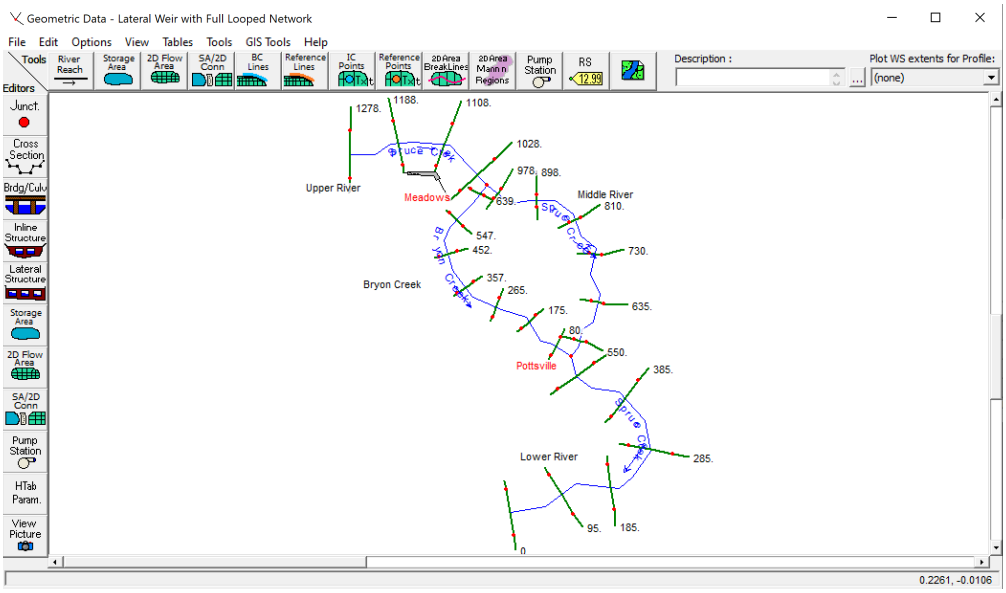
ME 498 K; Spring 2023 Homework 5

Aidan Hunt (ahunt94@uw.edu) Due: May 5, 2023

In this homework, you will practice importing, manipulating, and analyzing Pandas DataFrames. Post questions about this assignment to the class discussion board for the fastest response. Submit your response as a single `.py` file (this is to encourage you to use an IDE with a variable viewer and debugger, like Spyder).

Background

The Hydraulic Engineering Center's River Analysis System (HEC-RAS) is a tool for simulating river flows, and is useful for assessing how river channels, as well as structures in them like bridges and weirs, react to various flow profiles (like floods). We'll be looking at data from Spruce Creek and Bryon Creek in Pennsylvania. A simple discretized model of these creeks as implemented in HEC-RAS is shown below.



Below are the important points about this model for understanding the data you will work with:

- There are several "river stations" labeled with numbers which indicate points along the river. The furthest upstream river station is "1278", and the furthest downstream station is "0".
- The blue lines represent the "rivers" (they are labeled "Spruce Creek" and "Bryon Creek").
- Spruce creek is split into three "reaches": "Upper River", "Middle River", and "Low River". Bryon Creek has only one reach, also named "Bryon Creek".
- Below the "Upper River" section of Spruce Creek, the creek splits into "Bryon Creek" and the "Middle River" section of Spruce Creek, before rejoining downstream as the "Lower River" of Spruce Creek.

Flow through these creeks is simulated in HEC-RAS for three flow profiles: "Low Water", "Standard Flow", and "Flood". HEC-RAS estimates several flow metrics at each river station, such as the total flow rate, water surface elevation, the velocity in the channel, and the Froude number. The resulting data is saved in `hecras_flow_split.txt`. This assignment has two main components. First, you will import the data into Python as a DataFrame using the Pandas package. Second, you will develop a single generalized plotting function to visualize various slices of this dataset.

