assignment1

November 4, 2024

1 SIADS 515 Week 1 Homework (HW1)

1.1 A note about the `%%bash` magic command

The lecture material covered integrating Linux shell commands with Jupyter notebooks using the !, !!, %sx, or %system magic commands. There's one more integration to introduce to you: the %%bash magic command. bash is a popular linux shell, and starting a Jupyter notebook cell with the %%bash magic command will tell Jupyter that the remainder of the cell is to be interpreted as linux (bash) commands.

1.2 A note about the `tee` shell command

For the shell questions in this assignment, you'll generally need to redirect your output to a specifc file for the autograder cells to work. The `tee` command will come in handy, it takes input from STDIN and writes it to STDOUT *and* a file that you specify. This is helpful because you'll be able to see your answer and the file will be writen to allow the autograder to do its work.

[1]:
```
%%bash

# This cell cleans up the directory to avoid problems with the autograder.


echo "Cleaning up, removing created files ..."
echo ""
# remove all files in this directory that don't match these patterns
find "." \
    -not -path "." \
    -not -path "./assets" \
    -not -path "./assets/ride.csv" \
    -not -path "./.ipynb_checkpoints" \
    -not -path "./.ipynb_checkpoints/*" \
    -not -path "./*.ipynb" \
    -print -delete
echo ""
echo "... done."
```

```
Cleaning up, removing created files …

./solution05.csv
```

```
./solution01.txt
./solution03.txt
./solution07.csv
./solution02.txt
./solution06.csv
./solution04.txt

… done.
```

[2]:
```
# This cell checks if the contents of assets/ride.csv have changed
ride_hash = !openssl sha256 assets/ride.csv
assert ride_hash == ['SHA2-256(assets/ride.csv)=␣
↪fc7185388647b0fd8f6d2e8b661f61ff7ad2a3320a0efe6f61a8c459eee344bf'], \
    "Contents of assets/ride.csv seem to have changed."
```

IF THIS CELL ABOVE IS FAILING: Try restoring your ride.csv file with this process:

1. Open your assets folder and see if your ride.csv file is there.

- It it is missing, skip to step 3.

2. If it there, rename it to something else (like ride-broken.csv)
3. Open the "Help" panel at the top of the page.
4. Click "Get latest version"

1.3 Example Question

Copy the first 3 lines of `assets/ride.csv` into a file named `solution_example.txt`

(Note that this is an example problem to demonstrate how to answer shell-based questions in this assignment.)

