Program cs08
Fall 2022: Updated September 23, 2022

CS 5245: Data Programming
R. M. Parry

Read chapter 3 in Python Data Science Handbook and complete the following exercises. In this assignment, you will use the built-in `pandas` package to read CSV files, clean the data, and join different tables.

Complete computer setup by following the directions here:
http://cs.appstate.edu/~rmp/cs5245/setup.pdf

Complete the following activities using the **exact function names** and save the files using the **exact file names** as specified. Test your programs using the command line or an IDE to check their output.

1 Description

In this assignment, you will learn to use the `pandas` package in Python to clean and manipulate tabular data. You will write three different programs. Each will start with a main function like and you will add code to accomplish the various tasks, saving a CSV file as your result. Web-CAT will grade your assignment based on the CSV file it produces.

```
def main():
    print('it works!')

if __name__ == '__main__':
    main()
```

Download the following and put into the directory where you will work on this assignment.

1. basic_person.csv
2. student_detail_v.csv
3. person_detail_f.csv
4. city_name_map.py
5. food.csv

2 flat_file

This program focuses on combining the tables from a database containing middle school academic performance in northwest North Carolina. We will use the basic_person.csv, student_detail_v.csv, and person_detail_f.csv files.

Start with a Python file called 'flat_file.py'. You will need to import the `pandas` package as shown:

```
# flat_file.py
import pandas as pd
```

```
def main():
    # Load and process the files.
    # Save the file 'joined.csv' without the implicit index
    pass

if __name__ == '__main__':
    main()
```

The database separates demographic information from academic information for each student. Specifically, there is a `basic_person` table and a `person_detail` table. The `basic_person` table contains the demographic information (gender, city, state, zip, and ethnicity) and is indexed by an `acct_id`. The following example runs in the IPython shell; however, you will need to write a program in `flat_file.py` to accomplish these tasks. It's a good idea to have the shell and text editor open or use an IDE that supports both.

```
In [1]: import pandas as pd

In [2]: df_person = pd.read_csv('basic_person.csv')

In [3]: # demographic information
   ...: df_person
Out[3]:
       acct_id_new gender           city state    zip ethnicity
0                1      F     Wilkesboro    NC  28697         2
1                2      F      Morganton    NC  28655         0
2                3      F       Spindale    NC  28160         0
3                4      M     Hayesville    NC  28904         0
4                5      F      Ellenboro    NC  28040         0
...            ...    ...            ...   ...    ...       ...
17339        17340      M  Rutherfordton    NC  28139         0
17340        17341      F   Glade Valley    NC  28627         1
17341        17342      M     Burnsville    NC  28714         0
17342        17343      M          Warne    NC  28909         0
17343        17344      F    Forest City    NC  28043         0

[17344 rows x 6 columns]
```

The `person_detail` table contains the academic information (GPA and absences among others) and is indexed by a `person_detail_id`:

```
In [4]: df_student = pd.read_csv('person_detail_f.csv')

In [5]: # academic information per year
   ...: df_student
Out[5]:
       person_detail_id_new  school_id_new  ...  num_f  academic_year_id
0                         1             53  ...      0                12
1                         2             61  ...      1                12
2                         3             51  ...      0                11
3                         4             74  ...      0                12
4                         5             22  ...      4                12
...                     ...            ...  ...    ...               ...
29443                 29444             39  ...      0                11
```

```
29444                29445            40  ...      2               12
29445                29446            11  ...      0               12
29446                29447            12  ...      0               12
29447                29448            55  ...      0               12

[29448 rows x 13 columns]
```

Finally, the `student_detail` table provides the map between `student_id`, `acct_id`, and `person_detail_id`:

```
In [6]: df_map = pd.read_csv('student_detail_v.csv')

In [7]: # map from academic information (person_detail_id) to
   ...: # the corresponding student ID and demographic
   ...: # information (acct_id)
   ...: df_map
Out[7]:
       student_id_new  acct_id_new  person_detail_id_new
0                4115            1                 14124
1                4115            1                 16549
2                2919            2                 10850
3               13063            3                 29368
4                8385            4                 17675
...               ...          ...                   ...
28336            7762        17341                 14895
28337             413        17342                  6175
28338             413        17342                  8385
28339            4498        17343                  5000
28340            8208        17344                  7557

[28341 rows x 3 columns]
```

Each row of the `student_detail` table provides a different `person_detail_id` for a unique row of academic information. The academic information is generated each year so most students have more than one `person_detail_id` associated with their `student_id` and `acct_id`. In addition, some students move, so their demographic information changes. We want the latest demographic information and academic information for the data table we're constructing. Fortunately, the identification numbers have been assigned sequentially. So we only need to get the largest `acct_id` and `person_detail_id` for each student. This can be accomplished with the `DataFrame.groupby` and `GroupBy.max` functions.

Once we have the correct mapping for each student, we want to construct the data table that contains one row per student along with their latest demographic and academic information.

Requirements: Add your code to create the data file from the three CSVs in your main function in `flat_file.py`:

1. Create a frame with one row per student, containing their largest `acct_id` and `person_id`
2. Pull in the data from the person table based on `person_id`
3. Pull in the data from the basic table based on `acct_id`

When completed, you should have a table with exactly 16262 rows and 20 columns. The columns should include `'student_id_new'`, `'acct_id_new'`, and `'person_detail_id_new'` along with the

demographic and academic information. Save the file as `'joined.csv'` without the implicit index. So, if you load the file again using `pd.read_csv` it will contain exactly 20 columns again. The Web-CAT test will check the number of columns, column names, number of rows, and the data in the table. The order of the rows and columns does not matter.

