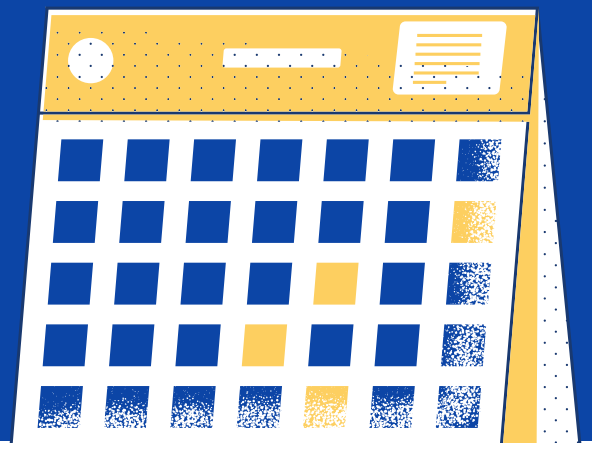
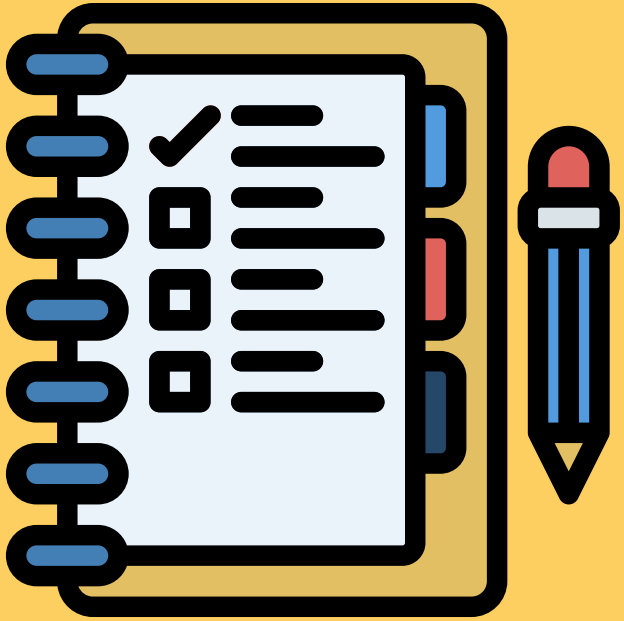
Data Fundamentals

Explore US Bikeshare Data

GOALS



- **Project Aim**: Analyze and compare US bikeshare data from Chicago, New York City, and Washington using Python, pandas, and NumPy.
- **Coding Goals**: Create an interactive terminal experience, handle user inputs robustly, display raw data upon request, and improve code efficiency.
- **Data Exploration**: Identify dataset columns, inspect for missing values, and understand each column's value types using pandas methods.
- **Submission Requirements**: Deliver high-quality, error-free code following the project rubric, and submit in either the provided workspace or locally after downloading project files.

