## Project 1 – Data professional track

## Project Submission

In this project, you will write Python code to import US bike share data and answer interesting questions about it by computing descriptive statistics. You will also write a script that takes in raw input to create an interactive experience in the terminal to present these statistics.

## Before You Submit

#### Check the Rubric

Your project will be evaluated by a Udacity reviewer according to this [Project Rubric](https://review.udacity.com/#!/rubrics/1379/view). Be sure to review it thoroughly before you submit. Your project "meets specifications" only if it meets specifications in all the criteria. If you see room for improvement in any category in which you do not meet specifications, be sure to take some time to revise your work until you feel it is up to expectations. In particular, there is one section of the rubric that focuses on the quality of your code. It is important that you not only obtain the correct answers with your code, but that you have followed good coding practices to obtain your solutions.

#### Gather Submission Materials

All you need to submit for this project is two files:

* **bikeshare.py**: Your code
* **readme.txt**: If you refer to other websites, books, and other resources to help you in solving tasks in the project, make sure that you document them in this file

There is no need for you to include any data files with your submission.

## Submitting the Project

When you're ready, click on the "Submit Project" button to go to the project submission page. You can submit your files as a .zip archive or you can link to a GitHub repository containing your project files. If you go with GitHub, note that your submission will be a snapshot of the linked repository at the time of submission. It is recommended that you keep each of your projects in a separate repository to avoid any potential confusion: if a reviewer gets multiple folders representing multiple projects, there might be confusion regarding what project is to be evaluated.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. In the meantime, you should feel free to proceed with your learning journey by continuing on to the next module in the program

**PROJECT SPECIFICATION**

**Explore US Bikeshare Data**

Code Quality

| CRITERIA | MEETS SPECIFICATIONS |
| --- | --- |
| Functionality of code | All code cells can be run without error.  *Tips*: Implement safeguards against invalid user inputs that can potentially break the codes. Please refer to the “Solicit and handle raw user input” rubric item for further details. |
| Choice of data types and structures | Appropriate data types (e.g. strings, floats) and data structures (e.g. lists, dictionaries) are chosen to carry out the required analysis tasks. |
| Use of loops and conditional statements | Loops and conditional statements are used to process the data correctly. |
| Use of packages | Packages are used to carry out advanced tasks. |
| Use of functions | Functions are used to reduce repetitive code. |
| Use of good coding practices | Docstrings, comments, and variable names enable the readability of the code.  *Tips*: Please refer to the Python’s documentation [PEP 257 -- Docstring Conventions](https://www.python.org/dev/peps/pep-0257/). Example of docstring conventions:  def function(a, b):  """Do X and return a list.""" |

Script and Questions

| CRITERIA | MEETS SPECIFICATIONS |
| --- | --- |
| Solicit and handle raw user input | Raw input is solicited and handled correctly to guide the interactive question-answering experience; no errors are thrown when unexpected input is entered.  User inputs should be made **case insensitive**, which means the input should accept the string of "Chicago" and its case variants, such as "chicago", "CHICAGO", or "cHicAgo".  You should also implement **error handlings** so your program does not throw any errors due to invalid inputs. For example, if the user enters "Los Angeles" for the city, the error handling should reject the user input and avoid breaking the codes. |
| Use descriptive statistics to answer questions about the data. Raw data is displayed upon request by the user. | Descriptive statistics are **correctly computed** and used to answer the questions posed about the data.  Raw data is displayed upon request by the user in the following manner:   * Your script should prompt the user if they want to see 5 lines of raw data, * Display that data if the answer is 'yes', * Continue iterating these prompts and displaying the **next** 5 lines of raw data at each iteration, * Stop the program when the user says 'no' or there is no more raw data to display.   Tips: you can implement the while loop and track the row index in order to display the continuous raw data. |

**Suggestions to Make Your Project Stand Out!**

* Change the structure of bikeshare.py to make the code more efficient or in better style.
* Ask and answer additional questions about the data beyond the questions already provided.
* Make the interactive experience wow-worthy! Add images, make it into a web app, etc. (If you do create a web app, make sure to include clear directions how to execute it.) Make it your own!