

# Data Fundamentals



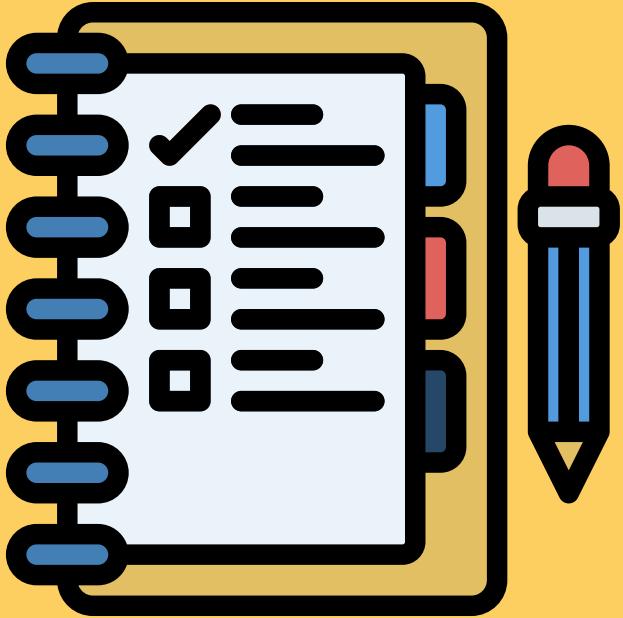
---

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

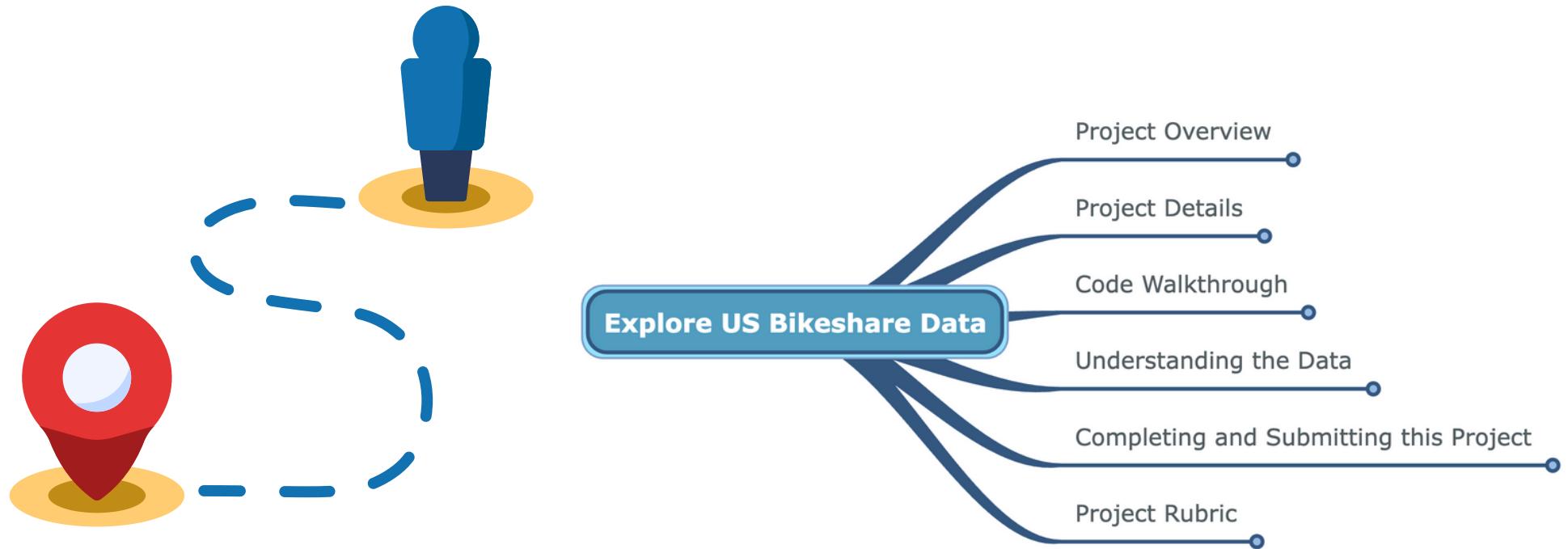


---

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

# ROADMAP

---



# 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.

```
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 BIKE SHARE DATA

---

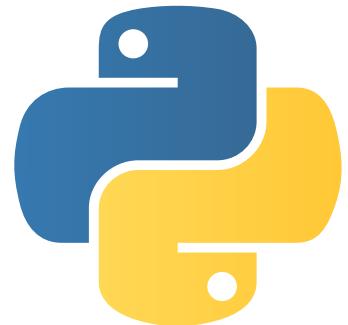
# PROJECT OVERVIEW

---

# SOFTWARE SETUP

---

Python3.x



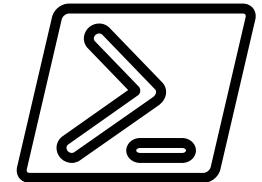
Libraries



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.

4

Explore US Bikeshare Data

←

Explore US Bikeshare Data

←

Explore US Bikeshare Data

←

Why Python Programming

Data Types and Operators

Data Structures in Python

Control Flow

Functions

Scripting

NumPy

Pandas

Advanced Topics

Explore US Bikeshare Data

1

2

3

4

5

6

7

8

Downloadable resources

Videos Zip File

1. Project Overview

2. Project Details

3. Code Walkthrough

4. Understanding the Data

5. Practice Problem #1

6. Practice Solution #1

7. Practice Problem #2

8. Practice Solution #2

Start	End
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	20
20	21
21	22
22	23
23	24
24	25
25	26
26	27
27	28
28	29
29	30
30	31
31	32
32	33
33	34
34	35
35	36
36	37
37	38
38	39
39	40
40	41
41	42
42	43
43	44
44	45
45	46
46	47
47	48
48	49
49	50
50	51
51	52
52	53
53	54
54	55
55	56
56	57
57	58
58	59
59	60
60	61
61	62
62	63
63	64
64	65
65	66
66	67
67	68
68	69
69	70
70	71
71	72
72	73
73	74
74	75
75	76
76	77
77	78
78	79
79	80
80	81
81	82
82	83
83	84
84	85
85	86
86	87
87	88
88	89
89	90
90	91
91	92
92	93
93	94
94	95
95	96
96	97
97	98
98	99
99	100
100	101
101	102
