OMIS 30: Intro to Programming (with Python)

Week 7, Class 1

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, export
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

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

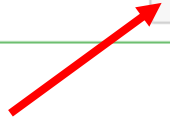
Review - Loading Pandas and the Dataset

In [1]:
```
# Standard Python line to import pandas library
# It's imported as 'pd' for less typing later!

import pandas as pd
```

In [2]:
```
# Reading in the file
# Pandas has many built in readers for different file types

df = pd.read_csv("Demographic_Statistics_By_Zip_Code.csv")
```



- You can name the ‘dataframe’ anything but remember you will type that name many times!
- Common to just name it ‘df’ if you are only loading up one dataset
- Could name the data something like demo_df if you are loading up multiple datasets

Review - How do we see a list of column names?

- df. ????

Review - How do we see a list of column names?

- df.**columns**

```
df.columns
```

```
Index(['JURISDICTION NAME', 'COUNT PARTICIPANTS', 'COUNT FEMALE',
       'PERCENT FEMALE', 'COUNT MALE', 'PERCENT MALE', 'COUNT GENDER UNKNOWN',
       'PERCENT GENDER UNKNOWN', 'COUNT GENDER TOTAL', 'PERCENT GENDER TOTAL',
       'COUNT PACIFIC ISLANDER', 'PERCENT PACIFIC ISLANDER',
       'COUNT HISPANIC LATINO', 'PERCENT HISPANIC LATINO',
       'COUNT AMERICAN INDIAN', 'PERCENT AMERICAN INDIAN',
       'COUNT ASIAN NON HISPANIC', 'PERCENT ASIAN NON HISPANIC',
       'COUNT WHITE NON HISPANIC', 'PERCENT WHITE NON HISPANIC',
       'COUNT BLACK NON HISPANIC', 'PERCENT BLACK NON HISPANIC',
       'COUNT OTHER ETHNICITY', 'PERCENT OTHER ETHNICITY',
       'COUNT ETHNICITY UNKNOWN', 'PERCENT ETHNICITY UNKNOWN',
       'COUNT ETHNICITY TOTAL', 'PERCENT ETHNICITY TOTAL',
       'COUNT PERMANENT RESIDENT ALIEN', 'PERCENT PERMANENT RESIDENT ALIEN',
       'COUNT US CITIZEN', 'PERCENT US CITIZEN', 'COUNT OTHER CITIZEN STATUS',
       'PERCENT OTHER CITIZEN STATUS', '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',
       'COUNT PUBLIC ASSISTANCE UNKNOWN', 'PERCENT PUBLIC ASSISTANCE UNKNOWN',
       'COUNT PUBLIC ASSISTANCE TOTAL', 'PERCENT PUBLIC ASSISTANCE TOTAL'],
      dtype='object')
```

Review - get distinct counts of a column

- How do we get the distinct counts for the column below:
  - Column name: **PERCENT RECEIVES PUBLIC ASSISTANCE**

Review - get distinct counts of a column

- How do we get the distinct counts for the column below:
  - Column name: **PERCENT RECEIVES PUBLIC ASSISTANCE**
- df['PERCENT RECEIVES PUBLIC ASSISTANCE'].value_counts()

```
df['PERCENT RECEIVES PUBLIC ASSISTANCE'].value_counts()
```

```
0.00    150
0.40      7
0.26      4
1.00      4
0.20      4
0.57      4
0.25      3
0.24      3
0.08      3
0.18      3
0.29      3
0.31      2
0.42      2
0.73      2
0.37      2
0.38      2
0.30      2
0.45      2
0.43      2
0.52      2
```

Review - Histogram of a column

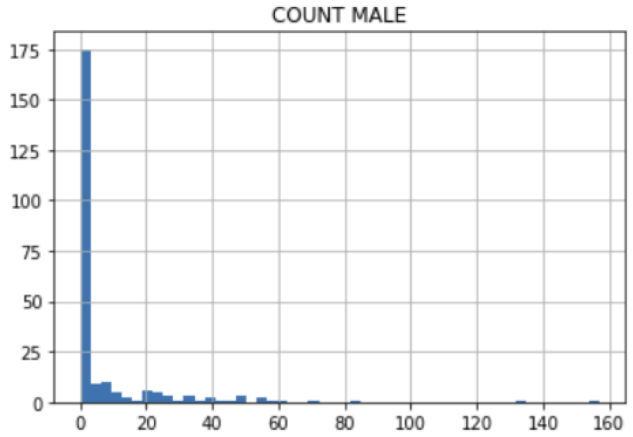
- How do we make a histogram for the column below:
  - Column name: **COUNT MALE**

