SI 506 Lecture 14

Topics

1. Challenges

1.0 Data

The following data sets are drawn from the the [Center for Disease Control and Prevention](#) (CDC) and the [National Center for Health Statistics](#) (NCHS).

Before commencing the challenges review the data contained in the following files:

File	Source	Description
`mi-covid-washtenaw-202111.csv`	[CDC](#)	COVID-19 county level community transmission levels for Washtenaw County (home of Ann Arbor)
`mi-covid-vax_counties-202111.csv`	[CDC](#)	COVID-19 county level vaccination rates for Michigan counties located within two counties of either Macomb or Wayne counties, inclusive.
`mi-nchs-urban_rural_codes.csv`	[NCHS](#)	Urban/rural classification scheme for U.S. counties and county-equivalent entities (2013)

Examine each file. Note how the data is structured. Determine which "columns" permit the data to be linked. Inspect the string values and consider which values can/ought to be cast to a different type (e.g. string -> integer).

A test fixture CSV file is also provided:

- fxt-vax_ur_countries.csv

You will replicate this file (assigning it a different name).

1.1 Comparing files

Each test fixture CSV file (prefixed with `fxt_`) represents the correct file output that *must* be generated by the program you write. Use them at periodic intervals to compare your current output against the expected output.

💡 In VS Code you can compare or "diff" the file you generate against the appropriate test fixture file. After calling the `write_csv` function and generating a new file do the following:

1. Hover over the file with your cursor then right click and choose the "Select for Compare" option.

2. Next, hover over the appropriate test fixture file then right click and choose the "Compare with Selected" option.

3. Lines highlighted in red indicate character and/or indention mismatches. If any mismatches are encountered revise your code, close the comparison pane, revise your code, regenerate your file, and compare it again to the test fixture file. Repeat as necessary until the files match.

❗ Your output **must** match each fixture file line for line and character for character. Review these files; they are akin to answer keys and should be utilized for comparison purposes as you work your way through the assignment.

2.0 Challenges

Challenge 01

1. In the `main` function block, call the function `read_csv` to retrieve the data from the file `mi-covid-washtenaw-202111.csv`. Assign the return value to a variable named `wash_data`.

2. Use indexing (no loops of any kind) to return the `wash_data` list element with the highest reported test positivity percentage over the previous seven (7) days (e.g. column "percent_test_results_reported_positive_last_7_days") and assign the value of the expression to a variable named `max_pos_tests`.

3. Use slicing (no loops of any kind) to return all elements that contain data for the week 07-13 November 2021 and assign the value of the expression to a variable named `week_02`.

4. Use slicing (no loops of any kind) to reverse the order of the list and assign the value of the expression to a variable named `reverse_order`.

5. BONUS: Use slicing (no loops of any kind) to reverse the order of the daily record elements only, excluding the "header" row (1st element). Assign the slice to a variable named `wash_data_reversed`. Then add the `wash_data` "header' element as the first element in the new list.

```
wash_data_reversed = [
    ['state_name', 'county_name', 'fips_code', 'report_date',
'cases_per_100K_7_day_count_change',
'percent_test_results_reported_positive_last_7_days',
'community_transmission_level'], ['Michigan', 'Washtenaw County',
'26161', '2021/11/20', '409.410', '6.67', 'high'],
    ...
    ['Michigan', 'Washtenaw County', '26161', '2021/11/01', '159.410',
'3.50', 'high']
    ]
```

Challenge 02

Task: Retrieve the CDC data and implement a function that can return a nested list representing a county from the `vax_counties` list.

1. In the `main` function block, call the `read_csv` function and retrieve the Center for Disease Control (CDC) county vaccination rates contained in the file `mi-covid-vax_counties-202111.csv` for

select Michigan counties. Assign the return value to a variable named `vax_data`.

2. The "headers" row constitutes the first element in `vax_data`. Access the element and assign it to a variable named `vax_headers`.

3. Next access the remaining "row" elements and assign them to a variable named `vax_counties`.

   ❗ Recall that individual data elements accessed from a CSV file will be imported as strings. Some strings may need to be converted to a different data type in the challenges below.

4. Implement the function named `get_county`. The function defines two parameters: `counties` (`list`) and `county_name` (`str`). Review the function's docstring for more information about the function's expected behavior.

   💡 the `counties` list contains nested list elements, each of which represents a Michigan county and its county-specific vaccination data.

5. After implementing `get_county` return to the `main` function. Call the function passing it the `vax_counties` list and the string "washtenaw county" (lower case required). Assign the return value to a variable named `washtenaw_cty`.