The resulting `'joined.csv'` file should look something like this, although the rows and columns can be in any order. Here I'm just showing the first 5 columns.

```
In [1]: import pandas as pd
   ...: df = pd.read_csv('joined.csv')
   ...: df.iloc[:, :5].head(n=20)
Out[1]:
    person_detail_id_new  acct_id_new  student_id_new gender             city
0                   3221        15485               1      F          Spindale
1                  13118        17185               2      F           Creston
2                  13067        10165               3      M     Rutherfordton
3                  28185        13890               4      M         Ellenboro
4                  12510        11313               5      M       Forest City
5                   6310         9161               6      F    Moravian Falls
6                  12112        16969               7      M         Morganton
7                   6157        17012               8      M     Rutherfordton
8                  15921         1339               9      F  North Wilkesboro
9                  27837        12639              10      M        Burnsville
10                  8897         2164              11      M     Rutherfordton
11                 17994         6698              12      M           Valdese
12                 24757         4559              13      F         Jefferson
13                 26821        14674              14      F          Marshall
14                 27626         7572              15      M          Caroleen
15                 25603         7423              16      M        Wilkesboro
16                  7578        13575              17      F           Valdese
17                 23832         9243              18      F      N Wilkesboro
18                 24830         8413              19      M               NaN
19                 13615        10729              20      F         Ellenboro
```

Useful functions:

- `DataFrame.groupby`
- `DataFrame.join`
- `DataFrame.merge`
- `DataFrame.reset_index`
- `DataFrame.set_index`
- `DataFrame.to_csv`
- `GroupBy.max`
- `pandas.merge`
- `pandas.read_csv`

3 cleaned

For this part we will clean the city, state, and zip code information in the demographics file. Remember, the joined file only contains some of the demographic information, so if we want a thorough cleaning we'll use the basic_person.csv file. No need to replace the values in 'joined.csv' from the previous section; Web-CAT will detect these changes as errors.

Start with a Python file called 'clean.py'. You will need to import the pandas, string, and possibly the regular expression re package as shown:

```
# clean.py
import pandas as pd
import string
import re

def main():
    # Load and process the 'basic_person.csv' file.
    # Save the file 'cleaned.csv' without the implicit index
    pass

if __name__ == '__main__':
    main()
```

1. Read the basic_person.csv file into a DataFrame. It should look like this:

```
In [2]: df
Out[2]:
       acct_id_new gender           city state    zip ethnicity
0                1      F     Wilkesboro    NC  28697         2
1                2      F      Morganton    NC  28655         0
2                3      F       Spindale    NC  28160         0
3                4      M     Hayesville    NC  28904         0
4                5      F      Ellenboro    NC  28040         0
...            ...    ...            ...   ...    ...       ...
17339        17340      M  Rutherfordton    NC  28139         0
17340        17341      F   Glade Valley    NC  28627         1
17341        17342      M     Burnsville    NC  28714         0
17342        17343      M          Warne    NC  28909         0
17343        17344      F    Forest City    NC  28043         0

[17344 rows x 6 columns]
```

2. Now suppose we want to know how many students live in each city. Use the Series.value_counts function to produce the following Series. To see all the elements, we'll need to change the maximum number of rows that pandas will show:

```
In [4]: pd.set_option('display.max_rows', 500)
   ...: c = df['city'].value_counts()
   ...: print(c)
   ...: print(len(c), 'cities')

Morganton                2216        North Wilkesboro          937
Forest City              1522        Wilkesboro                620
Rutherfordton            1092        Connelly Springs          561
```

Ellenboro	528
Marshall	444
Valdese	381
Hickory	359
Robbinsville	354
Hayesville	344
Bostic	341
Bryson City	334
Millers Creek	330
Spindale	317
Mooresboro	316
Burnsville	291
West Jefferson	282
Boone	281
Sparta	272
Mars Hill	242
Jefferson	191
Hays	179
Lansing	147
Elkin	140
Roaring River	130
Moravian Falls	129
Purlear	127
Union Mills	126
Ronda	118
Vilas	113
Cherokee	113
Traphill	91
Hildebran	85
Hot Springs	84
Crumpler	79
Creston	78
Ennice	77
Boomer	75
Whittier	73
Henrietta	72
Todd	70
Drexel	70
Fleetwood	68
Zionville	67
Ferguson	63
Sugar Grove	60
Warrensville	58
Laurel Springs	52
Caroleen	50
State Road	49
Glade Valley	48
McGrady	43
Banner Elk	41
Warne	34
Glen Alpine	34
Lake Lure	34
Cliffside	32
Thurmond	31
Deep Gap	27
Green Mountain	26
Nebo	26
Brasstown	25
Hamptonville	24
Piney Creek	24
Icard	23
Morganton	23
Blowing Rock	21
Almond	18
Rutherford College	17
Jonesville	14
Lenoir	14
Leicester	14
Grassy Creek	12
Weaverville	11
Sylva	9
Micaville	9
Rhodhiss	8
Union Grove	8
Murphy	8
Forest City	7
Harris	7
Elk Park	6
N Wilkesboro	6
Valdese	6
Shelby	6
Granite Falls	6
Marion	5
Spruce Pine	5
Roaring Gap	5
N. Wilkesboro	5
Bostic	5
Galax	4
West Jefferson	4
Rutherfordton	4
Mill Spring	4
Forest city	4
North WIlkesboro	4
BURNSVILLE	4
Mouth of Wilson	3
Ellenboro NC	3
Spindale	3
Sugar Grove	3
North Wilkesboro	3
Harris NC	3
Forest City NC	2
Mc Grady	2
Lawndale	2
Bakersville	2
Caroleen NC	2
Rutherfordton NC	2
Miller's Creek	2
Spindale NC	2

