Python in 10 minutes

Part 7

Dr. Mark Williamson, PhD
Biostatistics, Epidemiology, and Research Design Core (BERDC)
Dakota Cancer Collaborative on Translational Activity (DaCCoTA)
University of North Dakota (UND)

Purpose:

- Quick, bite-size guides to basic usage and tasks in Python
- I’m no expert, I’ve just used it for various tasks, and it has made my life easier and allowed me to do things I couldn’t manually
- I’d like to share that working knowledge with you

Lesson 7: Linking Data

Last time, we learned different ways to edit data, which included managing whitespace, adding content, deleting content, and changing content. Today, we'll try out hand at linking data from different files together. We'll examine how to:

1) link state names to abbreviations using a pre-made dictionary
2) link state names to abbreviations using one file to create a dictionary
3) link state names to abbreviations by looping through one file inside another
4) link two files together by state abbreviation, where not all records are the same
5) link three files together by sequence number, where not all records are the same

Lesson 7: The Datasets in Question

State datasets

US_state_population.csv

Abbr	Population
AL	4779736
AK	710231
AZ	6392017
AR	2915918
CA	37253956
CO	5029196
CT	3574097
DE	897934
FL	18801310
GA	9687653

US_state_abbreviations.csv

Abbr	State
AL	Alabama
AK	Alaska
AZ	Arizona
AR	Arkansas
CA	California
CO	Colorado
CT	Connecticut
DE	Delaware
FL	Florida
GA	Georgia

US_state_brain_cancer.csv*

Abbr	Total_Incidence	Male_Incidence	Female_Incidence
AL	7	7.9	6.1
AK	7.2	7.8	6.3
AZ	6.7	8	5.5
AR	7.6	9.1	6.2
CA	7.7	9.1	6.4
CO	7.1	8.2	6
CT	7.4	8.8	6.2
FL	7.3	8.6	6
GA	7.2	8.3	6.2
HI	7.3	8.6	5.8

*Not all states are included (missing data)

NHANES datasets

- 3 datasets from the NHANES survey 2017-2018
- Demographics (DEMO_test.csv) with sequence id and 25 other variables
- Albumin (ALB_test.csv) with sequence id and 2 other variables
- Physical Activity (PAQ_test.csv) with sequence id and 4 other variables