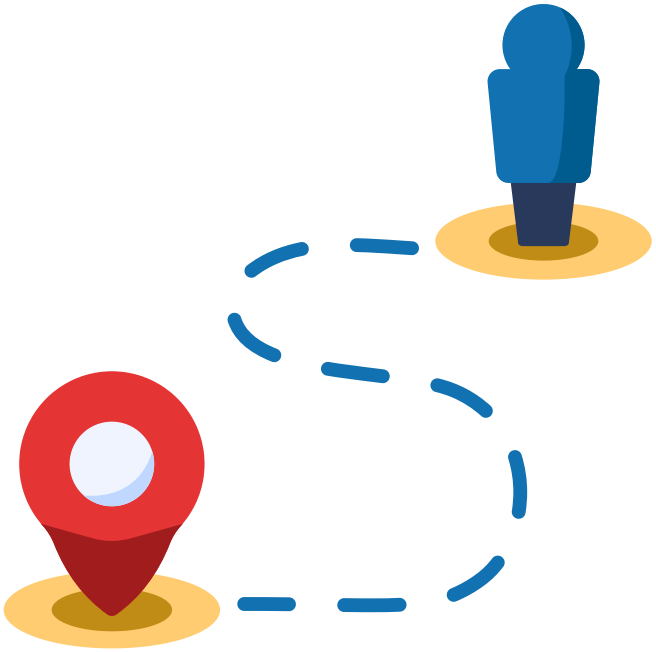


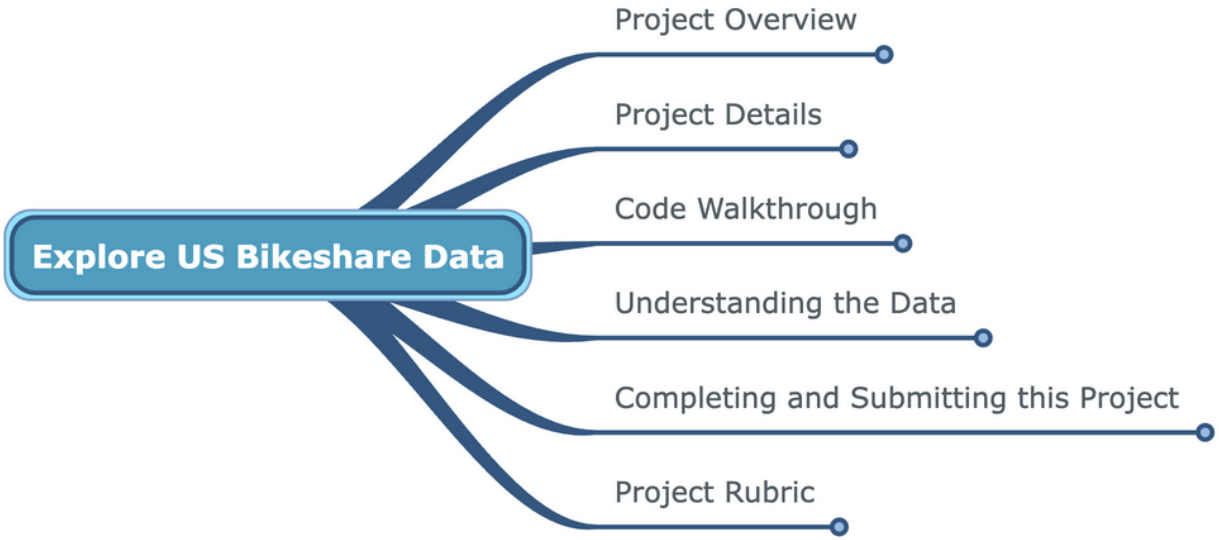
AGENDA

- Welcome
- Project Overview
- Project Details
- Code Walkthrough
- Understanding the Data
- Completing and Submitting the Project
- Project Rubric
- Wrap Up and Next Steps
- Q&A

Behind every data point, there's a story waiting to be told.

ROADMAP




Explore US Bikeshare Data
Project Overview
Project Details
Code Walkthrough
Understanding the Data
Completing and Submitting this Project
Project Rubric

ITERATORS AND GENERATORS

ITERATORS

Iterator is an object following the iterator protocol with **__iter__()** (returns the iterator itself) and **__next__()** methods (yields next value). Iteration stops when all items are consumed, triggering a **StopIteration** exception.

```python
class MyIterator:
    def __init__(self, max_val):
        self.max_val = max_val
        self.cur_val = 0

    def __iter__(self):
        return self

    def __next__(self):
        if self.cur_val < self.max_val:
            val = self.cur_val
            self.cur_val += 1
            return val
        else:
            raise StopIteration

my_iter = MyIterator(5)

for i in my_iter:
    print(i)  # Will print numbers 0 through 4
```

GENERATORS

Generators are functions that yield a series of results. When called, they return a generator object. Using **next()**, they execute until hitting a **yield**, which pauses execution and saves the state for continuation, unlike **return** which concludes the function.

```
def my_generator(max_val):
    num = 0
    while num < max_val:
        yield num
        num += 1

gen = my_generator(5)

for i in gen:
    print(i)  # Will print numbers 0 through 4
```