HAYESVILLE	2	State Road	1
Hudson	2	Noth Wilkesboro	1
Fontana Village	2	Purlear	1
Valdese NC	2	Casar	1
Zionville	2	North Wilkesboro NC	1
Beech Mountain	2	BUIRNSVILLE	1
Cherokee	2	Linville	1
Morganton NC	2	Green Mtn	1
Glendale Springs	2	Fletcher	1
MORGANTON	2	Banner Elk	1
Vale	2	Robbinville	1
Green Mountain	2	MOUTH OF WILSON	1
Connellys Springs	2	Andrews	1
Rutherford	2	Ruthefordton	1
Forest City	2	Mroganton	1
Connelly	2	Lansing	1
HIldebran	1	Cullowhee	1
Mooesboro	1	Mill Springs	1
Burlington	1	Green MTN NC	1
Cliffside NC	1	WARNE	1
Chesnee	1	Newton	1
Lincolnton	1	Franklin	1
brasstown	1	SPRUCE PINE	1
Wests Jefferson	1	Roaring River	1
Otto	1	North Wilkesbor	1
Boone, NC	1	Hot Springs	1
Robbinsville	1	Marshall	1
Hendersonville	1	warne	1
Forest City, NC 28043	1	Valese	1
Boone	1	Sparta	1
Cliffside	1	Almond	1
Connelly Spring	1	morganton	1
Piney Crek	1	bryson city	1
Ferguson	1	Wilkesboro28	1
Gaffney	1	Topton	1
Jefferson	1	Hayesville	1
Aberdeen	1	Independence	1
Jefferson	1	GREEN MOUNTAIN	1
Charlotte	1	Wilkeboro	1
Millers	1	Fallston	1
Harris	1	Mount Airy	1
BOSTIC	1	Green Mtn. NC	1
CHerokee	1	MoravianFalls	1
Mooresville	1	Marble	1
N.Wilkesboro	1	Boomer	1
Statesville	1	Cherryville	1
Brasstown	1	Marhall	1
Connelly Springs	1	Glen ALPINE	1
Icard NC	1	Jonesville	1
Burnsville NC	1	Alexander	1
Hiddenite	1	Candler	1
Bryson city	1	CONNELLY SPRINGS	1
Morgnton	1	Foresst City	1
North Wilkesboro, NC	1	Durham	1

```
valdese                    1
Icard                      1
Yadkinville                1
Hiawassee                  1
North Wilesboro            1
Bryson City                1
Mooresboro NC              1
Mars hill                  1
Clyde                      1
Rhodhiss NC                1
Drexel                     1
Dillsboro                  1
Hiawassee                  1
Mars HIll                  1
Wilkesboro, NC 28697       1
Ronda                      1
Deep GAP                   1
Boonville                  1
Moravian Falls,            1
Burnsville                 1
Rutherford College NC      1
Glen Alpine NC             1
Henrrietta                 1
Hildebran NC               1
Scottville                 1
Name: city, dtype: int64
247 cities
```

3. Some of the more interesting cities have only one occurrence. If you sort the index we can see some more details:

```
In [6]: c = df['city'].value_counts().sort_index()
   ...: print(c)
   ...: print(len(c), 'cities')
```

```
 Jefferson                 1
Aberdeen                   1
Alexander                  1
Almond                    18
Almond                     1
Andrews                    1
BOSTIC                     1
BUIRNSVILLE                1
BURNSVILLE                 4
Bakersville                2
Banner Elk                41
Banner Elk                 1
Beech Mountain             2
Blowing Rock              21
Boomer                    75
Boomer                     1
Boone                    281
Boone                      1
Boone, NC                  1
Boonville                  1
Bostic                   341
Bostic                     5
Brasstown                 25
Brasstown                  1
Bryson City              334
Bryson City                1
Bryson city                1
Burlington                 1
Burnsville               291
Burnsville                 1
Burnsville NC              1
CHerokee                   1
CONNELLY SPRINGS           1
Candler                    1
Caroleen                  50
Caroleen NC                2
Casar                      1
Charlotte                  1
Cherokee                 113
Cherokee                   2
Cherryville                1
Chesnee                    1
Cliffside                 32
Cliffside                  1
Cliffside NC               1
Clyde                      1
Connelly                   2
Connelly Spring            1
Connelly Springs         561
Connelly Springs           1
Connellys Springs          2
Creston                   78
Crumpler                  79
Cullowhee                  1
Deep GAP                   1
Deep Gap                  27
Dillsboro                  1
Drexel                    70
Drexel                     1
Durham                     1
Elk Park                   6
Elkin                    140
Ellenboro                528
Ellenboro NC               3
```