Lesson 7: Easy Example 1

Goal: Create a new file that has the full state name as well

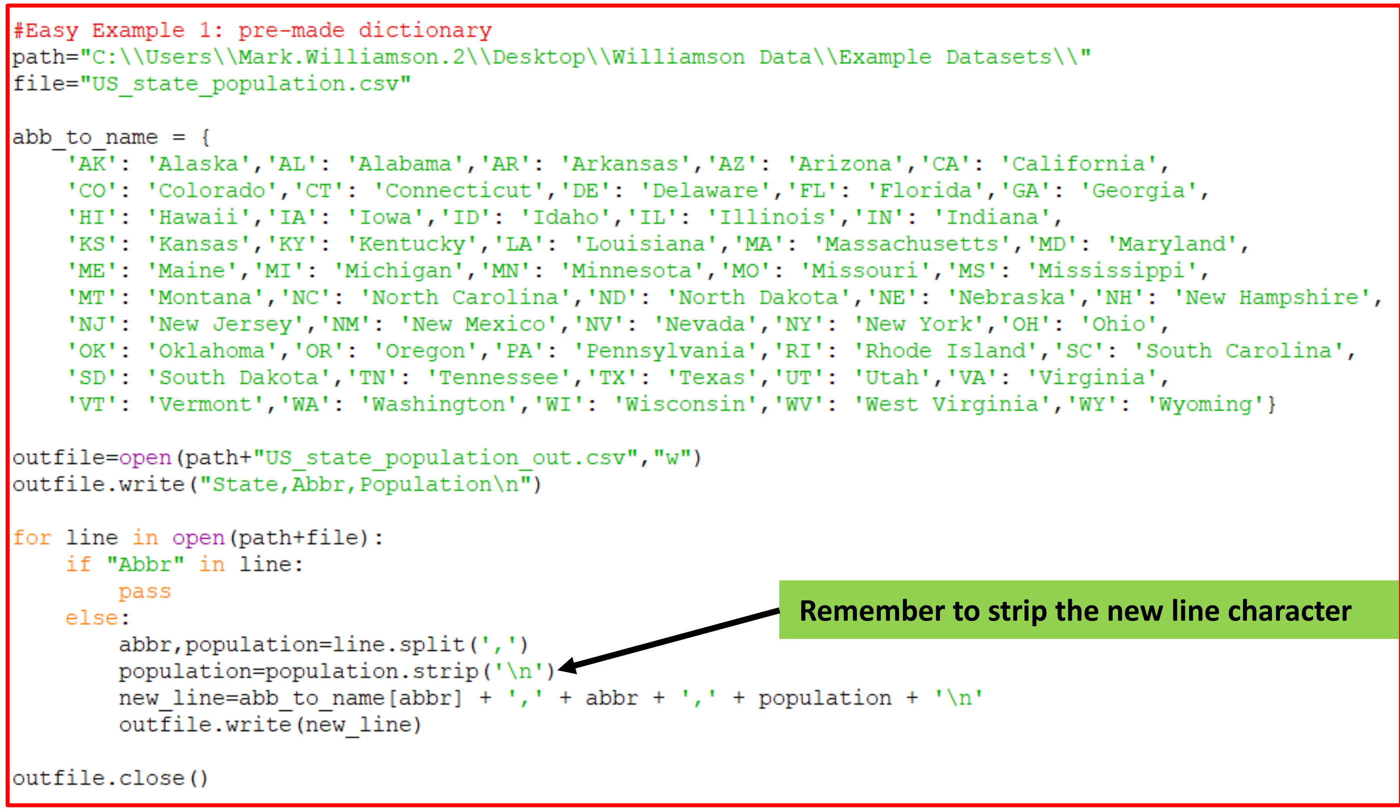
Exploration

Try adding US territory entries (Guam, Puerto Rico, etc.) to the dictionary and find population information to add to the population file

Procedure

- Download the first dataset (US_state_population)
- Open Python and start a new file
- Create a **path** and **file** variable
- Create the **abb_to_name** dictionary (can copy and paste provided text file -> **abb to name dict.txt**)
- Create an **outfile** and write the column names to the file
- Create a for-loop for each line
- Create an if-else statement that checks if "Abbr" is in the line (indicates a header) and passes if true
- Else, split the line into the variables **abbr** and **population**
- Create a **new_line** variable that consists of the full name of the states, the abbreviation, and the population, separated by commas and ending with a new line character
- Write **new_line** to the **outfile** and close the file when done

#abb_to_name[abbr] gives the full state name associated with that abbreviation in the dictionary



```
#Easy Example 1: pre-made dictionary
path="C:\\Users\\Mark.Williamson.2\\Desktop\\Williamson Data\\Example Datasets\\"
file="US_state_population.csv"

abb_to_name = {
    'AK': 'Alaska','AL': 'Alabama','AR': 'Arkansas','AZ': 'Arizona','CA': 'California',
    'CO': 'Colorado','CT': 'Connecticut','DE': 'Delaware','FL': 'Florida','GA': 'Georgia',
    'HI': 'Hawaii','IA': 'Iowa','ID': 'Idaho','IL': 'Illinois','IN': 'Indiana',
    'KS': 'Kansas','KY': 'Kentucky','LA': 'Louisiana','MA': 'Massachusetts','MD': 'Maryland',
    'ME': 'Maine','MI': 'Michigan','MN': 'Minnesota','MO': 'Missouri','MS': 'Mississippi',
    'MT': 'Montana','NC': 'North Carolina','ND': 'North Dakota','NE': 'Nebraska','NH': 'New Hampshire',
    'NJ': 'New Jersey','NM': 'New Mexico','NV': 'Nevada','NY': 'New York','OH': 'Ohio',
    'OK': 'Oklahoma','OR': 'Oregon','PA': 'Pennsylvania','RI': 'Rhode Island','SC': 'South Carolina',
    'SD': 'South Dakota','TN': 'Tennessee','TX': 'Texas','UT': 'Utah','VA': 'Virginia',
    'VT': 'Vermont','WA': 'Washington','WI': 'Wisconsin','WV': 'West Virginia','WY': 'Wyoming'}

outfile=open(path+"US_state_population_out.csv","w")
outfile.write("State,Abbr,Population\n")

for line in open(path+file):
    if "Abbr" in line:
        pass
    else:
        abbr,population=line.split(',')
        population=population.strip('\n')
        new_line=abb_to_name[abbr] + ',' + abbr + ',' + population + '\n'
        outfile.write(new_line)

outfile.close()
```