Challenge 03

Task. Implement a function that permits retrieval of any county element using the CSV "headers" row as an index value lookup mechanism.

1. Implement the function `get_attribute`. The function defines three parameters: `county` (`list`), `headers` (`list`), and `column_name` (`str`). Review the docstring to better understand the function's expected behavior.

   Once implemented you can then employ the function to return any single county attribute (e.g., Series_Complete_Pop_Pct, Series_Complete_12Plus, Series_Complete_65PlusPop_Pct) by leveraging the CSV's "headers" row of column names to look up the element's index value.

   💡 This function can be implemented with one line of code.

2. After implementing `get_attribute` return to the `main` function. Call the function `get_attribute` and retrieve the "Series_Complete_Pop_Pct" value for Washtenaw county by passing the appropriate arguments to the function (i.e., `washtenaw_cty`, `vax_headers`, etc.). Assign the return value to a variable named `washtenaw_cty_vax_complete_pct`.

   💡 The "Series_Complete_Pop_Pct" value represents a county's vaccination completion status (e.g., percent value of fully vaccinated residents).

Challenge 04

Task: Convert CDC floating point numbers represented as strings to type `float`.

1. Implement the function named `convert_to_float`. The function defines a single parameter: `county` (`list`). Review the docstring to better understand the function's expected behavior. The following table lists CDC data that should be converted to a `float`.

Column name	type	convert to
Series_Complete_Pop_Pct	`str`	`float`
Series_Complete_12PlusPop_Pct	`str`	`float`
Series_Complete_18PlusPop_Pct	`str`	`float`
Series_Complete_65PlusPop_Pct	`str`	`float`

In the function block choose a `for` loop that facilitates assigning a new value to a list element.

Employ conditional statements in tandem with the function `is_floating_point_number` (provided) to determine whether or not a string represents a floating point number.

- If a string represents a floating point number (i.e., a number with a fractional component) convert the string to a `float` and assign it to the corresponding element.

- Otherwise, do not mutate the element.

💡 Recall that an `if` statement can include a call to a function that returns a boolean value.

2. After implementing `convert_to_float` return to the `main` function. Loop over the `vax_counties` list and inside the loop block call `convert_to_float` in order to "clean" each county's floating point number data values.

Challenge 05

Task: Identify "outlier" counties with vaccination rates for adults aged 18+ that are either less than fifty percent (50%) or greater than seventy percent (70%).

1. Create an empty accumulator list named `outliers`.

2. Loop over `vax_counties`. In the function block call `get_attribute` and retrieve the "Series_Complete_18PlusPop_Pct" value for each county. If the county's vaccination rate is either less than fifty percent (50%) or greater than seventy percent (70%), append the county to the `outliers` list.

💡 The `continue` statement should prove useful for solving this challenge.

Challenge 06

Task: Retrieve the NCHS data and implement a function to retrieve the Urban/Rural codes. Incorporate the codes into the `vax_counties` list and write the new list to a file.

1. In the `main` function, call the `read_csv` function and retrieve the National Center for Health Statistics (NCHS) urban/rural county codes contained in the file `mi-nchs-urban_rural_codes.csv` for select Michigan counties. Assign the return value to a variable named `nchs_codes_data`.

2. Access the "row" elements (exclude the "header" row) and assign them to a variable named `nchs_codes`.

3. Implement the function named `get_nchs_code`. The function defines two parameters: `nchs_codes` (`list`) and `county` (`list`). Review the function's docstring for more information about the function's expected behavior.

   💡 Note that the function returns a `tuple` comprising a county's NCHS urban/rural code and the urban/rural code descriptor.

4. After implementing `get_nchs_code` return to the `main` function. Implement a `for i in range()` loop sync'd to the length of `vax_counties`. Inside the loop block call `get_nchs_code` and unpack the return value (`< ur_code >, < ur_code_name >`) and assign the items to two variables (names re your choice). Then mutate the county list by placing the UR Code and UR Code Name in the 5th and 6th positions in the list.

5. Likewise, mutate the `vax_headers` list by placing the strings "UR_Code" and "UR_ Code_Name" in the 5th and 6th positions in the list.

6. Call the function `write_csv` and pass to it as arguments the filepath `stu-vax_ur_counties.csv`, `vax_counties`, and `vax_headers`. Compare your output file to the test fixture file `fxt-vax_ur_counties.csv`. Both files *must* match, line for line, and character for character.