Ennice	77
Fallston	1
Ferguson	63
Ferguson	1
Fleetwood	68
Fletcher	1
Fontana Village	2
Foresst City	1
Forest City	2
Forest City	1522
Forest City	7
Forest City NC	2
Forest City, NC 28043	1
Forest city	4
Franklin	1
GREEN MOUNTAIN	1
Gaffney	1
Galax	4
Glade Valley	48
Glen ALPINE	1
Glen Alpine	34
Glen Alpine NC	1
Glendale Springs	2
Granite Falls	6
Grassy Creek	12
Green MTN NC	1
Green Mountain	26
Green Mountain	2
Green Mtn	1
Green Mtn. NC	1
HAYESVILLE	2
HIldebran	1
Hamptonville	24
Harris	7
Harris	1
Harris NC	3
Hayesville	344
Hayesville	1
Hays	179
Hendersonville	1
Henrietta	72
Henrrietta	1
Hiawassee	1
Hiawassee	1
Hickory	359
Hiddenite	1
Hildebran	85
Hildebran NC	1
Hot Springs	84
Hot Springs	1
Hudson	2
Icard	23
Icard	1
Icard NC	1
Independence	1
Jefferson	191
Jefferson	1
Jonesville	14
Jonesville	1
Lake Lure	34
Lansing	147
Lansing	1
Laurel Springs	52
Lawndale	2
Leicester	14
Lenoir	14
Lincolnton	1
Linville	1
MORGANTON	2
MOUTH OF WILSON	1
Marble	1
Marhall	1
Marion	5
Mars HIll	1
Mars Hill	242
Mars hill	1
Marshall	444
Marshall	1
Mc Grady	2
McGrady	43
Micaville	9
Mill Spring	4
Mill Springs	1
Miller's Creek	2
Millers	1
Millers Creek	330
Mooesboro	1
Mooresboro	316
Mooresboro NC	1
Mooresville	1
Moravian Falls	129
Moravian Falls,	1
MoravianFalls	1
Morganton	2216
Morganton	23
Morganton NC	2
Morgnton	1
Mount Airy	1
Mouth of Wilson	3
Mroganton	1
Murphy	8
N Wilkesboro	6
N. Wilkesboro	5
N.Wilkesboro	1
Nebo	26
Newton	1
North WIlkesboro	4
North Wilesboro	1

```
North Wilkesbor              1
North Wilkesboro           937
North Wilkesboro             3
North Wilkesboro NC          1
North Wilkesboro, NC         1
Noth Wilkesboro              1
Otto                         1
Piney Creek                 24
Piney Crek                   1
Purlear                    127
Purlear                      1
Rhodhiss                     8
Rhodhiss NC                  1
Roaring Gap                  5
Roaring River              130
Roaring River                1
Robbinsville               354
Robbinsville                 1
Robbinville                  1
Ronda                      118
Ronda                        1
Ruthefordton                 1
Rutherford                   2
Rutherford College          17
Rutherford College NC        1
Rutherfordton             1092
Rutherfordton                4
Rutherfordton NC             2
SPRUCE PINE                  1
Scottville                   1
Shelby                       6
Sparta                     272
Sparta                       1
Spindale                   317
Spindale                     3
Spindale NC                  2
Spruce Pine                  5
State Road                  49
State Road                   1
Statesville                  1
Sugar Grove                 60
Sugar Grove                  3
Sylva                        9
Thurmond                    31
Todd                        70
Topton                       1
Traphill                    91
Union Grove                  8
Union Mills                126
Valdese                    381
Valdese                      6
Valdese NC                   2
Vale                         2
Valese                       1
Vilas                      113
WARNE                        1
Warne                       34
Warrensville                58
Weaverville                 11
West Jefferson             282
West Jefferson               4
Wests Jefferson              1
Whittier                    73
Wilkeboro                    1
Wilkesboro                 620
Wilkesboro, NC 28697         1
Wilkesboro28                 1
Yadkinville                  1
Zionville                   67
Zionville                    2
brasstown                    1
bryson city                  1
morganton                    1
valdese                      1
warne                        1
Name: city, dtype: int64
247 cities
```

We can see some misspellings, some cities begin with a space, some have the state appended at the end, some have all capital letters, and some are all lowercase. Some cities appear more than once. Perhaps there is some trailing white space.

4. Now we're going to create a data cleaning function and `apply` it to every city in the data set. Copy your code for reading the CSV file into your main function in `clean.py`. It should look like this:

```
def main():
    df = pd.read_csv('basic_person.csv', index_col='acct_id_new')
    # clean the data here.
    print(df.groupby(by='city').count())
```

5. Running your program in the terminal or bash shell:

```
$ python clean.py
Morganton                 2216
Forest City               1522
Rutherfordton             1092
North Wilkesboro           937
Wilkesboro                 620
...
Mill Springs                 1
North Wilkesboro, NC         1
BOSTIC                       1
Lansing                      1
Hot Springs                  1
Name: city, dtype: int64
247 cities
```

6. One way to do this is to write a function that processes one city at a time. Then, use the Series.apply function to process every value in a column using the function. The `Series.apply` function is analogous to Python's built-in `map` function. Let's define a "clean" function above your `main` function:

```
def clean_city(city):
    return city
```

Currently the `clean_city` function is just a placeholder (it doesn't change the city), we will use it process every element in a column using the pandas.Series.apply function. Just below the comment `# clean data here.`, enter the following.

```
    df['city'] = df['city'].apply(clean_city)
```

This says take the `'city'` column and apply the `clean_city` function to every item just like the Python built-in `map` function. Without the assignment, the function does not change your `DataFrame`.

7. First, let's take care of any leading or trailing white spaces. The str.strip function does this for a single string. Add the following line to your `clean_city` function:

```
        city = city.strip()
```

Running your program should produce an error at this point. The `strip` function can only be called on strings and the `'city'` column contains missing values encoded as a `np.nan` (i.e., `float` type). We only want to apply our `clean_city` function to values that are not missing. One way to do this is to check whether or not the value of the `city` variable is an instance of the `str` type using the isinstance function. For example, `isinstance(x, str)` returns `True` when `x` is a string and `False` otherwise. Edit your `clean_city` function to only strip the `city` if it is a string. After making the edit your program should show 208 unique city names:

```
$ python clean.py
Morganton                 2239
Forest City               1529
Rutherfordton             1096
North Wilkesboro           940
Wilkesboro                 620
Connelly Springs           562
Ellenboro                  528
Marshall                   445
Valdese                    387
Hickory                    359
Robbinsville               355
Bostic                     346
Hayesville                 345
Bryson City                335
Millers Creek              330
Spindale                   320
Mooresboro                 316
Burnsville                 292
West Jefferson             286
Boone                      282
Sparta                     273
```