EXPLORE US BIKESHARE DATA

PROJECT OVERVIEW

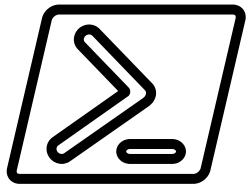
SOFTWARE SETUP

Python3.x

Libraries

Text Editor

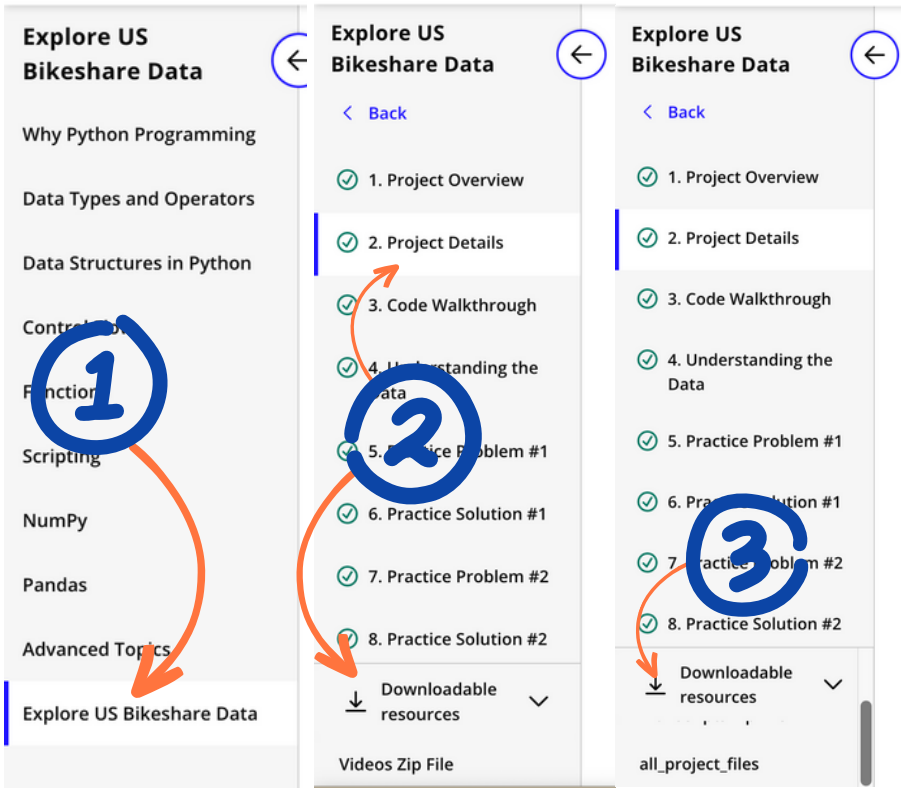
Terminal



Anaconda is an open-source distribution of Python and R, tailored for data science and machine learning tasks, providing package management, Jupyter Notebooks, and popular libraries in one package.

DATA COLLECTION

Gather data related to bike share systems for the cities of Chicago, New York City, and Washington.



DATA IMPORT

Write Python code to import the gathered data using pandas library.

```python
# Import the pandas library
import pandas as pd

# Absolute file path

# For Windows, it might look like this:
# r is used to make it a raw string
absolute_path = r'C:\Users\Username\Documents\my_file.csv'

# For Mac/Linux, it might look like this:
# absolute_path = '/Users/username/Documents/my_file.csv'

# Relative file path
# This assumes that the file is in the same directory as your Python script
relative_path = 'my_file.csv'

# Attempt to load the data from the absolute file path
try:
    df = pd.read_csv(absolute_path)
    print("Loaded data from absolute file path.")
except FileNotFoundError:
    # If that fails, attempt to load the data from the relative file path
    try:
        df = pd.read_csv(relative_path)
        print("Loaded data from relative file path.")
    except FileNotFoundError:
        print("File not found in either location.")
```

```python
# Import the pandas library
import pandas as pd

# List of file names
file_names = ['data1.csv', 'data2.csv', 'data3.csv']

# Base absolute path
# For Windows, it might look like this:
# base_absolute_path = r'C:\Users\Username\Documents'

# For Mac/Linux, it might look like this:
base_absolute_path = '/Users/username/Documents'

# Empty dictionary to store dataframes
dfs = {}

for file_name in file_names:
    # Create absolute path
    absolute_path = f"{base_absolute_path}/{file_name}"

    # Attempt to load the data from the absolute file path
    try:
        df = pd.read_csv(absolute_path)
        print(f"Loaded data from absolute file path for {file_name}.")
        dfs[file_name] = df
    except FileNotFoundError:
        # If that fails, attempt to load the data from the relative file path
        try:
            df = pd.read_csv(file_name)
            print(f"Loaded data from relative file path for {file_name}.")
            dfs[file_name] = df
        except FileNotFoundError:
            print(f"File {file_name} not found in either location.")

# Display the first 5 rows of each DataFrame
for file_name, df in dfs.items():
    print(f"First 5 rows for {file_name}:")
    print(df.head())
```

