OMIS 30: Intro to Programming (with Python)

Week 7, Class 2

Introduction to Programming

Instructor: Denis Vrdoljak

Learn programming with Python

Office Hours

Instructor	Days available
Yuan Wang (our TA)	M 2:30-3:30p, W 9-10a
Mike Davis (other section's instructor)	Tu 9:30-10:20a, Th 12-1p
Denis Vrdoljak	Tu 3:40-4:20p, W 5-6p

Course Topics

1. ~~Computer setup - intros~~
2. ~~Shell - cd, mkdir, move, rename, copy, pwd, touch, echo, nano, vim~~
3. ~~Grep, bash, scripts~~
4. ~~Python Basics - print, input, math, PEMDAS~~
5. ~~Pseudo-code~~, algorithm design, ~~comments~~
6. ~~Loops~~ & ~~Nested Loops~~
7. Moving files around, ~~input~~
8. Dates, times, epoch, time-series
9. Arrays, ~~lists~~, dicts, ~~sets~~, etc.
10. ~~And, or, If, elif~~, try, except
11. ~~Functions~~
12. ~~Strings~~, ~~upper, lower~~, regex
13. Computation time, flops, sorting
14. JSON, dicts, iteritems
15. **Pandas**
16. Jupyter, virtual environments
17. Web-apps/web-pages
18. Web-scraping
19. **Plotting, graphing**
20. Git

Goals from last week

- Intro to Classes
- Understand Object Oriented Programming
- Intro to Pandas

By the end of this week you should:

- Advanced Pandas
- More Plotting & Graphing

Review - What is Pandas?

A Python library for data exploration, manipulation, and analysis.

- Good for data sets that are small enough to fit into the local computer memory (~5-10gb)
- Provides an ‘easy’ to use interface to interact with the data

Dataset

For the examples we will use a sample dataset from the link below:

https://data.cityofnewyork.us/api/views/kku6-nxdu/rows.csv?accessType=DOWNLOAD

Review - Sorting by a column in Pandas?

- How do we sort by the column below:
  - Column name: **PERCENT RECEIVES PUBLIC ASSISTANCE**

Review - Sorting by a column in Pandas?

- How do we sort by the column below:
  - Column name: **PERCENT RECEIVES PUBLIC ASSISTANCE**
- df.sort_values(['PERCENT RECEIVES PUBLIC ASSISTANCE'], ascending = False)

```
df.sort_values(['PERCENT RECEIVES PUBLIC ASSISTANCE'], ascending = False)
```

COUNT CITIZEN STATUS UNKNOWN	PERCENT CITIZEN STATUS UNKNOWN	COUNT CITIZEN STATUS TOTAL	PERCENT CITIZEN STATUS TOTAL	COUNT RECEIVES PUBLIC ASSISTANCE	PERCENT RECEIVES PUBLIC ASSISTANCE	COUNT NRECEIVES PUBLIC ASSISTANCE	PERCENT NRECEIVES PUBLIC ASSISTANCE
0	0	3	100	3	1.00	0	0.00
0	0	1	100	1	1.00	0	0.00
0	0	2	100	2	1.00	0	0.00
0	0	2	100	2	1.00	0	0.00
0	0	44	100	35	0.80	9	0.20
0	0	11	100	8	0.73	3	0.27
0	0	11	100	8	0.73	3	0.27
0	0	7	100	5	0.71	2	0.29
0	0	13	100	9	0.69	4	0.31

Review - Multiply a column by 100

- How do we multiply a column PERCENT US CITIZEN by 100 and store it in a new column names US_CITIZEN_%?