Remember to strip the new line character

Lesson 7: Easy Example 2

Goal: Create a new file that has the full state name as well

Procedure

- Download the second dataset (US_state_abbreviations.csv)
- Set dataset to the variable **file2**

# Create the dictionary

- Create an empty dictionary called **abb_to_name2**
- Create a for-loop for each line of file2, passing over the headers
- Split the line into **abbr** and **state** variables, then fill the **abb_to_name2** dictionary with them

# Create and fill the output file

- Create **outfile2** and write the column names to the file
- Create a for-loop for each line of file, passing over the headers
- Split the line into the variables **abbr** and **population**
- Create a **new_line** variable that consists of the full name of the states, the abbreviation, and the population, separated by commas and ending with a new line character
- Write **new_line** to **outfile2** and close the file when done

```
#Easy Example 2: two files, using one to create a dictionary
file2="US_state_abbreviations.csv"

abb_to_name2 ={}
for line in open(path+file2):
    if "Abbr" in line:
        pass
    else:
        abbr,state=line.split(',')
        state=state.strip('\n')
        abb_to_name2[abbr]=state

outfile2=open(path+"US_state_population_out2.csv","w")
outfile2.write("State,Abbr,Population\n")

for line in open(path+file):
    if "Abbr" in line:
        pass
    else:
        abbr,population=line.split(',')
        population=population.strip('\n')
        new_line=abb_to_name2[abbr] + ',' + abbr + ',' + population + '\n'
        outfile2.write(new_line)

outfile2.close()
```

Lesson 7: Easy Example 3

Goal: Create a new file that has the full state name as well

Procedure

- Create **outfile3** and write the column names to the file
- Create a for-loop for each line of **file**, passing over the headers
- Split the line into the variables **abbr** and **population**
- Create a variable called **state** and keep it empty
- Create a for-loop inside the else statement that opens up **file2**
- Split **line2** into the variables **abbr2** and **state2**
- Use and if statement to check if **abbr** is equal to **abbr2**
- If true, set **state** as the value from **state2**
- Create a **new_line** variable that consists of the full name of the states, the abbreviation, and the population, separated by commas and ending with a new line character
- Write **new_line** to **outfile3** and close the file when done

Illumination

When creating nested (one inside another) for-loops or if-else statements, make sure to pay careful attention to indentation

```
#Easy Example 3: two files, looping through one file inside the other

outfile3=open(path+"US_state_population_out3.csv","w")
outfile3.write("State,Abbr,Population\n")

for line in open(path+file):
    if "Abbr" in line:
        pass
    else:
        abbr,population=line.split(',')
        population=population.strip('\n')
        state=''
        for line2 in open(path+file2):
            abbr2,state2=line2.split(',')
            if abbr==abbr2:
                state=state2.strip('\n')
        new_line=state + ',' + abbr + ',' + population + '\n'
        outfile3.write(new_line)

outfile3.close()
```

Remember to strip the new line character

Lesson 7: More Complex Example 1

Goal: Create a new file that adds state cancer data and fills in the missing records with N/A

Exploration

Try a different way to indicate missing data rather than 'N/A'. Some ideas would be 'missing', '.', or '_'.

Procedure

- Download the third dataset (US_state_brain_cancer.csv)
- Set dataset to the variable **file3**
- Create **outfile4** and write the column names to the file
- Create a for-loop for each line of **file2**, passing over the headers
- Split the line into the variables **abbr** and **state**
- Create variables called **total, male**, and **female** and set them equal to 'N/A'
- Create a for-loop inside the else statement that opens up **file3**
- Split **line2** into the variables **abbr2**, **total2**, **male2**, and **female2**
- Use and if statement to check if **abbr** is equal to **abbr2**
- If true, set **total** as the value from **total2,** doing the same for **male** and **female**
- Create a **new_line** variable will all variable separated by comma
- Write **new_line** to **outfile4** and close the file when done

```python
#More Complex Example 1: two files, not all records the same

file3="US_state_brain_cancer.csv"

outfile4=open(path+"US_state_brain_cancer_out.csv","w")
outfile4.write("State,Abbr,Total,Male,Female\n")

for line in open(path+file2):
    if "Abbr" in line:
        pass
    else:
        abbr,state=line.split(',')
        state=state.strip('\n')
        total='N/A'
        male='N/A'
        female='N/A'
        for line2 in open(path+file3):
            abbr2,total2,male2,female2=line2.split(',')
            female2=female2.strip('\n')
            if abbr==abbr2:
                total=total2
                male=male2
                female=female2
        new_line=state + ',' + abbr + ',' + total + ',' + male + ',' + female + '\n'
        outfile4.write(new_line)

outfile4.close()
```