Importing the Data

Your first task is to write a function that imports the data into Python using the Pandas package.

```
def importHECRAS(fileName):
    ''' docstring goes here '''
```

As in class, you should use the pd.read_csv function to create a DataFrame from the HEC-RAS data. When using `pd.read_csv`, you will need to specify values for several optional parameters:

- You may assume that individual columns are separated by 2 or more space characters. Set the `delimiter` parameter to `'\s{2,}'` and the `engine` parameter to `'python'`.
- You should ignore all content in the file before the first line of actual data, as well as any empty lines in the data (find the correct parameters to set in the `pd.read_csv` documentation).

Additionally, your final DataFrame should have the following columns:

1. "River" : Name of the river corresponding to each row of data.
2. "Reach" : Name of the reach corresponding to each row of data.
3. "Station" : River station number corresponding to each row of data.
4. "Profile" : Name of the flow profile corresponding to each row of data.
5. "Q" : Flow rate at each River/Reach/Station/Profile combination, in cfs
6. "h" : Water surface (W. S.) elevation at each River/Reach/Station/Profile combination, in ft
7. "V" : Velocity in the channel at each River/Reach/Station/Profile combination, in ft/s
8. "Fr" : The Froude number at each River/Reach/Station/Profile combination (dimensionless)

Note that this means we are only keeping a subset of the columns in the text file, and we are specifying the column names directly (there are a number of ways that you can accomplish these steps). You do not need to specify a new index for the DataFrame.

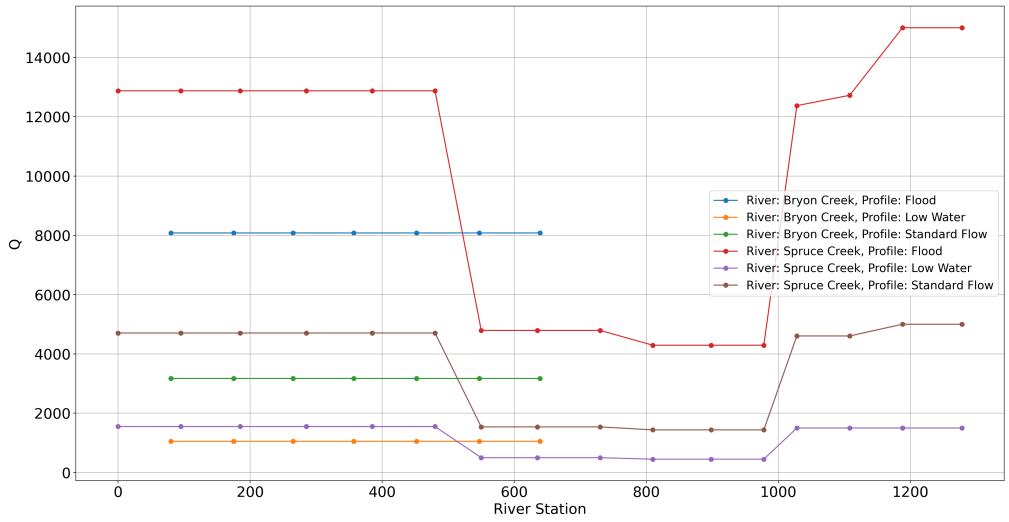
Visualizing the Data

Your next task is to write a flexible plotting function for visualizing various flow metrics as a function of river station and flow profile. The river stations and flow profiles plotted will be chosen based on user inputs.

```
def plotHECRAS(riverData, metric, riverName=None, reachName=None, profileName=None)
    ''' docstring goes here '''
```

Your function will take two required parameters: `riverData`, which is the full DataFrame as imported in the section above, and `metric`, which is a `str` that represents a particular flow metric we are interested in plotting. For example, a call to `plotHECRAS(riverData, 'Q')` would plot the flow rate `riverData['Q']` vs `riverData['Station']` for all rivers and profiles in `riverData` as in the plot shown.

However, your function will also take optional parameters `riverName`, `reachName`, and `profileName`. If the user specifies a particular river (case-sensitive), then only data corresponding to that river should be plotted. Similarly, if a reach is provided (case-sensitive), only data corresponding to river stations in that reach should be plotted. Similarly, if the user specifies a particular profile name (e.g., "Flood", case-sensitive), then only data corresponding to that profile should be plotted. To accomplish this, you will want to slice the rows of the input DataFrame according to the user inputs. `Examples of output plots with these optional inputs are posted on the Canvas page for this assignment`.