Mars Hill	242
Jefferson	193
Hays	179
Lansing	148
Elkin	140
Roaring River	131
Moravian Falls	129
Purlear	128
Union Mills	126
Ronda	119
Cherokee	115
Vilas	113
Traphill	91
Hildebran	85
Hot Springs	85
Crumpler	79
Creston	78
Ennice	77
Boomer	76
Whittier	73
Henrietta	72
Drexel	71
Todd	70
Zionville	69
Fleetwood	68
Ferguson	64
Sugar Grove	63
Warrensville	58
Laurel Springs	52
State Road	50
Caroleen	50
Glade Valley	48
McGrady	43
Banner Elk	42
Lake Lure	34
Warne	34
Glen Alpine	34
Cliffside	33
Thurmond	31
Green Mountain	28
Deep Gap	27
Nebo	26
Brasstown	26
Icard	24
Piney Creek	24
Hamptonville	24
Blowing Rock	21
Almond	19
Rutherford College	17
Jonesville	15
Leicester	14
Lenoir	14
Grassy Creek	12
Weaverville	11
Sylva	9
Micaville	9
Rhodhiss	8
Murphy	8
Union Grove	8
Harris	8
Granite Falls	6
N Wilkesboro	6
Shelby	6
Elk Park	6
Roaring Gap	5
Marion	5
Spruce Pine	5
N. Wilkesboro	5
Forest city	4
Mill Spring	4
BURNSVILLE	4
North WIlkesboro	4
Galax	4
Ellenboro NC	3
Harris NC	3
Mouth of Wilson	3
Morganton NC	2
Spindale NC	2
Valdese NC	2
Lawndale	2
Rutherford	2
Caroleen NC	2
Mc Grady	2
Glendale Springs	2
Forest City	2
Fontana Village	2
MORGANTON	2
Connellys Springs	2
Rutherfordton NC	2
Hiawassee	2
Vale	2
Hudson	2
Beech Mountain	2
Connelly	2
Bakersville	2
HAYESVILLE	2
Miller's Creek	2
Forest City NC	2
Piney Crek	1
Wilkeboro	1
SPRUCE PINE	1
Millers	1
Glen ALPINE	1
Franklin	1
Hiddenite	1
North Wilesboro	1
Scottville	1
Fletcher	1

```
Glen Alpine NC              1
Henrrietta                  1
Rhodhiss NC                 1
Valese                      1
Rutherford College NC       1
Gaffney                     1
bryson city                 1
Ruthefordton                1
Dillsboro                   1
Mount Airy                  1
HIldebran                   1
Aberdeen                    1
Fallston                    1
Wests Jefferson             1
Casar                       1
Morgnton                    1
Durham                      1
Moravian Falls,             1
Burlington                  1
Foresst City                1
Otto                        1
morganton                   1
Lincolnton                  1
BUIRNSVILLE                 1
Marble                      1
Independence                1
Mars HIll                   1
Deep GAP                    1
Wilkesboro, NC 28697        1
Icard NC                    1
Noth Wilkesboro             1
Hildebran NC                1
Cherryville                 1
warne                       1
Boone, NC                   1
North Wilkesbor             1
valdese                     1
GREEN MOUNTAIN              1
Newton                      1
brasstown                   1
North Wilkesboro NC         1
Burnsville NC               1
BOSTIC                      1
Candler                     1
Cullowhee                   1
Bryson city                 1
North Wilkesboro, NC        1
CONNELLY SPRINGS            1
Statesville                 1
Mooesboro                   1
Cliffside NC                1
N.Wilkesboro                1
Mooresboro NC               1
Chesnee                     1
Wilkesboro28                1
Mroganton                   1
Linville                    1
Clyde                       1
Forest City, NC 28043       1
Topton                      1
Marhall                     1
Robbinville                 1
Yadkinville                 1
Green Mtn. NC               1
CHerokee                    1
Mars hill                   1
MoravianFalls               1
Alexander                   1
MOUTH OF WILSON             1
Green Mtn                   1
Mooresville                 1
Hendersonville              1
Mill Springs                1
WARNE                       1
Green MTN NC                1
Connelly Spring             1
Andrews                     1
Boonville                   1
Charlotte                   1
Name: city, dtype: int64
208 cities
```

8. In addition to the leading and trailing spaces, there may be more than one space between words. To remove them, use the `re.sub` function from the regular expression package. This reduces the number of cities by one to 207. Specifically, "Forest City" is the culprit.

9. Capitalization. You might have noticed that some of the cities are not capitalized while others are in all-capitals. There is a string.title function that capitalizes every word in a string (and unfortunately those after an apostrophe, see "Miller'S Creek"). Edit your `clean_city` function to consistently capitalize the words using `string.title`. Notice that `string` is a Python package that contains useful functions for strings. After doing this, 184 city names should remain:

```
$ python clean.py
Morganton                2242
Forest City              1535
Rutherfordton            1096
North Wilkesboro          944
Wilkesboro                620
Connelly Springs          563
Ellenboro                 528
Marshall                  445
Valdese                   388
Hickory                   359
Robbinsville              355
Hayesville                347
Bostic                    347
Bryson City               337
Millers Creek             330
Spindale                  320
Mooresboro                316
Burnsville                296
West Jefferson            286
Boone                     282
Sparta                    273
Mars Hill                 244
Jefferson                 193
Hays                      179
Lansing                   148
Elkin                     140
Roaring River             131
Moravian Falls            129
Purlear                   128
Union Mills               126
Ronda                     119
Cherokee                  116
Vilas                     113
Traphill                   91
Hildebran                  86
Hot Springs                85
Crumpler                   79
Creston                    78
Ennice                     77
Boomer                     76
Whittier                   73
Henrietta                  72
Drexel                     71
Todd                       70
Zionville                  69
Fleetwood                  68
Ferguson                   64
Sugar Grove                63
Warrensville               58
Laurel Springs             52
State Road                 50
Caroleen                   50
Glade Valley               48
Mcgrady                    43
Banner Elk                 42
Warne                      36
Glen Alpine                35
Lake Lure                  34
Cliffside                  33
Thurmond                   31
Green Mountain             29
Deep Gap                   28
Brasstown                  27
Nebo                       26
Icard                      24
Hamptonville               24
Piney Creek                24
Blowing Rock               21
Almond                     19
Rutherford College         17
Jonesville                 15
Leicester                  14
Lenoir                     14
Grassy Creek               12
Weaverville                11
Micaville                   9
Sylva                       9
Harris                      8
Murphy                      8
Union Grove                 8
Rhodhiss                    8
Granite Falls               6
Shelby                      6
Elk Park                    6
N Wilkesboro                6
Spruce Pine                 6
N. Wilkesboro               5
Roaring Gap                 5
Marion                      5
Mill Spring                 4
Mouth Of Wilson             4
Galax                       4
Harris Nc                   3
Ellenboro Nc                3
Rutherford                  2
Glendale Springs            2
Connellys Springs           2
Mc Grady                    2
Bakersville                 2
Beech Mountain              2
Fontana Village             2
Miller'S Creek              2
Caroleen Nc                 2
Lawndale                    2
Hudson                      2
Rutherfordton Nc            2
Connelly                    2
```

```
Morganton Nc                2
Valdese Nc                  2
Forest City Nc              2
Hiawassee                   2
Spindale Nc                 2
Vale                        2
Hildebran Nc                1
Ruthefordton                1
Noth Wilkesboro             1
Cherryville                 1
Yadkinville                 1
Durham                      1
Glen Alpine Nc              1
Forest City, Nc 28043       1
Piney Crek                  1
Marhall                     1
Franklin                    1
North Wilkesbor             1
Valese                      1
Wests Jefferson             1
Fletcher                    1
Green Mtn Nc                1
Mroganton                   1
Candler                     1
North Wilesboro             1
Millers                     1
Andrews                     1
Hiddenite                   1
Clyde                       1
Green Mtn. Nc               1
Wilkesboro28                1
Boonville                   1
Linville                    1
Gaffney                     1
Topton                      1
Boone, Nc                   1
Burnsville Nc               1
Wilkesboro, Nc 28697        1
Morgnton                    1
Statesville                 1
Wilkeboro                   1
Dillsboro                   1
Mooesboro                   1
Lincolnton                  1
Marble                      1
Moravianfalls               1
Casar                       1
Connelly Spring             1
Robbinville                 1
Cullowhee                   1
Chesnee                     1
North Wilkesboro, Nc        1
Hendersonville              1
Cliffside Nc                1
Scottville                  1
North Wilkesboro Nc         1
N.Wilkesboro                1
Charlotte                   1
Icard Nc                    1
Mill Springs                1
Alexander                   1
Fallston                    1
Mooresboro Nc               1
Green Mtn                   1
Henrrietta                  1
Otto                        1
Rhodhiss Nc                 1
Burlington                  1
Rutherford College Nc       1
Aberdeen                    1
Independence                1
Mount Airy                  1
Foresst City                1
Moravian Falls,             1
Buirnsville                 1
Mooresville                 1
Newton                      1
Name: city, dtype: int64
184 cities
```