Review - Histogram of a column

- How do we make a histogram for the column below:
  - Column name: **COUNT MALE**
- df.hist(['COUNT MALE'], bins=50)

```
df.hist(['COUNT MALE'], bins=50)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000000008EEFEF0>]],
      dtype=object)
```



Pandas “Advanced” functionality

- Sorting
- Filtering
- Making new columns
- File I/O

Pandas Sorting with .sort_values()

- df.sort_values(column_name, ascending = True/False)
- To sort by the Jurisdictions with the highest Percent of Hispanic/Latino:
- df.sort_values('PERCENT HISPANIC LATINO', ascending = False)

Pandas Sorting with .sort_values()

```
df.sort_values('PERCENT HISPANIC LATINO', ascending = False)
```

JURISDICTION NAME	COUNT PARTICIPANTS	COUNT FEMALE	PERCENT FEMALE	COUNT MALE	PERCENT MALE	COUNT GENDER UNKNOWN	PERCENT GENDER UNKNOWN	COUNT GENDER TOTAL	PERCENT GENDER TOTAL	COUNT PACIFIC ISLANDER	PERCENT PACIFIC ISLANDER	COUNT HISPANIC LATINO	PERCENT HISPANIC LATINO
11372	2	1	0.50	1	0.50	0	0	2	100	0	0.00	2	1.00
10032	13	4	0.31	9	0.69	0	0	13	100	0	0.00	9	0.69
10465	21	17	0.81	4	0.19	0	0	21	100	0	0.00	14	0.67
10471	43	24	0.56	19	0.44	0	0	43	100	0	0.00	27	0.63
10463	59	33	0.56	26	0.44	0	0	59	100	0	0.00	36	0.61
11226	10	7	0.70	3	0.30	0	0	10	100	0	0.00	6	0.60
10456	5	3	0.60	2	0.40	0	0	5	100	0	0.00	3	0.60
10040	5	4	0.80	1	0.20	0	0	5	100	0	0.00	3	0.60
11368	5	4	0.80	1	0.20	0	0	5	100	0	0.00	3	0.60

Pandas Filtering by boolean expressions

- Can ‘slice’ the dataframe by using a boolean expression:

- Saw df [df[‘col_name’] == <value>] last week

- Can also do any other comparisons like <, >, <=, >=, !=
  - And string together with
  - & = and
  - | = or

Pandas Advanced Filtering

- We want to filter on:
  - % Male above 50% and
  - % Receives Public Assistance equal or below 30%
- df[(df['PERCENT MALE'] > 0.5) & (df['PERCENT RECEIVES PUBLIC ASSISTANCE'] <= 0.3)]
  - Notice the () parenthesis around each slice statement

```
df[ (df['PERCENT MALE'] > 0.5) & (df['PERCENT RECEIVES PUBLIC ASSISTANCE'] <= 0.3) ]
```

	JURISDICTION NAME	COUNT PARTICIPANTS	COUNT FEMALE	PERCENT FEMALE	COUNT MALE	PERCENT MALE	COUNT GENDER UNKNOWN	PERCENT GENDER UNKNOWN	COUNT GENDER TOTAL
5	10006	6	2	0.33	4	0.67	0	0	6
7	10009	2	0	0.00	2	1.00	0	0	2
11	10013	8	1	0.13	7	0.88	0	0	8
52	10306	1	0	0.00	1	1.00	0	0	1
94	11101	1	0	0.00	1	1.00	0	0	1
114	11213	38	17	0.45	21	0.55	0	0	38
124	11223	109	53	0.49	56	0.51	0	0	109
134	11234	59	29	0.49	30	0.51	0	0	59

Pandas Advanced Filtering

- We can do the same thing with an OR filter:
  - % Permanent Resident Alien above 60% or
  - % US citizen below 10%
- df[(df['PERCENT PERMANENT RESIDENT ALIEN'] > 0.6) | (df['PERCENT US CITIZEN'] <= 0.10)]
  - Again - notice the () parenthesis around each slice statement

```
df[ (df['PERCENT PERMANENT RESIDENT ALIEN'] > 0.6) | (df['PERCENT US CITIZEN'] <= 0.10) ]
```

