

[← Back to Data Analyst Nanodegree](#)

Explore US Bikeshare Data

REVIEW

CODE REVIEW 3

HISTORY

▼ home/bikeshare.py 1

```
1 import time
2 import pandas as pd
3 import numpy as np
4
5
6 CITY_DATA = { 'chicago': 'chicago.csv',
7               'new york': 'new_york_city.csv',
8               'washington': 'washington.csv' }
9
10 def get_filters():
11     """
12     Asks user to specify a city, month, and day to analyze.
13     Returns:
14         (str) city - name of the city to analyze
15         (str) month - name of the month to filter by, or "all" to apply no month filter
16         (str) day - name of the day of week to filter by, or "all" to apply no day filter
17     """
18
19     invalid_inputs = "Invalid input. Please try again with the mentioned city names"
20
21     print('Hey there! Let\'s now explore some US bikeshare data!')
22     # TO DO: get user raw_input for city (chicago, new york city, washington). HINT: Use a while loop to handle i
23     while 1 == 1 :
```

SUGGESTION

Convention is `while True`

```
24         city = input("\nEnter the name of the city to analyze, City names are as follows:\nchicago, new york, washington")
25         if city in ['chicago', 'new york', 'washington']:
26             break
27         else:
28             print(invalid_inputs)
29
30     # TO DO: get user raw_input for month (all, january, february, ... , june)
31     while 1 == 1 :
32         month = input("\nEnter the month for which you want to analyze:\njanuary, february, march, april, may, june\n")
33         if month in ['january', 'february', 'march', 'april', 'may', 'june', 'all']:
34             break
35         else:
36             print(invalid_inputs)
37
38     # TO DO: get user raw_input for day of week (all, monday, tuesday, ... sunday)
39     while 1 == 1 :
40         day = input("\nEnter the day for which you want the data \nmonday, tuesday, wednesday, thursday, friday, saturday, sunday\n")
41         if day in ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday', 'all']:
42             break
43         else:
44             print(invalid_inputs)
45
46     print('\n\n')
47     return city, month, day
48
49
50
51 def load_data(city, month, day):
52     """
53     Loads data for the specified city and filters by month and day if applicable.
54     Args:
55         (str) city - name of the city to analyze
56         (str) month - name of the month to filter by, or "all" to apply no month filter
57         (str) day - name of the day of week to filter by, or "all" to apply no day filter
58     Returns:
59         df - Pandas DataFrame containing city data filtered by month and day
60     """
61     file_name = CITY_DATA[city]
62     print('Accessing data from: ' + file_name)
63     df = pd.read_csv(file_name)
64
65     # convert the Start Time column to datetime
66     df['Start Time'] = pd.to_datetime(df['Start Time'], format = '%Y-%m-%d %H:%M:%S')
67
68     # filter by month if applicable
69     if month != 'all':
70         # extract month and day of week from Start Time to create new columns
71         df['month'] = df['Start Time'].dt.month
72
73         # use the index of the months list to get the corresponding int
74         months = ['january', 'february', 'march', 'april', 'may', 'june']
75         month = months.index(month) + 1
76
77         # filter by month to create the new dataframe
78         df = df.loc[df['month'] == month]
79
80     # filter by day of week if applicable
81     if day != 'all':
82         df['day_of_week'] = df['Start Time'].dt.weekday_name
83
84         # filter by day of week to create the new dataframe
85         df = df.loc[df['day_of_week'] == day.title()]
86     return df
87
88
89 def time_stats(df):
90     """Displays statistics on the most frequent times of travel."""
91
92     print('\nMost Frequent Times of Travel...\n')
93     start_time = time.time()
94
95     # Convert the Start Time column to datetime
96     df['Start Time'] = pd.to_datetime(df['Start Time'], format = '%Y-%m-%d %H:%M:%S')
97
98     # Create new columns for month, weekday, hour
99     month = df['Start Time'].dt.month
100     weekday_name = df['Start Time'].dt.weekday_name
101     hour = df['Start Time'].dt.hour
102
103     # TO DO: display the most common month
104     most_common_month = month.mode()[0]
105     print('Most common month: ', most_common_month)
106
107     # TO DO: display the most common day of week
108     most_common_day_of_week = weekday_name.mode()[0]
109     print('Most common day of week: ', most_common_day_of_week)
110
111     # TO DO: display the most common start hour
112     common_start_hour = hour.mode()[0]
113     print('Most frequent start hour: ', common_start_hour)
114
115     print("\nThis took %s seconds." % (time.time() - start_time))
116     print('\n\n')
117
118 def station_stats(df):
119     """Displays statistics on the most popular stations and trip."""
120
121     print('\nMost Popular Stations and Trip...\n')
122     start_time = time.time()
123
124     # TO DO: display most commonly used start station
125     print('Most commonly used start station:', df['Start Station'].value_counts().idxmax())
126
127     # TO DO: display most commonly used end station
128     print('Most commonly used end station:', df['End Station'].value_counts().idxmax())
129
130     # TO DO: display most frequent combination of start station and end station trip
131     combine_stations = df['Start Station'] + " " + df['End Station']
132     common_station = combine_stations.value_counts().idxmax()
133     print('Most frequent trips are:\n{} \n{}\n'.format(common_station.split(' ')[0], common_station.split(' ')[1]))
134
135     print('\n\n')
136
137 def trip_duration_stats(df):
138     """Displays statistics on the total and average trip duration."""
139
140     print('\nCalculating Trip Duration...\n')
141     start_time = time.time()
142
143     # Convert seconds to readable time format
144     def secs_to_readable_time(seconds):
145         m, s = divmod(seconds, 60)
146         h, m = divmod(m, 60)
147         d, h = divmod(h, 24)
148         y, d = divmod(d, 365)
149         print('Years: {}, Days: {}, Hours: {}, Mins: {}, Secs: {}'.format(y, d, h, m, s))
150
151     total_travel_time = df['Trip Duration'].sum()
152     print('Total travel time: %s seconds' % total_travel_time)
153     secs_to_readable_time(total_travel_time)
154
155     # TO DO: display mean travel time
156     mean_travel_time = df['Trip Duration'].mean()
157     print('Mean travel time: %s seconds' % mean_travel_time)
158     secs_to_readable_time(mean_travel_time)
159
160     print("\nThis took %s seconds." % (time.time() - start_time))
161     print('\n\n')
162
163 def user_stats(df):
164     """Displays statistics on bikeshare users."""
165
166     print('\nCalculating User Stats...\n')
167     start_time = time.time()
168
169     # TO DO: Display counts of user types
170     user_types = df['User Type'].value_counts()
171     print(user_types)
172
173     # TO DO: Display counts of gender
174     if 'Gender' in df.columns:
175         gender_count = df['Gender'].value_counts()
176         print(gender_count)
177
178     # TO DO: Display earliest, most recent, and most common year of birth
179     if 'Birth Year' in df.columns:
180         earliest_birth_year = df['Birth Year'].min()
181         most_recent_birth_year = df['Birth Year'].max()
182         common_birth_year = df['Birth Year'].mode()[0]
183         print('Earliest year of birth: ' + str(earliest_birth_year))
184         print('Most recent year of birth: ' + str(most_recent_birth_year))
185         print('Most common year of birth: ' + str(common_birth_year))
186
187     print("\nThis took %s seconds." % (time.time() - start_time))
188     print('\n\n')
189
190 def raw_data(df):
191     user_input = input('Do you want to see raw data? Enter yes or no.\n')
192     line_number = 0
193     while 1 == 1 :
194         if user_input.lower() != 'no':
195             print(df.iloc[line_number : line_number + 5])
196             line_number += 5
197             user_input = input('\nDo you want to see more raw data? Enter yes or no.\n')
198         else:
199             break
200
201 def main():
202     while 1 == 1 :
203         city, month, day = get_filters()
204         df = load_data(city, month, day)
205
206         time_stats(df)
207         station_stats(df)
208         trip_duration_stats(df)
209         user_stats(df)
210         raw_data(df)
211         restart = input('\nWould you like to restart? Enter yes or no.\n')
212         if restart.lower() != 'yes':
213             break
214
215 if __name__ == "__main__":
216     main()
```