If the abbreviation is not found in the brain cancer file, the variables of total, male, and female will remain as 'N/A'

Lesson 7: More Complex Example 2

Remember, for dictionaries, it is Dictionary[KEY]=VALUE

Goal: combine three NHANES datasets by sequence

Procedure

- Download the 4th, 5th, and 6th datasets, set them to variables, create **outfile5,** and create three empty sequence dictionaries
- Create a for-loop for each line of **file4,** writing the headers to **outfile5**
- Split the line into the variable **seqn** (1st entry) and the list **col_names** (next entries)
- Set **seqn** as the key and the **col_names** list as the value for **seqn_dict1**
- Run the same procedure for file5 and file6, except stripping the SEQN before writing the headers to **outfile5**
- Create a for-loop for each sequence in **seqn_dict1**
- Create an empty variable called **out_line** and a list called **list1** and set it equal to the list for the sequence in **seqn_dict1**
- Create if-else statements to check **list2** and **list3** for the sequence and then add the list if present or a list with 'N/A's
- Create a list called **list_list** that is the combination of all three lists
- Create two for-loops, one inside the other, to add each item from each list to the **out_line** followed by a comma
- Strip the last comma from **out_line**, then write the sequence, a comma, and **out_line** to **outfile5**, and finally, close **outfile5** when done

```python
#More Complex Example 2: three larger, messier files (datasets from NHANES)

file4="DEMO_test.csv"
file5="ALB_test.csv"
file6="PAQ_test.csv"

outfile5=open(path+"DEMO_ALB_PAQ_out.csv","w")

seqn_dict1={}
seqn_dict2={}
seqn_dict3={}

for line in open(path+file4):
    if "SEQN," in line:
        outfile5.write(line.strip('\n'))
    else:
        seqn=line.split(',')[0]
        col_names=line.split(',')[1:]
        col_names[-1]=col_names[-1].strip('\n')
        seqn_dict1[seqn]=col_names

for line in open(path+file5):
    if "SEQN," in line:
        line2=line.strip("SEQN")
        outfile5.write(line2.strip('\n'))
    else:
        seqn=line.split(',')[0]
        col_names=line.split(',')[1:]
        col_names[-1]=col_names[-1].strip('\n')
        seqn_dict2[seqn]=col_names

for line in open(path+file6):
    if "SEQN," in line:
        line2=line.strip("SEQN")
        outfile5.write(line2)
    else:
        seqn=line.split(',')[0]
        col_names=line.split(',')[1:]
        col_names[-1]=col_names[-1].strip('\n')
        seqn_dict3[seqn]=col_names
```

Empty Sequence Dictionaries

These steps keep SEQN from being repeated across multiple column headers

```python
for seqn in seqn_dict1:
    out_line=''
    list1=seqn_dict1[seqn]
    if seqn in seqn_dict2:
        list2=seqn_dict2[seqn]
    else:
        list2=['NA','NA']
    if seqn in seqn_dict3:
        list3=seqn_dict3[seqn]
    else:
        list3=['NA','NA','NA','NA']
    list_list=[list1,list2,list3]
    for l in list_list:
        for i in l:
            out_line=out_line + i + ','
    out_line=out_line[0:-1]
    outfile5.write(seqn + ',' + out_line+'\n')

outfile5.close()
```

The 'l' is each list, while the 'i' is each item in the list

Lesson 7: Summary

- Python can link different data together in a variety of ways
- Lines in one file came be compared to lines in another file and then written together into a new file
- Datapoints can be stored in dictionaries, compared to lines in a file, and written together into a new file
- Datapoints can be stored in multiple dictionaries, each one can be compared to another, and written together into a new file
- Please complete a brief, 5-question assessment:

  https://und.qualtrics.com/jfe/form/SV_e2Pr73mRxKv2R9A