[3]:
```
%%bash

head -3 assets/ride.csv | tee solution_example.txt
```

```
Type,Local Number,Message,Field 1,Value 1,Units 1,Field 2,Value 2,Units 2,Field
3,Value 3,Units 3,Field 4,Value 4,Units 4,Field 5,Value 5,Units 5,Field 6,Value
6,Units 6,Field 7,Value 7,Units 7,Field 8,Value 8,Units 8,Field 9,Value 9,Units
9,Field 10,Value 10,Units 10,Field 11,Value 11,Units 11,Field 12,Value 12,Units
12,Field 13,Value 13,Units 13,Field 14,Value 14,Units 14,Field 15,Value 15,Units
15,Field 16,Value 16,Units 16,Field 17,Value 17,Units 17,Field 18,Value 18,Units
18,Field 19,Value 19,Units 19,Field 20,Value 20,Units 20,Field 21,Value 21,Units
21,Field 22,Value 22,Units 22,Field 23,Value 23,Units 23,Field 24,Value 24,Units
24,Field 25,Value 25,Units 25,Field 26,Value 26,Units 26,Field 27,Value 27,Units
27,Field 28,Value 28,Units 28,Field 29,Value 29,Units 29,Field 30,Value 30,Units
30,Field 31,Value 31,Units 31,Field 32,Value 32,Units 32,Field 33,Value 33,Units
33,Field 34,Value 34,Units 34,Field 35,Value 35,Units 35,Field 36,Value 36,Units
36,Field 37,Value 37,Units 37,Field 38,Value 38,Units 38,Field 39,Value 39,Units
39,Field 40,Value 40,Units 40,Field 41,Value 41,Units 41,Field 42,Value 42,Units
```

```
42,Field 43,Value 43,Units 43,Field 44,Value 44,Units 44,Field 45,Value 45,Units
45,Field 46,Value 46,Units 46,Field 47,Value 47,Units 47,Field 48,Value 48,Units
48,Field 49,Value 49,Units 49,Field 50,Value 50,Units 50,Field 51,Value 51,Units
51,Field 52,Value 52,Units 52,Field 53,Value 53,Units 53,Field 54,Value 54,Units
54,Field 55,Value 55,Units 55,Field 56,Value 56,Units 56,Field 57,Value 57,Units
57,Field 58,Value 58,Units 58,Field 59,Value 59,Units 59,Field 60,Value 60,Units
60,Field 61,Value 61,Units 61,Field 62,Value 62,Units 62,Field 63,Value 63,Units
63,Field 64,Value 64,Units 64,Field 65,Value 65,Units 65,Field 66,Value 66,Units
66,Field 67,Value 67,Units 67,Field 68,Value 68,Units 68,Field 69,Value 69,Units
69,Field 70,Value 70,Units 70,Field 71,Value 71,Units 71,Field 72,Value 72,Units
72,Field 73,Value 73,Units 73,Field 74,Value 74,Units 74,Field 75,Value 75,Units
75,Field 76,Value 76,Units 76,Field 77,Value 77,Units 77,Field 78,Value 78,Units
78,Field 79,Value 79,Units 79,Field 80,Value 80,Units 80,Field 81,Value 81,Units
81,Field 82,Value 82,Units 82,
Definition,0,file_id,serial_number,1,,time_created,1,,unknown,1,,manufacturer,1,
,product,1,,number,1,,type,1,,
Data,0,file_id,serial_number,"3938543757",,time_created,"896018500",,manufacture
r,"1",,garmin_product,"1765",,type,"4",,,,,,,,
```

[4]:
```python
# Example Question, Check 1
# 0 points

with open('solution_example.txt') as ans:
    lines = ans.readlines()

assert len(lines) == 3, 'solution file must be 3 lines long'

# There are no hidden autograder tests in this cell.
```

1.4 Question 1

Copy the last 5 lines of `assets/ride.csv` into a file called `solution01.txt`.

[5]:
```bash
%%bash

tail -n 5 assets/ride.csv > solution01.txt
#raise NotImplementedError()
```

[6]:
```python
# Question 1, Check 1
# 1 point

with open('solution01.txt') as ans:
    lines = ans.readlines()

assert len(lines) == 5, 'solution file must be 5 lines long'

# There are no hidden autograder tests in this cell.
```

[7]:
```
# Question 1, Check 2
# 1 point

# There are hidden autograder tests in this cell.
```

1.5 Question 2

Copy lines 15 through 20 from `assets/ride.csv` into a file called `solution02.txt`.

[8]:
```
%%bash

sed -n '15,20p' assets/ride.csv > solution02.txt

#raise NotImplementedError()
```

[9]:
```
# Question 2, Check 1
# 1 point

with open('solution02.txt') as ans:
    lines = ans.readlines()

assert len(lines) == 6, 'solution file must be 6 lines long'

# There are no hidden autograder tests in this cell.
```

[10]:
```
# Question 2, Check 2
# 1 points

# There are hidden autograder tests in this cell.
```

1.6 Question 3

Copy all lines from `assets/ride.csv` that contain “Data,10”, “Data,11”, “Data,12”, or “Data,13” into a file called `solution03.txt`. The lines in the solution should be in the same order as they appear in the ride.csv file.