AWESOME

Thanks for turning the duration into a readable format!

```
152     # TO DO: display total travel time
153     total_travel_time = df['Trip Duration'].sum()
154     print('Total travel time: %s seconds' % total_travel_time)
155     secs_to_readable_time(total_travel_time)
156
157     # TO DO: display mean travel time
158     mean_travel_time = df['Trip Duration'].mean()
159     print('Mean travel time: %s seconds' % mean_travel_time)
160     secs_to_readable_time(mean_travel_time)
161
162     print("\nThis took %s seconds." % (time.time() - start_time))
163     print('\n\n')
164
165 def user_stats(df):
166     """Displays statistics on bikeshare users."""
167
168     print('\nCalculating User Stats...\n')
169     start_time = time.time()
170
171     # TO DO: Display counts of user types
172     user_types = df['User Type'].value_counts()
173     print(user_types)
174
175     # TO DO: Display counts of gender
176     if 'Gender' in df.columns:
177         gender_count = df['Gender'].value_counts()
178         print(gender_count)
179
180     # TO DO: Display earliest, most recent, and most common year of birth
181     if 'Birth Year' in df.columns:
```

AWESOME

Perfect way to filter for Washington data!

```
183     earliest_birth_year = df['Birth Year'].min()
184     most_recent_birth_year = df['Birth Year'].max()
185     common_birth_year = df['Birth Year'].mode()[0]
186     print('Earliest year of birth: ' + str(earliest_birth_year))
187     print('Most recent year of birth: ' + str(most_recent_birth_year))
188     print('Most common year of birth: ' + str(common_birth_year))
189
190     print("\nThis took %s seconds." % (time.time() - start_time))
191     print('\n\n')
192
193 def raw_data(df):
194     user_input = input('Do you want to see raw data? Enter yes or no.\n')
195     line_number = 0
196     while 1 == 1 :
197         if user_input.lower() != 'no':
198             print(df.iloc[line_number : line_number + 5])
199             line_number += 5
200             user_input = input('\nDo you want to see more raw data? Enter yes or no.\n')
201         else:
202             break
203
204 def main():
205     while 1 == 1 :
206         city, month, day = get_filters()
207         df = load_data(city, month, day)
208
209         time_stats(df)
210         station_stats(df)
211         trip_duration_stats(df)
212         user_stats(df)
213         raw_data(df)
214         restart = input('\nWould you like to restart? Enter yes or no.\n')
215         if restart.lower() != 'yes':
216             break
217
218 if __name__ == "__main__":
219     main()
```

RETURN TO PATH