Review - Multiply a column by 100

- How do we multiply a column PERCENT US CITIZEN by 100 and store it in a new column names US_CITIZEN_%?
- **df['US_CITIZEN_%'] = df['PERCENT US CITIZEN'] * 100**

```
df['US_CITIZEN_%'] = df['PERCENT US CITIZEN'] * 100
df.head()
```

ERCENT CITIZEN STATUS NKNOWN	COUNT CITIZEN STATUS TOTAL	PERCENT CITIZEN STATUS TOTAL	COUNT RECEIVES PUBLIC ASSISTANCE	PERCENT RECEIVES PUBLIC ASSISTANCE	COUNT NRECEIVES PUBLIC ASSISTANCE	PERCENT NRECEIVES PUBLIC ASSISTANCE	COUNT PUBLIC ASSISTANCE UNKNOWN	PERCENT PUBLIC ASSISTANCE UNKNOWN	COUNT PUBLIC ASSISTANCE TOTAL	PERCENT PUBLIC ASSISTANCE TOTAL	US_CITIZEN_%
0	44	100	20	0.45	24	0.55	0	0	44	100	95.0
0	35	100	2	0.06	33	0.94	0	0	35	100	94.0
0	1	100	0	0.00	1	1.00	0	0	1	100	100.0
0	0	0	0	0.00	0	0.00	0	0	0	0	0.0
0	2	100	0	0.00	2	1.00	0	0	2	100	50.0

Pandas Joining & Grouping Functionality

- Groupby
- Joins

Pandas Groupby

- Useful to group the data by values in one column
  - df.groupby(col_name)
  - Can add an operator on the end like:
    - .count(), .sum(), .mean()
    - df.groupby(col_name).sum()

Pandas Groupby

- For example:
  - df.groupby(['PERCENT US CITIZEN']).sum()

```
df.groupby(['PERCENT US CITIZEN']).sum()
```

PERCENT US CITIZEN	JURISDICTION NAME	COUNT PARTICIPANTS	COUNT FEMALE
0.00	1462225	2	1
0.50	21432	4	2
0.67	20480	6	4
0.71	21695	14	8
0.75	11434	4	1
0.80	21837	15	11
0.86	20480	21	11
0.89	20492	133	82

Pandas Joins

- Add two dataframes together on rows or columns:
  - df1.append(df2)
    - Adds the **rows** of df2 to the end of df1
    - Useful when you have additional ‘new’ rows to add/update the initial dataframe
    - Column names need to be the same
  - pd.concat([df1, df2], axis = 1)
    - Adds df2 **columns** to the right of df1
    - Useful when the two dataframes are the same number of rows and sorted in the same order
    - If any columns are the same will create two versions of that column

Test Datasets

df1

	ID	Amount
0	1	100
1	2	300
2	3	150
3	4	200
4	5	500
5	6	1000

df2 2

	ID	Amount
0	7	400
1	8	123

df1_1

	ID	State
0	1	""
1	2	NY
2	3	CA
3	4	""
4	5	""
5	6	CO

df2

	ID	State
0	2	NY
1	3	CA
2	6	CO
3	7	OR

Pandas Join Examples

df1.append(df1_1)

```
df1.append(df1_1)
```

	ID	Amount
0	1	100
1	2	300
2	3	150
3	4	200
4	5	500
5	6	1000
0	7	400
1	8	123

df1

df1_1

pd.concat([df1, df2], axis = 1)

```
pd.concat([df1,df2], axis=1)
```

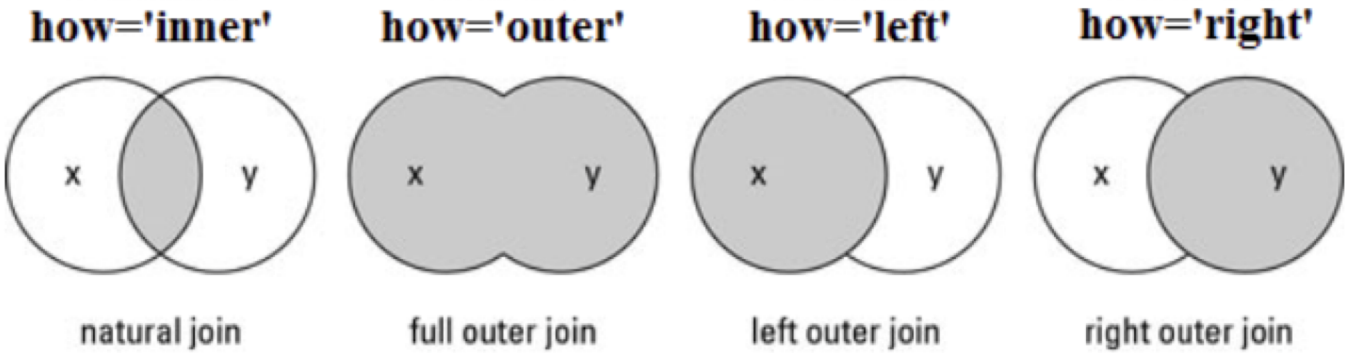
	ID	Amount	ID	State
0	1	100	1	""
1	2	300	2	NY
2	3	150	3	CA
3	4	200	4	""
4	5	500	5	""
5	6	1000	6	CO