[11]:
```
%%bash

grep -E "Data,10|Data,11|Data,12|Data,13" assets/ride.csv > solution03.txt

#raise NotImplementedError()
```

[12]:
```
# Question 3, Check 1
# 1 point

with open('solution03.txt') as ans:
    lines = ans.readlines()
```

```
assert len(lines) > 1, 'solution file must have more than one line'

# There are no hidden autograder tests in this cell.
```

[13]:
```
# Question 3, Check 2
# 1 point

# There are hidden autograder tests in this cell.
```

1.6.1 Question 4

How many lines in `assets/ride.csv` contain the string “Data,9”?

Create a file called `solution04.txt` that contains answer (this file should have one line, and that should be a number).

[14]:
```
%%bash

grep -c "Data,9" assets/ride.csv > solution04.txt

#raise NotImplementedError()
```

[15]:
```
# Question 4, Check 1
# 1 point

with open('solution04.txt') as ans:
    lines = ans.readlines()

assert len(lines) == 1, 'solution file should be 1 line long'

# There are no hidden autograder tests in this cell.
```

[16]:
```
# Question 4, Check 2
# 1 point

# There are hidden autograder tests in this cell.

# Checking if line 1 of solution04.txt ONLY contains the correct answer.
```

1.7 Three solutions to the same problem...

For Question 5, 6, 7, you will create a CSV file for each with the same content based on the data from `assets/ride.csv`. You’ll use a different solution for each problem: - Question 5: with shell commands - Question 6: with Python (without Pandas) - Question 7: with Pandas

You’ll notice that although the `ride.csv` file is a csv file, it looks a bit different than most csv files. The first few columns are always of the form (using regular expression notation)

"(Data|Definition),[\d]+,record". After those columns, the data appear as a key-value-unit triple. For example:

```
Data,9,record,timestamp,"896018545",s,position_lat,"504719750",semicircles,position_long,"-998
```

In this case, you have:

```
timestamp,"896018545",s
position_lat,"504719750",semicircles
position_long,"-998493490",semicircles
speed,"1.773",m/s
```

The data we are interested in comes from the lines in `assets/ride.csv` that start with "Data,9,record".

For each question, you will create a CSV file with these headers: - `timestamp` - `latitude` - `longitude` - `distance` - `altitude` - `speed`

The table below shows the required headers and gives the first row as an example. (Your solution should provide rows for all the entries that match the conditions above.)

timestamp	latitude	longitude	distance	altitude	speed
896018545	504719750	-998493490	10.87	285.79999999999995	1.773
…	…	…	…	…	…
…	…	…	…	…	…

1.8 Question 5 - Shell

Create a new file that matches the format described above. - The file should be named `solution05.csv` - Solve the problem using only Linux shell commands

A reminder about appending to existing files: remember that you can append to an existing file with the `>>` operator. This works just like the `>` operator. The difference is: `- > target_file` will erase the target file before writing to it `- >> target_file` will append to the target file

You can also do something similar with the tee command: `- | tee target_file` will erase the target file before writing to it `- | tee -a target_file` (note the -a flag) will append to the target file

[17]:
```
%%bash

echo "timestamp,latitude,longitude,distance,altitude,speed" > solution05.csv

grep -E 'Data,9,record' assets/ride.csv | awk -F, '
{
    for (i=1; i<=NF; i++) {
        if ($i == "timestamp") ts=$(i+1);
        if ($i == "position_lat") lat=$(i+1);
        if ($i == "position_long") long=$(i+1);
        if ($i == "distance") dist=$(i+1);
        if ($i == "altitude") alt=$(i+1);
        if ($i == "speed") spd=$(i+1);
    }
    print ts "," lat "," long "," dist "," alt "," spd;
}
' >> solution05.csv

#raise NotImplementedError()
```