DATA EXPLORATION

Explore the imported data by computing various descriptive statistics like common travel times, popular stations, trip durations, and user information.

```
'''
Assuming you have a CSV data file (let's say named data.csv) with columns
for start_station, end_station, travel_time, trip_duration, and user_info
'''
import pandas as pd

# Load the CSV data into a pandas DataFrame
df = pd.read_csv('data.csv')

# Compute common travel times
common_travel_times = df['travel_time'].mode()[0]
print(f'Common travel times: {common_travel_times}')

# Find popular stations
popular_start_station = df['start_station'].mode()[0]
popular_end_station = df['end_station'].mode()[0]
print(f'Most popular start station: {popular_start_station}')
print(f'Most popular end station: {popular_end_station}')

# Compute average trip duration
average_trip_duration = df['trip_duration'].mean()
print(f'Average trip duration: {average_trip_duration} minutes')

# If 'user_info' is a categorical variable (e.g., 'registered' vs 'casual')
# We can compute the frequency of each category
user_info_counts = df['user_info'].value_counts()
print('User information counts:\n', user_info_counts)
```

INTERACTIVE TERMINAL EXPERIENCE

Create a script that generates an interactive terminal experience, taking user input and presenting corresponding computed statistics.

```python
# This function prompts the user to enter five numbers and returns them as a list.
def collect_numbers():
    nums = []
    for i in range(5):
        while True:
            try:
                num = float(input(f"Enter number {i+1}: "))
                nums.append(num)
                break
            except ValueError:
                print("Invalid input. Please enter a number.")
    return nums

# This function takes a list of numbers and returns their average.
def calculate_average(nums):
    return sum(nums) / len(nums)

# Main function that collects five numbers from the user, calculates their average, and prints the result.
def main():
    nums = collect_numbers()
    avg = calculate_average(nums)
    print(f"The average of your numbers is: {avg}")

# If this script is run directly (not imported as a module), the main function is called.
if __name__ == "__main__":
    main()
```