df1

df2

Pandas Join Types

- pd.merge(x, y, on = “column_name”, how = <see below>)
- Types of joins in Pandas:



Pandas Inner/Outer Join Examples

pd.merge(df1, df2_2, on='ID', how = 'outer')

```
pd.merge(df1, df2_2, on='ID', how = 'outer')
```

	ID	Amount	State
0	1	100.0	NaN
1	2	300.0	NY
2	3	150.0	CA
3	4	200.0	NaN
4	5	500.0	NaN
5	6	1000.0	CO
6	7	NaN	OR

pd.merge(df1, df2_2, on='ID', how = 'inner')

```
pd.merge(df1, df2_2, on='ID', how = 'inner')
```

	ID	Amount	State
0	2	300	NY
1	3	150	CA
2	6	1000	CO

Pandas Left/Right Join Examples

pd.merge(df1, df2_2, on='ID', how = 'left')

```
pd.merge(df1, df2_2, on='ID', how = 'left')
```

	ID	Amount	State
0	1	100	NaN
1	2	300	NY
2	3	150	CA
3	4	200	NaN
4	5	500	NaN
5	6	1000	CO

pd.merge(df1, df2_2, on='ID', how = 'right')

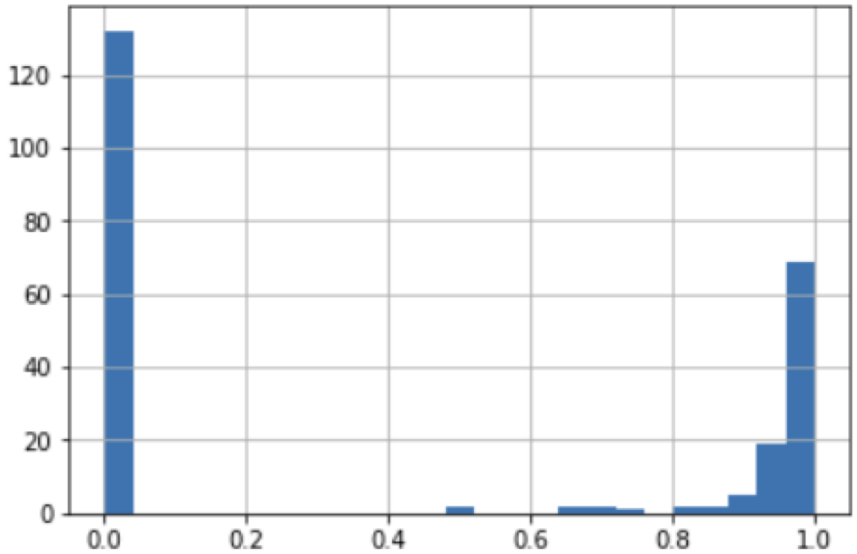
```
pd.merge(df1, df2_2, on='ID', how = 'right')
```

	ID	Amount	State
0	2	300.0	NY
1	3	150.0	CA
2	6	1000.0	CO
3	7	NaN	OR

Pandas Chart Review + Options

```
df['PERCENT US CITIZEN'].hist(bins = 25)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xb78e7f0>
```



