Explore US Bikeshare Data

REVIEW | CODE REVIEW 6 | HISTORY

home/bikeshare.py 6

```
import time
import pandas as pd
import numpy as np

CITY_DATA = { 'chicago': 'chicago.csv',
              'new york': 'new_york_city.csv',
              'washington': 'washington.csv' }

def get_filters():
    """
    Asks user to specify a city, month, and day to analyze.
    Returns:
        (str) city - name of the city to analyze
        (str) month - name of the month to filter by, or "all" to apply no month filter
        (str) day - name of the day of week to filter by, or "all" to apply no day filter
    """

    invalid_inputs = "Invalid input. Please try again with the mentioned city names"

    print('Hey there! Let\'s now explore some US bikeshare data!')
    # TO DO: get user raw_input for city (chicago, new york city, washington). HINT: Use a while loop to handle i
    while 1 == 1 :
```

SUGGESTION

Convention is `while True`

```
        city = input("\nEnter the name of the city to analyze,City names are as follows:\nchicago,new york,washin
        if city in ['chicago', 'new york', 'washington']:
            break
        else:
            print(invalid_inputs)

    # TO DO: get user raw_input for month (all, january, february, ... , june)
    while 1 == 1 :
        month = input("\n Enter the month for which you want to analyze:\njanuary,\nfebruary,\nmarch,"
                      "\napril,\nmay,\njune\nto filter by, or \"all\" to apply no month filter\n").lower()
        if month in ["january", "february", "march", "april", "may", "june", "all"]:
            break
        else:
            print(invalid_inputs)

    # TO DO: get user raw_input for day of week (all, monday, tuesday, ... sunday)
    while 1 == 1 :
        day = input("\nEnter the day for which you want the data \nMonday,\nTuesday,\nWednesday,\nThursday,"
                    "\nFriday,\nSaturday,\nSunday\nof week to filter by, or \"all\" to apply no day filter\n").lower()
        if day in ["monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday", "all"]:
            break
        else:
            print(invalid_inputs)

    print('-'*40)
    return city, month, day


def load_data(city, month, day):
    """
    Loads data for the specified city and filters by month and day if applicable.
    Args:
        (str) city - name of the city to analyze
        (str) month - name of the month to filter by, or "all" to apply no month filter
        (str) day - name of the day of week to filter by, or "all" to apply no day filter
    Returns:
        df - Pandas DataFrame containing city data filtered by month and day
    """
    file_name = CITY_DATA[city]
    print ('Accessing data from: ' + file_name)
    df = pd.read_csv(file_name)

    # convert the Start Time column to datetime
    df['Start Time'] = pd.to_datetime(arg = df['Start Time'], format = '%Y-%m-%d %H:%M:%S')

    # filter by month if applicable
    if month != 'all':
        # extract month and day of week from Start Time to create new columns
        df['month'] = df['Start Time'].dt.month

        # use the index of the months list to get the corresponding int
        months = ['january', 'february', 'march', 'april', 'may', 'june']
        month = months.index(month) + 1

        # filter by month to create the new dataframe
        df = df.loc[df['month'] == month]

    # filter by day of week if applicable
    if day != 'all':
        df['day_of_week'] = df['Start Time'].dt.weekday_name

        # filter by day of week to create the new dataframe
        df = df.loc[df['day_of_week'] == day.title()]
    return df


def time_stats(df):
    """Displays statistics on the most frequent times of travel."""

    print('\nMost Frequent Times of Travel...\n')
    start_time = time.time()

    # Convert the Start Time column to datetime
    df['Start Time'] = pd.to_datetime(arg = df['Start Time'], format = '%Y-%m-%d %H:%M:%S')

    # Create new columns for month, weekday, hour
    month = df['Start Time'].dt.month
    weekday_name = df['Start Time'].dt.weekday_name
    hour = df['Start Time'].dt.hour

    # TO DO: display the most common month
    most_common_month = month.mode()[0]
    print('Most common month: ', most_common_month)

    # TO DO: display the most common day of week
    most_common_day_of_week = weekday_name.mode()[0]
    print('Most common day of week: ', most_common_day_of_week)

    # TO DO: display the most common start hour
    common_start_hour = hour.mode()[0]
    print('Most frequent start hour: ', common_start_hour)

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)


def station_stats(df):
    """Displays statistics on the most popular stations and trip."""

    print('\nMost Popular Stations and Trip...\n')
    start_time = time.time()

    # TO DO: display most commonly used start station
    print('Most commonly used start station:', df['Start Station'].value_counts().idxmax())

    # TO DO: display most commonly used end station
    print('Most commonly used end station:', df['End Station'].value_counts().idxmax())

    # TO DO: display most frequent combination of start station and end station trip
    combine_stations = df['Start Station'] + "*" + df['End Station']
    common_station = combine_stations.value_counts().idxmax()
    print('Most frequent trips are:\n{} \nto\n{}'.format(common_station.split('*')[0], common_station.split('*')[

    print('-'*40)


def trip_duration_stats(df):
    """Displays statistics on the total and average trip duration."""

    print('\nCalculating Trip Duration...\n')
    start_time = time.time()
    # convert seconds to readable time format
    def secs_to_readable_time(seconds):
        m, s = divmod(seconds,60)
        h, m = divmod(m,60)
        d, h = divmod(h,24)
        y, d = divmod(d,365)
        print('Years: {}, Days: {}, Hours: {}, Mins: {}, Secs: {}'.format(y,d,h,m,s))
```

AWESOME!

Thanks for turning the duration into a readable format!

```
    # TO DO: display total travel time
    total_travel_time = df['Trip Duration'].sum()
    print('Total travel time:\n')
    secs_to_readable_time(total_travel_time)

    # TO DO: display mean travel time
    mean_travel_time = df['Trip Duration'].mean()
    print('\nMean travel time: {} seconds'.format(mean_travel_time))

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)


def user_stats(df):
    """Displays statistics on bikeshare users."""

    print('\nCalculating User Stats...\n')
    start_time = time.time()

    # TO DO: Display counts of user types
    user_types = df['User Type'].value_counts()
    print(user_types)

    # TO DO: Display counts of gender
    if 'Gender' in df.columns:
        gender_count = df['Gender'].value_counts()
        print(gender_count)

    # TO DO: Display earliest, most recent, and most common year of birth
    if 'Birth Year' in df.columns:
```

AWESOME!

Perfect way to filter for Washington data!

```
        earliest_birth_year = df['Birth Year'].min()
        most_recent_birth_year = df['Birth Year'].max()
        common_birth_year = df['Birth Year'].mode()[0]
        print("\nEarliest year of birth: " + str(earliest_birth_year))
        print("\nMost recent year of birth: " + str(most_recent_birth_year))
        print("\nMost common year of birth: " + str(common_birth_year))

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)

def raw_data(df):
    user_input = input('Do you want to see raw data? Enter yes or no.\n')
    line_number = 0

    while 1 == 1:
        if user_input.lower() != 'no':
            print(df.iloc[line_number : line_number + 5])
            line_number += 5
            user_input = input('\nDo you want to see more raw data? Enter yes or no.\n')
        else:
            break

def main():
    while 1 == 1 :
        city, month, day = get_filters()
        df = load_data(city, month, day)

        time_stats(df)
        station_stats(df)
        trip_duration_stats(df)
        user_stats(df)
        raw_data(df)
        restart = input('\nWould you like to restart? Enter yes or no.\n')
        if restart.lower() != 'yes':
            break


if __name__ == "__main__":
    main()
```

RETURN TO PATH