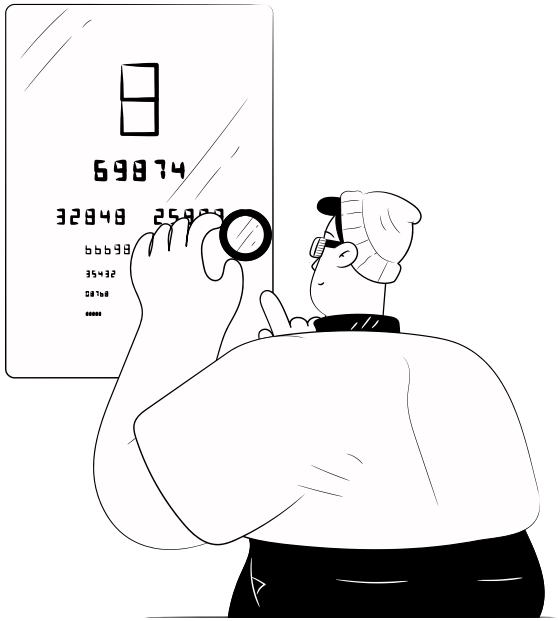
CODE DOCUMENTATION

Thoroughly document your code with relevant comments.

```python
def collect_numbers():
    """
    Prompt the user to input five numbers.

    Returns:
    list: A list containing five numbers entered by the user.
    """
    nums = []  # Initialize an empty list to store numbers

    # Loop five times to collect five numbers
    for i in range(5):
        # Continue looping until a valid number is input
        while True:
            try:
                # Prompt user to input a number
                num = float(input(f"Enter number {i+1}: "))
                # Add the entered number to the list
                nums.append(num)
                break
            except ValueError:  # Catch exception if the input is not a number
                print("Invalid input. Please enter a number.")
    return nums

def calculate_average(nums):
    """
    Calculate the average of a list of numbers.

    Args:
    nums (list): A list of numbers.

    Returns:
    float: The average of the numbers in the input list.
    """
    # Calculate and return the average of the numbers
    return sum(nums) / len(nums)

def main():
    """
    The main function of the script. It prompts the user to enter five numbers, calculates their average, and prints the result.
    """
    nums = collect_numbers()  # Call function to collect numbers from user
    avg = calculate_average(nums)  # Call function to calculate average of numbers
    print(f"The average of your numbers is: {avg}")  # Print the average

if __name__ == "__main__":
    main()  # If script is run directly, call the main function
```

CODE TESTING

Run extensive tests on your code to ensure it works as expected, even for edge cases and error handling.