[18]:
```
# Question 5, Check 1
# 1 point

import csv

def close_match(a, b):
    return round(float(a), 3) == round(float(b), 3)


with open('solution05.csv') as ans:
    ans_reader = csv.reader(ans, delimiter=',', quotechar='"')

    header = next(ans_reader)
    assert header[0] == 'timestamp', 'incorrect header'
    assert header[1] == 'latitude', 'incorrect header'
    assert header[2] == 'longitude', 'incorrect header'
    assert header[3] == 'distance', 'incorrect header'
    assert header[4] == 'altitude', 'incorrect header'
    assert header[5] == 'speed', 'incorrect header'

    first_line = next(ans_reader)
    # checks values to three decimal places
    assert close_match(first_line[0], 896018545), 'incorrect value'
    assert close_match(first_line[1], 504719750), 'incorrect value'
    assert close_match(first_line[2], -998493490), 'incorrect value'
    assert close_match(first_line[3], 10.87), 'incorrect value'
    assert close_match(first_line[4], 285.79999999999995), 'incorrect value'
    assert close_match(first_line[5], 1.773), 'incorrect value'

# There are no hidden autograder tests in this cell.
```

[19]:
```
# Question 5, Check 2
# 1 point

# There are hidden tests in this cell to check the solution has the correct␣
 ↪number of rows.
```

[20]:
```
# Question 5, Check 3
# 12 points
# There are hidden tests in this cell to check the solution has all the correct␣
↪values for each column.
```

1.9 Q06 - Python without Pandas

Create a new file that matches the format described above: - The file should be named `solution06.csv` - Don't use any shell or Jupyter magic commands - Solve the problem using Python - You may use the Python standard library (e.g. the built-in csv package) - You may NOT use third-party libraries (e.g. Pandas)

[21]:
```
import csv

# Define the output file
output_file = 'solution06.csv'

# Create and write the headers to the output file
with open(output_file, 'w', newline='') as outfile:
    writer = csv.writer(outfile)
    writer.writerow(['timestamp', 'latitude', 'longitude', 'distance',␣
↪'altitude', 'speed'])

# Open the input file and process it line by line
with open('assets/ride.csv', newline='') as infile:
    reader = csv.reader(infile)
    for row in reader:
        if row[:3] == ['Data', '9', 'record']:
            timestamp = row[row.index('timestamp') + 1]
            latitude = row[row.index('position_lat') + 1]
            longitude = row[row.index('position_long') + 1]
            distance = row[row.index('distance') + 1]
            altitude = row[row.index('altitude') + 1]
            speed = row[row.index('speed') + 1]

            # Write the extracted data to the output file
            with open(output_file, 'a', newline='') as outfile:
                writer = csv.writer(outfile)
                writer.writerow([timestamp, latitude, longitude, distance,␣
↪altitude, speed])


#raise NotImplementedError()
```

[22]:
```
# Question 6, Check 1
# 1 point

import csv
```

```
def close_match(a, b):
    return round(float(a), 3) == round(float(b), 3)


with open('solution06.csv') as ans:
    ans_reader = csv.reader(ans, delimiter=',', quotechar='"')

    header = next(ans_reader)
    assert header[0] == 'timestamp', 'incorrect header'
    assert header[1] == 'latitude', 'incorrect header'
    assert header[2] == 'longitude', 'incorrect header'
    assert header[3] == 'distance', 'incorrect header'
    assert header[4] == 'altitude', 'incorrect header'
    assert header[5] == 'speed', 'incorrect header'

    first_line = next(ans_reader)
    # checks values to three decimal places
    assert close_match(first_line[0], 896018545), 'incorrect value'
    assert close_match(first_line[1], 504719750), 'incorrect value'
    assert close_match(first_line[2], -998493490), 'incorrect value'
    assert close_match(first_line[3], 10.87), 'incorrect value'
    assert close_match(first_line[4], 285.79999999999995), 'incorrect value'
    assert close_match(first_line[5], 1.773), 'incorrect value'

# There are no hidden autograder tests in this cell.
```

[23]:
```
# Question 6, Check 2
# 1 point

# There are hidden tests in this cell to check the solution has the correct␣
↪number of rows.
```

[24]:
```
# Question 6, Check 3
# 12 points

# There are hidden tests in this cell to check the solution has all the correct␣
↪values for each column.
```

1.10 Q07 - Pandas

Create a new file that matches the format described above: - The file should be named `solution07.csv` - Don't use any shell or Jupyter magic commands - Solve the problem using functionality provided by the Pandas library in Python

The How do I select a subset of a DataFrame? part of the pandas tutorial could be helpful, if you need a refresher on filtering and selecting data in a DataFrame.

