMILY
MEDICAL
SOCIAL
SHOPPING
PHOTOGRAPHY
SPORTS
TRAVEL_AND_LOCAL
TOOLS
PERSONALIZATION
PRODUCTIVITY
PARENTING
WEATHER
VIDEO_PLAYERS
NEWS_AND_MAGAZINES
MAPS_AND_NAVIGATION
Category