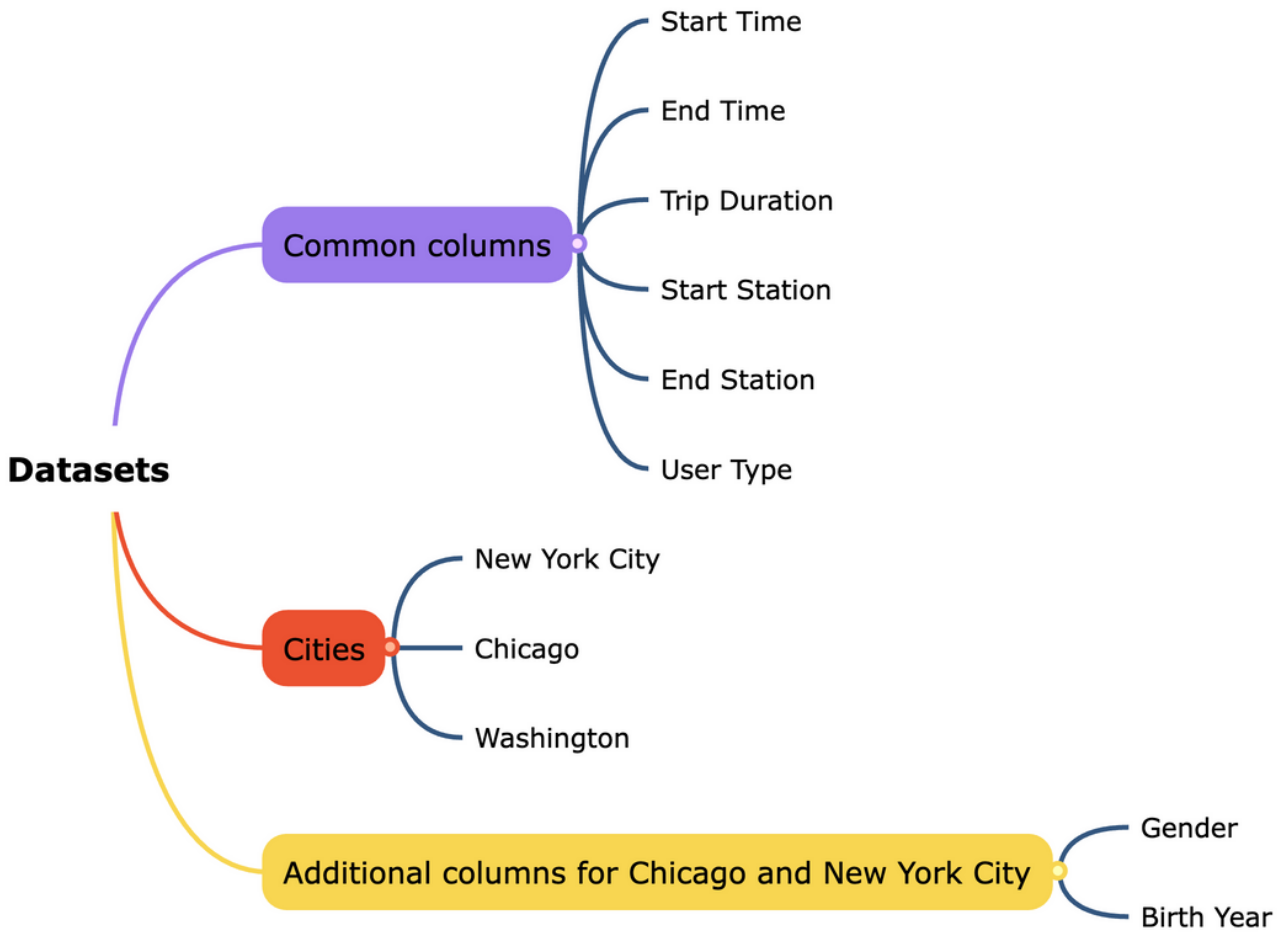
PROJECT SUBMISSION

Submit your final project, ensuring all required files are included and your code adheres to the project rubric.

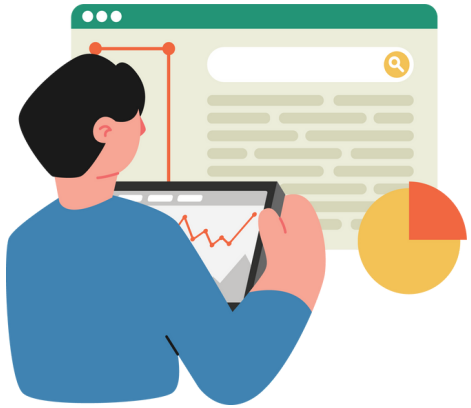
PROJECT DETAILS

DATASETS

The dataset comprises data from the first half of 2017 from the three cities.



STATISTICS



1. **Popular times of travel:** most common month, day of the week, hour of the day.
2. **Popular stations and trips**: most common start and end stations, and most frequent combination of start station and end station.
3. **Trip duration**: total and average travel time.
4. **User info**: counts of each user type, each gender (only for NYC and Chicago), and statistics around birth years (only for NYC and Chicago).