[25]:
```
import pandas as pd

# Read the CSV file
data = pd.read_csv('assets/ride.csv', header=None)

# Filter rows starting with 'Data,9,record'
filtered_data = data[data.iloc[:, :3].apply(lambda x: x.tolist() == ['Data',␣
↪'9', 'record'], axis=1)]

# Extract the relevant columns
timestamps = filtered_data.apply(lambda row: row[row == 'timestamp'].index[0] +␣
↪1, axis=1)
latitudes = filtered_data.apply(lambda row: row[row == 'position_lat'].index[0]␣
↪+ 1, axis=1)
longitudes = filtered_data.apply(lambda row: row[row == 'position_long'].
↪index[0] + 1, axis=1)
distances = filtered_data.apply(lambda row: row[row == 'distance'].index[0] +␣
↪1, axis=1)
altitudes = filtered_data.apply(lambda row: row[row == 'altitude'].index[0] +␣
↪1, axis=1)
speeds = filtered_data.apply(lambda row: row[row == 'speed'].index[0] + 1,␣
↪axis=1)

# Create the output DataFrame
output = pd.DataFrame({
    'timestamp': filtered_data.apply(lambda row: row.iloc[timestamps[row.
↪name]], axis=1),
    'latitude': filtered_data.apply(lambda row: row.iloc[latitudes[row.name]],␣
↪axis=1),
    'longitude': filtered_data.apply(lambda row: row.iloc[longitudes[row.
↪name]], axis=1),
    'distance': filtered_data.apply(lambda row: row.iloc[distances[row.name]],␣
↪axis=1),
    'altitude': filtered_data.apply(lambda row: row.iloc[altitudes[row.name]],␣
↪axis=1),
    'speed': filtered_data.apply(lambda row: row.iloc[speeds[row.name]], axis=1)
})

# Write to the CSV file
output.to_csv('solution07.csv', index=False)

##raise NotImplementedError()
```

[26]:
```
# Question 7, Check 1
# 1 point
```

```
import csv

def close_match(a, b):
    return round(float(a), 3) == round(float(b), 3)


with open('solution07.csv') as ans:
    ans_reader = csv.reader(ans, delimiter=',', quotechar='"')

    header = next(ans_reader)
    assert header[0] == 'timestamp', 'incorrect header'
    assert header[1] == 'latitude', 'incorrect header'
    assert header[2] == 'longitude', 'incorrect header'
    assert header[3] == 'distance', 'incorrect header'
    assert header[4] == 'altitude', 'incorrect header'
    assert header[5] == 'speed', 'incorrect header'

    first_line = next(ans_reader)
    # checks values to three decimal places
    assert close_match(first_line[0], 896018545), 'incorrect value'
    assert close_match(first_line[1], 504719750), 'incorrect value'
    assert close_match(first_line[2], -998493490), 'incorrect value'
    assert close_match(first_line[3], 10.87), 'incorrect value'
    assert close_match(first_line[4], 285.79999999999995), 'incorrect value'
    assert close_match(first_line[5], 1.773), 'incorrect value'

# There are no hidden autograder tests in this cell.
```

[27]:
```
# Question 7, Check 2
# 1 point

# There are hidden tests in this cell to check the solution has the correct␣
 ↪number of rows.
```

[28]:
```
# Question 7, Check 3
# 12 points

# There are hidden tests in this cell to check the solution has all the correct␣
 ↪values for each column.
```