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

CITY_DATA = { 'Chicago': 'chicago.csv',
              'New York City': '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
    """

    print('\nHello! Let\'s explore some US bikeshare data!')
    # TO DO: get user input for city (chicago, new york city, washington). HINT: Use a while loop to handle invalid inputs



    while True:
      city = input("Write a city name: Chicago, New York City or Washington!").lower()
      if city not in ('New York City', 'Chicago', 'Washington'):
        print("Sorry, I didn't catch that. Try again.")
        continue
      else:
        break

    # TO DO: get user input for month (all, january, february, ... , june)
```

```
 while True:
    time = input("Do you want to filter as month, day, all or none?").lower()
    if time == 'month':
        month = input("Which month? January, Feburary, March, April, May or June?").lower()
        break

    elif time == 'day':
        day = input("Which day? Monday, Tuesday, Wednesday, Thursday, Friday, Saturday or
Sunday").lower()
        break

    elif time == 'all':
        month = input("Which month? January, Feburary, March, April, May or June?").lower()
        day = input("Which day? Monday, Tuesday, Wednesday, Thursday, Friday, Saturday or
Sunday").lower()
        break
    elif time == 'none':
        break
    else:
        input("You wrote the wrong word! Please type it again. month, day, all or none?")
        break

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




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
"""
# load data file into a dataframe
df = pd.read_csv(CITY_DATA[city])

# convert the Start Time column to datetime
df['Start Time'] = pd.to_datetime(df['Start Time'])

# extract month and day of week from Start Time to create new columns

df['month'] = df['Start Time'].dt.month
df['day_of_week'] = df['Start Time'].dt.weekday_name

# filter by month if applicable
if month != 'all':
                    # 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[df['month'] == month]

    # filter by day of week if applicable
if day != 'all':
    # filter by day of week to create the new dataframe
```

```
        df = df[df['day_of_week'] == day.title()]

    return df

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

    print('\nCalculating The Most Frequent Times of Travel...\n')
    start_time = time.time()

    # TO DO: display the most common month

   df['month'] = df['Start Time'].dt.month
   common_month = df['month'].mode()[0]
   print(common_month)



    # TO DO: display the most common day of week

    df['day_of_week'] = df['Start Time'].dt.week
    common_day_of_week = df['day_of_week'].mode()[0]
    print(common_day_of_week)

    # TO DO: display the most common start hour

    df['hour'] = df['Start Time'].dt.hour
    common_hour = df['hour'].mode()[0]
    print(common_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('\nCalculating The Most Popular Stations and Trip...\n')
    start_time = time.time()

    # TO DO: display most commonly used start station

    common_start = df['Start Station'].mode()[0]
    print(common_start)

    # TO DO: display most commonly used end station

    common_end = df['End Station'].mode()[0]
    print(common_end)



    # TO DO: display most frequent combination of start station and end station trip

    df['combination'] = df['Start Station'] + ' to ' + df['End Station']
    common_combination = df['combination'].mode()[0]
    print(common_combination)

    print("\nThis took %s seconds." % (time.time() - start_time))
    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()
```

```
    # TO DO: display total travel time

    total_travel = df['Trip Duration'].sum()
    print(total_travel)


    # TO DO: display mean travel time

    mean_travel = df['Trip Duration'].mean()
    print(mean_travel)


    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)
    print(user_types)

    # TO DO: Display counts of gender

    if 'Gender' in df:
```

```
gender = df['Gender'].value_counts()
print(gender)
else:
print("There is no gender information in this city.")


# TO DO: Display earliest, most recent, and most common year of birth

if 'Birth_Year' in df:
earliest = df['Birth_Year'].min()
print(earliest)
recent = df['Birth_Year'].max()
print(recent)
common_birth = df['Birth Year'].mode()[0]
print(common_birth)
else:
print("There is no birth year information in this city.")

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

def data(df):
raw_data = 0
while True:
answer = input("Do you want to see the raw data? Yes or No").lower()
if answer not in ['yes', 'no']:
answer = input("You wrote the wrong word. Please type Yes or No.").lower()
elif answer == 'yes':
raw_data += 5
print(df.iloc[raw_data : raw_data + 5])
again = input("Do you want to see more? Yes or No").lower()
if again == 'no':
```

```
    break
    elif answer == 'no':
    return
    def main():
    city = ""
    month = ""
    day = ""
    while True:
    city, month, day = get_filters(city, month, day)
    df = load_data(city, month, day)

    time_stats(df)
    station_stats(df)
    trip_duration_stats(df)
    user_stats(df)

    restart = input('\nWould you like to restart? Enter yes or no.\n')
    if restart.lower() != 'yes':
       break


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