As in past assignments, we want to handle cases in which the user provides invalid parameters. If the user inputs an invalid metric (i.e., a metric that is not `Q`, `h`, `V`, or `Fr`), your function should not produce any plots, and instead just print a message to the console saying that this column was not found in the input DataFrame. Similarly, if the user inputs invalid values for `riverName`, `reachName`, or `profileName`, or incompatible values for river and reach name (e.g., `riverName='Bryon Creek'` and `reachName='Upper River'`), your function should not produce any plots, and instead just print a message to the console indicating that slicing the data using these inputs results in an empty DataFrame. **You do not need to raise exceptions**; printing these messages is sufficient.

```
plotHECRAS(riverData, 'conductivity')
Column conductivity not found in the input DataFrame.
```

```
plotHECRAS(df, 'h', riverName='Spruce Creek', reachName='Waaaay upstream')
Resulting DataFrame is empty: check function inputs.
```

Writing such a generalized function may seem daunting, but by outlining this in pseudocode, we can break this seemingly complex behavior down into several simpler cases. First, we only want to move forward with analysis if the user provided a valid metric, so we can check that right away. In the metric is indeed valid, we can then slice the DataFrame to include only the subset of rivers, reaches, and profiles based on the user's inputs (if provided). If there is something left to plot, we can do so. Otherwise, we can inform the user that the DataFrame is empty and end execution.

```
def plotHECRAS(df, metric, riverName=None, reachName=None, profileName=None)
    ''' docstring goes here '''

    # If the provided metric is Q, h, V, or Fr
        # Slice the data frame based on the river name, if provided
        # Further slice the DataFrame based on the reach name, if provided
        # Further slice the DataFrame based on the given profile, if provided

        # If the final slice is not empty:
            # Plot metric vs river station for all rivers and profiles in the slice

        # Otherwise
            # Print message saying slice is empty

    # Otherwise
        # Print message saying metric is not in DataFrame
```

Additionally, note that slicing the DataFrame based on user inputs (but only if the user provided a value) is a similar task for the `riverName`, `reachName`, or `profileName` parameters...rather than copy/pasting similar value-checking code for all three of these parameters, we should write a flexible helper function that does this sort of "slice if user provided a value" task for us! You can use the following partially completed code; you only need to add a line of code that slices the DataFrame rows based on `colName` and `sliceVal`.

```
def sliceIfValid(df, colName, sliceVal):
    '''
    Given an input DataFrame (df), slices the rows of df to include only
    those where the column given by the string colName is equal to the
    value given by sliceVal. The sliced DataFrame is returned.

    If sliceVal is equal to None, or colName is not the name of a column
    in df, the original DataFrame is returned unsliced.
    '''

    if (sliceVal != None) and (colName in df.columns):
        # Slice the DataFrame somehow

    return df
```

Grading Criteria

In terms of output, your code will be graded on whether it 1) correctly imports the HEC-RAS data as a DataFrame in the requested format, 2) correctly plots a requested metric (`Q`, `h`, `V`, or `Fr`) as a function of river station, and 3) correctly handles the optional inputs that may be provided by the user. The plots generated by your function should have **labeled x and y axes**, as well as a **legend denoting the river and profile of each line** (see example plot). Otherwise, there are no other formatting requirements for your plot.

In terms of style, your code should be well-structured and well-documented as always, and follow the Style Guide. You don't need to define more than the three functions described in this specification (you may define more if you would find it helpful), but **every** function must have a docstring that describes its parameters, returns, and other behavior. Remember, how your functions respond to both valid and invalid user input cases is important behavior that should be documented.

You should be able to accomplish the tasks in this assignment with the `pd.read_csv` documentation and the slicing syntax we have used in class. If you need help with any aspect of this assignment, please post on the discussion board. Additionally, if you have any questions about general style requirements or the specific style requirements for this assignment, post them to the discussion board.