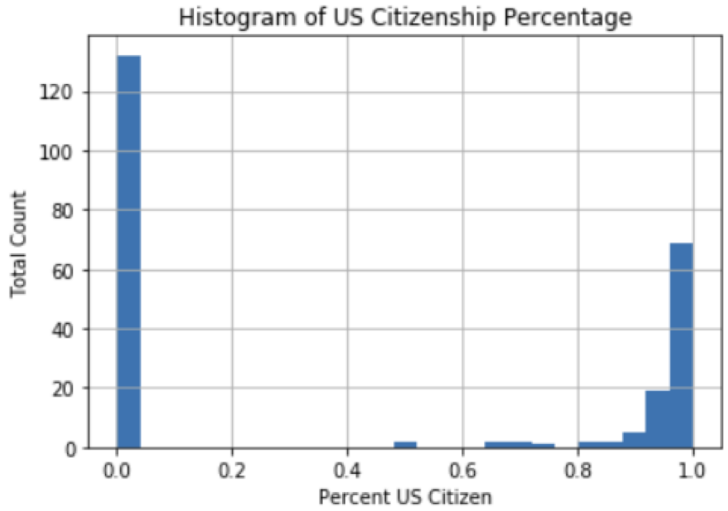
Pandas Chart - How to set the axis & title labels?

- A little complicated: .hist() call returns a graph object
- Pandas uses a library called **matplotlib** to plot the charts
- “import matplotlib.pyplot as plt”
- Need to see the axis labels and title to what you want like:
  - ax.set(xlabel='x label text', ylabel='y label text', title = "title text")
- Finally plt.plot() prints the new chart

Pandas Chart - How to set the axis & title labels?

```
ax = df['PERCENT US CITIZEN'].hist(bins = 25)
ax.set(xlabel='Percent US Citizen', ylabel='Total Count', title = "Histogram of US Citizenship Percentage")
plt.plot()
```

```
[]
```



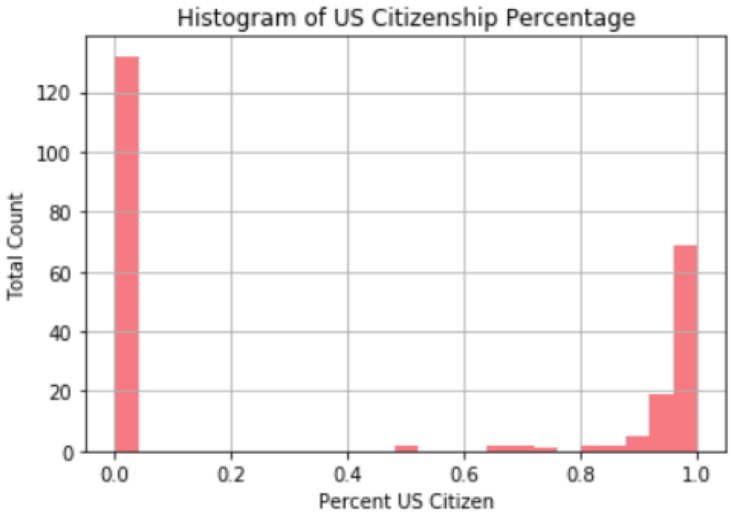
Pandas Chart Coloring Options

- How to set chart color and shade:
  - color = ‘color_type’ -> ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'w']
  - alpha = < 0 - 1.0 >
- Can add these options into the .hist() call like:
  - df[‘col_name’].hist(bins = 25, color = ‘b’, alpha = 0.5)

Pandas Coloring Options Example

```
ax = df['PERCENT US CITIZEN'].hist(bins = 25, color = 'r', alpha = 0.5)
ax.set(xlabel='Percent US Citizen', ylabel='Total Count', title = "Histogram of US Citizenship Percentage")
plt.plot()
```

```
[]
```



Pandas Time Series

- Reading in time-series data can be tricky
  - Most times Pandas reads the dates as a type object or string
  - Can use df.info() to check the types

```
ts.head()
```

	Month	Snow	Rain	Sun
0	2018-01	15	0	16
1	2018-02	10	0	18
2	2004-03	3	6	22
3	2004-04	1	5	24
4	2004-05	0	10	21

Notice:
Month column is of type object

```
ts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
Month    5 non-null object
 Snow    5 non-null int64
 Rain    5 non-null int64
 Sun     5 non-null int64
dtypes: int64(3), object(1)
memory usage: 240.0+ bytes
```