10. Now you might notice that some of the cities have their corresponding state and sometimes even zip code following. For example, 'Boone, Nc' and 'Forest City, Nc 28043'. The state was capitalized in the previous step. To remove these you can check if the city name contains a comma and remove the comma and everything else after it. If you search the listing above you should find 5 cities with commas. After performing the transformation you should have 179 unique city names remaining.

11. Then, you can check if the string str.endswith ' Nc'; if so, you can trim the last 3 characters using string indexing. After doing this, 161 cities should remain:

```
$ python clean.py
Morganton            2244
Forest City          1538
Rutherfordton        1098
North Wilkesboro      946
Wilkesboro            621
```

Connelly Springs	563
Ellenboro	531
Marshall	445
Valdese	390
Hickory	359
Robbinsville	355
Hayesville	347
Bostic	347
Bryson City	337
Millers Creek	330
Spindale	322
Mooresboro	317
Burnsville	297
West Jefferson	286
Boone	283
Sparta	273
Mars Hill	244
Jefferson	193
Hays	179
Lansing	148
Elkin	140
Roaring River	131
Moravian Falls	130
Purlear	128
Union Mills	126
Ronda	119
Cherokee	116
Vilas	113
Traphill	91
Hildebran	87
Hot Springs	85
Crumpler	79
Creston	78
Ennice	77
Boomer	76
Whittier	73
Henrietta	72
Drexel	71
Todd	70
Zionville	69
Fleetwood	68
Ferguson	64
Sugar Grove	63
Warrensville	58
Laurel Springs	52
Caroleen	52
State Road	50
Glade Valley	48
Mcgrady	43
Banner Elk	42
Glen Alpine	36
Warne	36
Cliffside	34
Lake Lure	34
Thurmond	31
Green Mountain	29
Deep Gap	28
Brasstown	27
Nebo	26
Icard	25
Hamptonville	24
Piney Creek	24
Blowing Rock	21
Almond	19
Rutherford College	18
Jonesville	15
Lenoir	14
Leicester	14
Grassy Creek	12
Weaverville	11
Harris	11
Micaville	9
Sylva	9
Rhodhiss	9
Union Grove	8
Murphy	8
Shelby	6
Granite Falls	6
Spruce Pine	6
Elk Park	6
N Wilkesboro	6
N. Wilkesboro	5
Roaring Gap	5
Marion	5
Mill Spring	4
Galax	4
Mouth Of Wilson	4
Beech Mountain	2
Connellys Springs	2
Connelly	2
Rutherford	2
Hudson	2
Mc Grady	2
Glendale Springs	2
Miller'S Creek	2
Hiawassee	2
Bakersville	2
Vale	2
Fontana Village	2
Green Mtn	2
Lawndale	2
Cherryville	1
Fletcher	1
Mooesboro	1
Hendersonville	1
Independence	1
Andrews	1
Candler	1

```
Fallston                1
Boonville               1
Chesnee                 1
Topton                  1
Alexander               1
Robbinville             1
N.Wilkesboro            1
Foresst City            1
Burlington              1
Wilkesboro28            1
Mount Airy              1
Charlotte               1
North Wilkesbor         1
Dillsboro               1
Gaffney                 1
Scottville              1
North Wilesboro         1
Mooresville             1
Durham                  1
Moravianfalls           1
Buirnsville             1
Green Mtn.              1
Piney Crek              1
Mroganton               1
Valese                  1
Franklin                1
Millers                 1
Statesville             1
Aberdeen                1
Yadkinville             1
Mill Springs            1
Morgnton                1
Lincolnton              1
Linville                1
Connelly Spring         1
Henrrietta              1
Wests Jefferson         1
Newton                  1
Marble                  1
Marhall                 1
Clyde                   1
Ruthefordton            1
Otto                    1
Cullowhee               1
Casar                   1
Noth Wilkesboro         1
Hiddenite               1
Wilkeboro               1
Name: city, dtype: int64
161 cities
```

12. At this point, fixing the remaining entries is a little tedious. For example, 'Connelly', 'Connelly Spring', and 'Connellys Springs' probably all refer to 'Connelly Spring' which has 563 students. Use the following dictionary to replace the `keys` with their `values`. That is, in your `clean_city` function if the `city` is a key in the dictionary replace it with the key's value.