COUNT ETHNICITY TOTAL	PERCENT ETHNICITY TOTAL	COUNT PERMANENT RESIDENT ALIEN	PERCENT PERMANENT RESIDENT ALIEN	COUNT US CITIZEN	PERCENT US CITIZEN	COUNT OTHER CITIZEN STATUS	PERCENT OTHER CITIZEN STATUS	COUNT CITIZEN STATUS UNKNOWN	PERCENT CITIZEN STATUS UNKNOWN
0	0	0	0.0	0	0.0	0	0.0	0	0
0	0	0	0.0	0	0.0	0	0.0	0	0
0	0	0	0.0	0	0.0	0	0.0	0	0
0	0	0	0.0	0	0.0	0	0.0	0	0
0	0	0	0.0	0	0.0	0	0.0	0	0
0	0	0	0.0	0	0.0	0	0.0	0	0
0	0	0	0.0	0	0.0	0	0.0	0	0
0	0	0	0.0	0	0.0	0	0.0	0	0
0	0	0	0.0	0	0.0	0	0.0	0	0

Pandas Making a New Column

- Can make a new column by:
- df[‘new_column_name’] = (value) or (formula)
- Some examples:
  - df[‘static_value’] = 1
  - df[‘log_col’] = df[‘column_name’] / 100
  - df[‘new_column’] = df[‘column1’] + df[‘column2’]

Pandas Making a New Column

- So for these examples:
  - df[‘month_of_survey’] = ‘October’
  - df[‘norm_participants’] = df[‘COUNT PARTICIPANTS’] / 100
  - df[‘AI_PI’] = df[‘COUNT PACIFIC ISLANDER’] + df[‘COUNT AMERICAN INDIAN’]

```
df['month_of_survey'] = 'October'
df['norm_participants'] = df['COUNT PARTICIPANTS'] / 100
df['AI_PI'] = df['COUNT PACIFIC ISLANDER'] + df['COUNT AMERICAN INDIAN']
df.head()
```

TNSL	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	month_of_survey	norm_participants	AI_PI
0	20	0.45	24	0.55	0	0	44	100	October	0.44	0
0	2	0.06	33	0.94	0	0	35	100	October	0.35	0
0	0	0.00	1	1.00	0	0	1	100	October	0.01	0
0	0	0.00	0	0.00	0	0	0	0	October	0.00	0
0	0	0.00	2	1.00	0	0	2	100	October	0.02	0

Pandas File Input

- Pandas can take input from a variety of files - most commonly:
  - CSV / Raw text (table)
  - JSON
  - Excel
  - HTML

- Full List: https://pandas.pydata.org/pandas-docs/stable/io.html

Pandas File Input/Output

- An explanation of parameters:
  - pd.read_csv('file_name', sep = ',', header = integer or True/False, names = column_names)
    - sep = separator - some common ones are a comma(',') , pipe('|') or tab ('\t')
    - header = # of header rows (if passed an integer) or T/F is there is just the first row (or no rows)
    - Names = can pass column names if you want to specify them, otherwise if a header is specified pandas will get the names from the header

Pandas File Output

- Pandas can take output to a variety of files also:
  - CSV / Raw text (table)
  - JSON
  - HTML

- df.to_csv('file_name')
  - Defaults to using the comma as a separator and the column names as the header

Other file input/output

- A non-pandas way to do file input and output is using:
  - **with open**(“file_name”) **as** file:
    - data = file.read()
    - Reads the file in as all text
    - <whatever else you want to do with the data>
  - The **with** open the file for use and closes the file when done
  - The **as** names the file a shorter name

Other file input/output

- Especially useful if the data is not formatted uniformly for pandas to read
  - Also can do if you need to do a line by line comparison (or if certain lines have different information to parse out):
  - **with open**(“file_name”) **as** file:
    - for line in file: <- iterates through each line of the file
      - <do something>

Other file input/output

- To write to a file can do the same thing:
  - with open(‘output_file_name’, ‘w’) as outfile:
    - file.write(‘Testing!’)
  - The **with** open the file for use and closes the file when done
  - The **as** names the file a shorter name
  - The **‘w’** means we want to write to the file

Homework 7.1

- Write a script to read the contents of a file, and write it to another file, but adding “OMIS30: “ at the beginning of every line.
- Write a script that will read in a csv file, add a header row to the beginning which it reads in from a second file, and write the output to a new file.

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!