Pandas Time Series

- Can use: pd.to_datetime("col_name") to convert to a datetime object
  - Works much better with a designated format:
  - pd.to_datetime("col_name", format = "%Y-%m-%d")
  - Full formatting at: https://docs.python.org/3.6/library/datetime.html#strftime-and-strptime-behavior

```
ts.Month = pd.to_datetime(ts.Month, format="%Y-%m")
```

```
ts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
Month    5 non-null datetime64[ns]
 Snow    5 non-null int64
 Rain    5 non-null int64
 Sun     5 non-null int64
dtypes: datetime64[ns](1), int64(3)
memory usage: 240.0 bytes
```

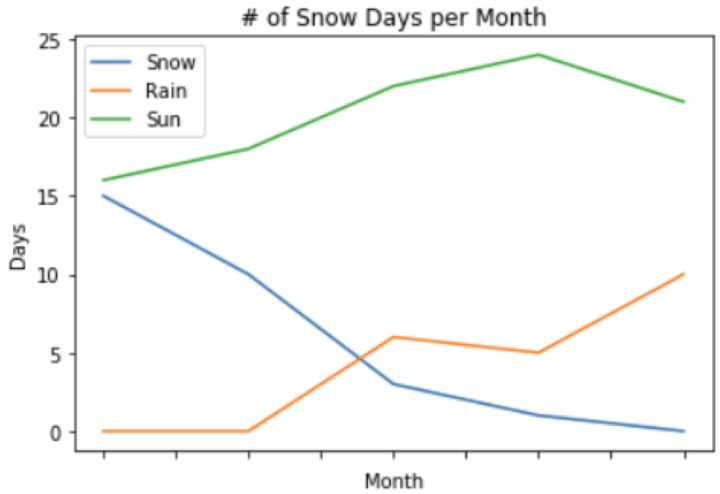
Notice:
Month column is now type 'datetime'

Pandas Time Series Charting Example

Can plot multiple lines on one chart

```
ax = ts.plot(x="Month", y=["Snow", "Rain","Sun"])
ax.set(xlabel='Month', ylabel='Days', title="# of Snow Days per Month")
plt.plot()
```

```
[]
```



What's missing from this chart?

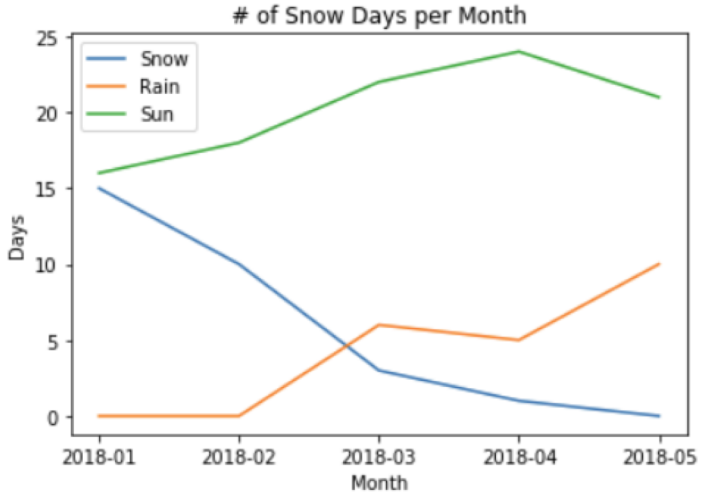
Pandas Time Series Charting Example

X-axis is missing the months!

Need to set the correct range for the months to appear

```
ax = ts.plot(x="Month", y=["Snow", "Rain","Sun"], xticks=range(0, len(ts)))
ax.set(xlabel='Month', ylabel='Days', title="# of Snow Days per Month")
plt.plot()
```

```
[]
```



Homework 7.2

Appendix

- Reserved keywords in Python:
  - https://pentangle.net/python/handbook/node52.html
  - DO NOT USE THESE as VARIABLE NAMES!
- Built-in Functions:
  - https://docs.python.org/3/library/functions.html
  - Review/Reference Material:
  - https://developers.google.com/edu/python/lists
- Dos/Windows vs Unix/Linux:
  - https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Step_by_Step_Guide/ap-doslinux.html
  - You should be familiar with the basic commands for navigating around the file structure and modifying/creating files/folders. You should be aware of the harder to remember ones (like grep and vi) so you know what to Google when you need them!