    http://cs.appstate.edu/~rmp/cs5710/city_name_map.py

    Place the Python file in the same folder as your `clean.py` file and import the dictionary like this:

    ```
    from city_name_map import city_name_map
    ```

    After completing this step, you should have 135 unique city names:

```
$ python clean.py
Morganton              2246
Forest City            1539
Rutherfordton          1101
North Wilkesboro        960
Wilkesboro              623
Connelly Springs        568
Ellenboro               531
Marshall                445
Valdese                 390
Hickory                 359
Robbinsville            356
Bostic                  347
Hayesville              347
Bryson City             337
Millers Creek           330
Spindale                322
Mooresboro              318
Burnsville              298
West Jefferson          287
Boone                   283
Sparta                  273
Mars Hill               244
Jefferson               193
Hays                    179
Lansing                 148
Elkin                   140
Moravian Falls          131
Roaring River           131
Purlear                 128
Union Mills             126
Ronda                   119
```

```
Cherokee                116
Vilas                   113
Traphill                 91
Hildebran                87
Hot Springs              85
Crumpler                 79
Creston                  78
Ennice                   77
Boomer                   76
Whittier                 73
Henrietta                73
Drexel                   71
Todd                     70
Zionville                69
Fleetwood                68
Ferguson                 64
Sugar Grove              63
Warrensville             58
Laurel Springs           52
Caroleen                 52
State Road               50
Glade Valley             48
McGrady                  45
Banner Elk               42
Warne                    36
Glen Alpine              36
Lake Lure                34
Cliffside                34
Green Mountain           32
Thurmond                 31
Deep Gap                 28
Brasstown                27
Nebo                     26
Icard                    25
Piney Creek              25
Hamptonville             24
Blowing Rock             21
Almond                   19
Rutherford College       18
Jonesville               15
Lenoir                   14
Leicester                14
Grassy Creek             12
Weaverville              11
Harris                   11
Rhodhiss                  9
Sylva                     9
Micaville                 9
Murphy                    8
Union Grove               8
Elk Park                  6
Spruce Pine               6
Granite Falls             6
Shelby                    6
Marion                    5
Mill Spring               5
Roaring Gap               5
Galax                     4
Mouth Of Wilson           4
Lawndale                  2
Hudson                    2
Fontana Village           2
Hiawassee                 2
Beech Mountain            2
Vale                      2
Bakersville               2
Miller'S Creek            2
Glendale Springs          2
Cherryville               1
Hendersonville            1
Franklin                  1
Marble                    1
Otto                      1
Hiddenite                 1
Casar                     1
Scottville                1
N.Wilkesboro              1
Valese                    1
Boonville                 1
Durham                    1
Linville                  1
Yadkinville               1
Gaffney                   1
Marhall                   1
Charlotte                 1
Independence              1
Mount Airy                1
Dillsboro                 1
Millers                   1
Chesnee                   1
Andrews                   1
Candler                   1
Alexander                 1
Burlington                1
Newton                    1
Clyde                     1
Lincolnton                1
Cullowhee                 1
Topton                    1
Fletcher                  1
Mooresville               1
Fallston                  1
Statesville               1
Aberdeen                  1
Name: city, dtype: int64
135 cities
```

13. Add a new function called `clean_state` and use it clean the values in the 'state' column of the DataFrame. When you are done there should only be four valid state abbreviations using all capital letters (NC=16085, VA=9, GA=2, SC=1).

14. Add a new function called `clean_zip_code` that cleans the 'zip' column. The zip code may have a combination of letters and numbers. Use the first 5 numbers to represent the 5-digit zip code. If there are fewer than 5 numbers, code it as a missing value using `float('nan')`. For example, the string `'j4325lj62'` should get cleaned to `'43256'`. There should be 165 different 5-digit zip codes after cleaning.

15. Use the `pd.to_csv` function to save the cleaned data frame as `'cleaned.csv'`

16. Put this code in your `main` function above and make sure it creates the file in the same directory as your `clean.py` file.

17. Optional: You might remember from NumPy that we'd rather not call a function for every element in an `array` or in this case a `Series`. It turns out there is a faster way to clean a `Series` object. `Series.str` provides vectorized string functions that can be applied to an entire `Series` and automatically handles missing data correctly. To get a list of available vectorized functions you can use the "tab" key and scrolling:

    ```
    In [7]: df['city'].str.capitalize
         capitalize contains endswith   findall     isalnum   islower   isupper
         casefold   count    extract    get         isalpha   isnumeric join
         cat        decode   extractall get_dummies isdecimal isspace   len     >
         center     encode   find       index       isdigit   istitle   ljust
    ```

    For bonus credit, you can implement the same functionality without importing the `string` package and instead using these vectorized methods. In many cases, this will be faster; however, in this case, it's actually slower and harder to read.

    For another bonus, if you find ways to do a better job of cleaning these data (find a mistake, or way to improve these results), let me know and I will award bonus points.

The resulting `'cleaned.csv'` file should look like this except the rows and columns can be in any order:

```
In [1]: import pandas as pd
   ...: df = pd.read_csv('cleaned.csv')
   ...: df.loc[12041:12060]
Out[1]:
       acct_id_new gender               city state      zip ethnicity
12041        12042      F          Wilkesboro    NC  28697.0         1
12042        12043      F              Harris    NC  28074.0         0
12043        12044      M            Caroleen   NaN  28019.0         0
12044        12045      F       Rutherfordton    NC  28139.0         0
12045        12046      F                 NaN   NaN      NaN         0
12046        12047      F                 NaN   NaN      NaN         0
12047        12048      M           Jefferson    NC  28640.0         0
12048        12049      M       Granite Falls    NC  28630.0         0
12049        12050      M         Glen Alpine    NC  28628.0         0
12050        12051      M           Morganton    NC  28655.0         0
12051        12052      M          Hayesville    NC  28904.0         0
12052        12053      F  Rutherford College   NaN  28671.0         0
12053        12054      M         Forest City    NC  28043.0         0
12054        12055      M       Millers Creek    NC  28651.0         2
```

```
12055        12056      M              Bostic    NC  28018.0         0
12056        12057      F        Robbinsville    NC  28771.0         0
12057        12058      M             Marshall    NC  28753.0         0
12058        12059      F           Jonesville    NC  28642.0         0
12059        12060      M            Morganton    NC  28655.0         0
12060        12061      M        Robbinsville    NC  28771.0         0
```

4 produce

In this section, you will apply your cleaning skills to a real data set showing the quantity and types of produce sold over a period of time.