PROJECT FILES

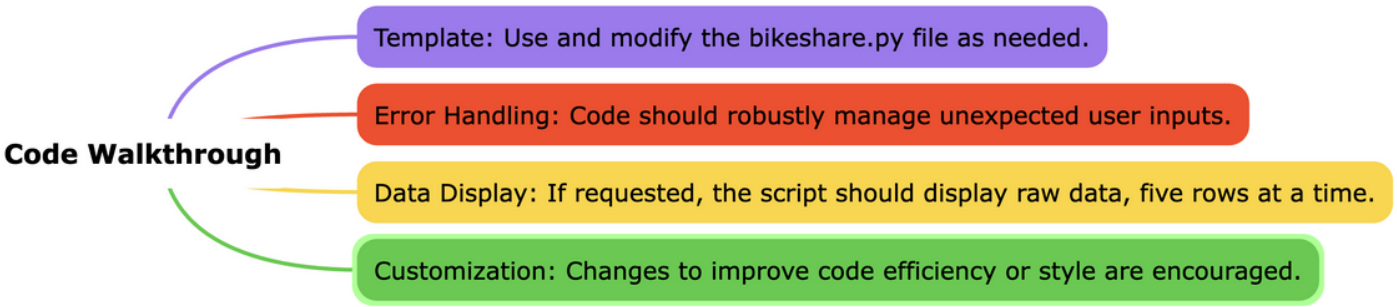
chicago.csv

new_york_city.csv

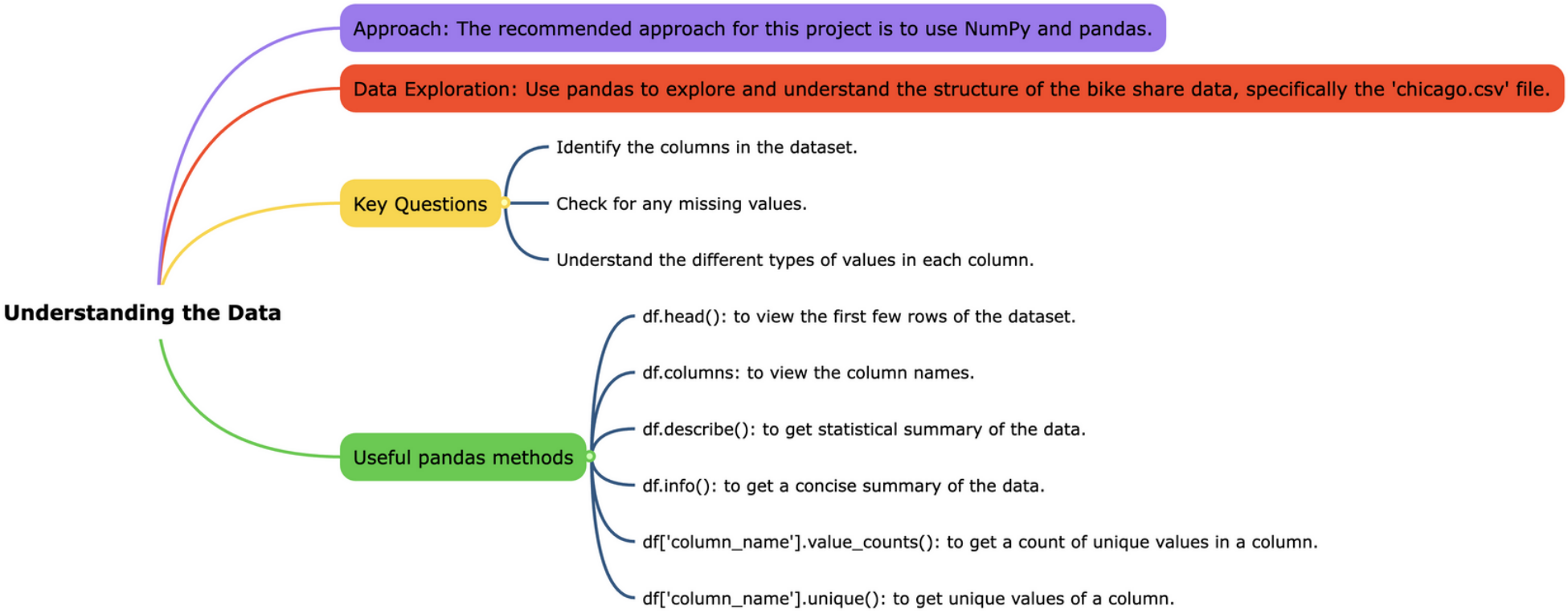
washington.csv

bikeshare_2.py

CODE WALKTHROUGH



UNDERSTANDING THE DATA

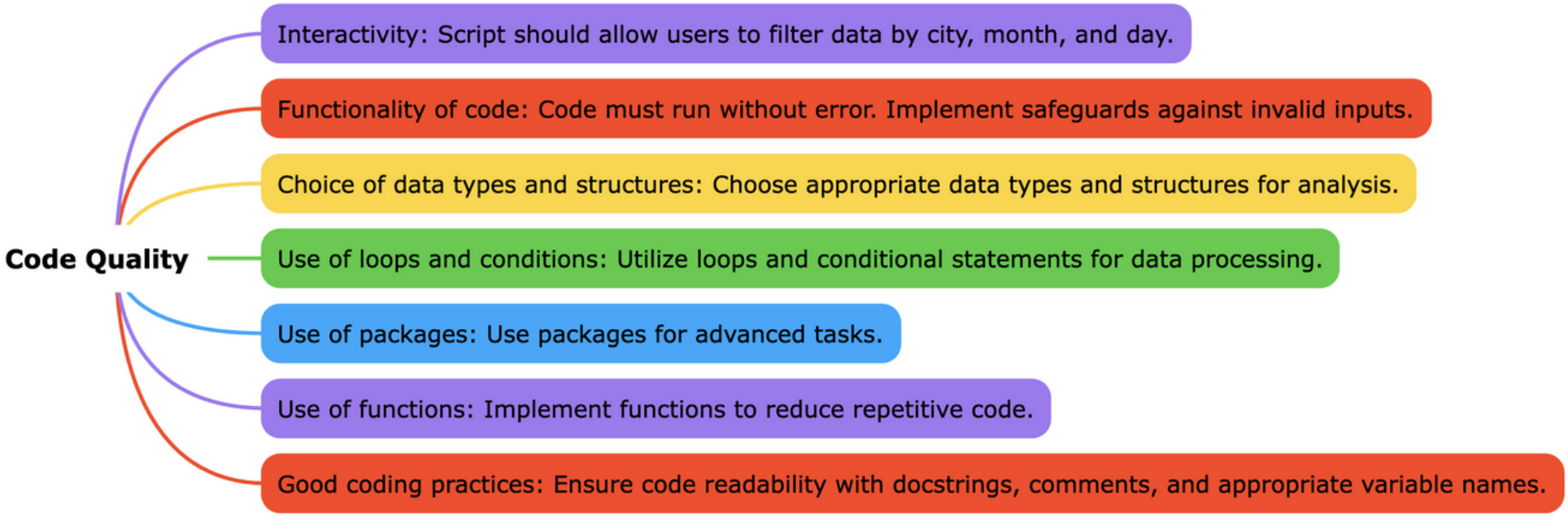


COMPLETING AND SUBMITTING THIS PROJECT

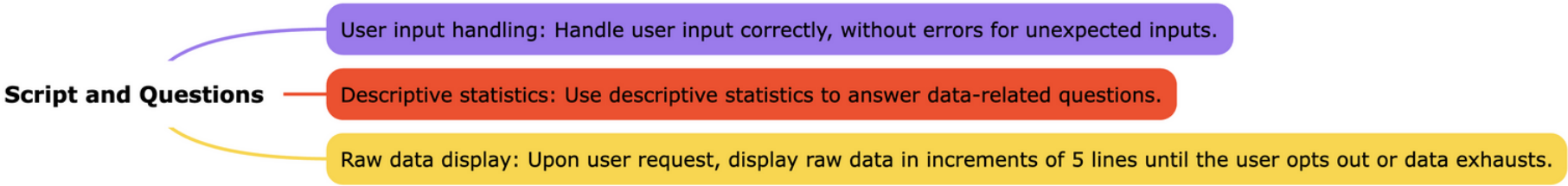
COMPLETING AND SUBMITTING THIS PROJECT

PROJECT RUBRIC

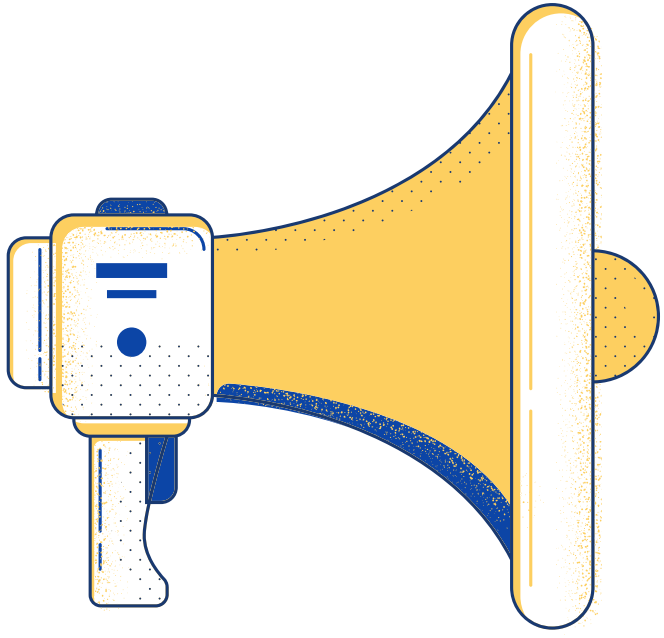
CODE QUALITY



SCRIPT AND QUESTIONS



SUGGESTIONS TO STAND OUT



Q&A Session: Let's explore and understand together

WE NEED YOUR FEEDBACK

Help us improve our sessions! Complete the **Oman Makeen Student Satisfaction Survey**.

Rate my competency, our engagement, and your overall satisfaction.

- Confidential and used solely for session improvement.

CLICK HERE

RESOURCES

- Explore US Bikeshare Data
- Beginners guide to streamlit
- Online Python
- Session-10 resources
- Python Installation & ENV 1
- Python Installation & ENV 2
- Python Installation & ENV 3
- Print Color

Your presence today has added value to our shared learning journey. Thank you for joining us!