Start with a Python file called `'produce.py'`. You will need to import the `pandas` package as shown:

```
# produce.py
import pandas as pd

def main():
    # Load and process the 'food.csv' file.
    # Save the file 'cleaned_produce.csv' without the implicit index
    pass

if __name__ == '__main__':
    main()
```

```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('food.csv')

In [3]: df.iloc[:, 1:]
Out[3]:
                    SubCategory        Unit Month Sold  Units Sold
0         Apples, Blushing Gold    1/4 Peck     19-Sep           5
1         Apples, Blushing Gold    1/4 Peck     19-Oct           2
2      Apples, Bramley Seedling    1/4 Peck     19-Sep           4
3              Apples, Chehalis      Bushel     19-Aug           3
4              Apples, Chehalis    1/4 Peck     19-Aug           2
...                         ...         ...        ...         ...
1283  Winter Squash, Spaghetti   4-6 pound     19-Sep           3
1284  Winter Squash, Spaghetti     2-4 lbs     19-Oct           4
1285  Winter Squash, Spaghetti     2-4 lbs     19-Nov           2
1286  Winter Squash, Spaghetti  1-2 pounds     19-Sep           2
1287  Winter Squash, Spaghetti  1-2 pounds     19-Oct           2

[1288 rows x 4 columns]
```

Each row represents a sales item with the category, subcategory, month sold, units sold, and the units. We want to create a table that contains the quantity of each category/subcategory of items sold each month. Different produce items use different units, sometimes units of volume, sometimes units of weight. Some are imperial units others are specific to produce (peck, bushel, etc.). In

addition, the units can vary within a type of produce. For example, "Apples, Chehalis" are sold in Bushels and quarter Pecks.

For the purpose of analyzing the relative quantity of each type of produce being purchased over time, we need each type of produce to have quantities in a consistent unit. It might not matter what unit we pick, it just needs to be consistent within each produce type so that we can see the relative change over time.

This cleaning task differs from the previous ones because you actually need more than one column to clean appropriately. We need to know the `'Unit'` and the `'Units Sold'` to clean both columns. To do this, we need to use the `DataFrame.apply` function and write a function that takes a row of the `DataFrame` as an input argument and returns a new cleaned version of the row. The row is actually a `Series` with the columns as the `Index`.

It will take some time to clean all of the rows, so for this assignment you will only clean the rows for one particular type of produce. Instead of every student cleaning the same item, you will clean the item specific to your App State user name. Use the following code to determine which item of produce to clean:

```
items = [
    'Apples, Early Yellow Transparent', 'Apples, Gala', 'Apples, Gold Rush',
    'Apples, Red Rome Beauty', 'Apples, Spice', 'Basil, Fresh - Sweet Genovese (green)',
    'Beets, Without Greens', 'Collards', 'Garlic Scapes', 'Jerusalem Artichokes',
    'Lettuce, Head', 'Lettuce, Loose Leaf Green', 'Microgreens, Sunshine Mix',
    'Okra, Green', 'Peppers, Bell (Green)', 'Peppers, Jalapeno', 'Pumpkin, Seminole',
    'Rosemary, Fresh', 'Sweet Potatoes, Orange', 'Watermelon, Jubilee'
]

def myhash(user_name):
    import hashlib
    m = hashlib.sha256()
    m.update(bytes(user_name, 'utf-8'))
    return int(m.hexdigest()[:16], 16)

user_name = '<put your ASU user name here>'
item = items[myhash(user_name) % len(items)]
print(f'{user_name} cleans subcategory {item}')
```

For this assignment, you will write a Python program in a file called `produce.py` that like before has a `main` function and the same `if __name__ == '__main__': main()` part. Copy the previous code into the top of your program (not in the `main` function) including assigning your `user_name` to determine which subcategory to clean. Write a program that uses the `DataFrame.apply` function to clean only those rows that have your item subcategory. The other rows should be unaffected. If the unit you are cleaning includes a range of values, use the central value. So, for "Bunch (4-8 oz)" you could set the `'Unit'` to ounce and the `'Units Sold'` to 6.

Save your cleaned file as `'cleaned_produce.csv'`

Some specific conversions from Food Bank of Central New York (Fruit-conversion-chart.pdf, Vegetable-conversion-chart.pdf, otherwise I guessed):

Type	Relation
Apples	1 bushel = 4 pecks
Collards	1 bunch = 14 ounces
Garlic Scapes	1 bundle = 8 ounces
Lettuce, Head	1 case = 28 heads
Okra, Green	1 quart = 1.375 lb
Peppers, Bell (Green)	1 pepper = 1/3 lb
Peppers, Jalapeno	1 pint = 1 small bag = 4 peppers = 16 oz
Rosemary, Fresh	1 bunch = 1 ounce = 2/3 large bunch
Sweet Potatoes, Orange	Pound = 1 Pound = 1 Pound

Submit to Web-CAT to compute your score!

1. Create a ZIP file for your `cs08` folder by right-clicking the folder and selecting:

   - **Send to** → **Compressed (zipped) folder** on Windows
   - **Compress Items** on MacOS
   - **Compress** on Linux

   You should find the new ZIP file in the same directory where `cs08` resides.

2. Login to `http://webcatvm.cs.appstate.edu:8080/Web-CAT` and submit your ZIP file for grading. You may submit as many times as you want before the deadline.