102	103
103	104
104	105
105	106
106	107
107	108
108	109
109	110
110	111
111	112
112	113
113	114
114	115
115	116
116	117
117	118
118	119
119	120
120	121
121	122
122	123
123	124
124	125
125	126
126	127
127	128
128	129
129	130
130	131
131	132
132	133
133	134
134	135
135	136
136	137
137	138
138	139
139	140
140	141
141	142
142	143
143	144
144	145
145	146
146	147
147	148
148	149
149	150
150	151
151	152
152	153
153	154
154	155
155	156
156	157
157	158
158	159
159	160
160	161
161	162
162	163
163	164
164	165
165	166
166	167
167	168
168	169
169	170
170	171
171	172
172	173
173	174
174	175
175	176
176	177
177	178
178	179
179	180
180	181
181	182
182	183
183	184
184	185
185	186
186	187
187	188
188	189
189	190
190	191
191	192
192	193
193	194
194	195
195	196
196	197
197	198
198	199
199	200
200	201
201	202
202	203
203	204
204	205
205	206
206	207
207	208
208	209
209	210
210	211
211	212
212	213
213	214
214	215
215	216
216	217
217	218
218	219
219	220
220	221
221	222
222	223
223	224
224	225
225	226
226	227
227	228
228	229
229	230
230	231
231	232
232	233
233	234
234	235
235	236
236	237
237	238
238	239
239	240
240	241
241	242
242	243
243	244
244	245
245	246
246	247
247	248
248	249
249	250
250	251
251	252
252	253
253	254
254	255
255	256
256	257
257	258
258	259
259	260
260	261
261	262
262	263
263	264
264	265
265	266
266	267
267	268
268	269
269	270
270	271
271	272
272	273
273	274
274	275
275	276
276	277
277	278
278	279
279	280
280	281
281	282
282	283
283	284
284	285
285	286
286	287
287	288
288	289
289	290
290	291
291	292
292	293
293	294
294	295
295	296
296	297
297	298
298	299
299	300
300	301
301	302
302	303
303	304
304	305
305	306
306	307
307	308
308	309
309	310
310	311
311	312
312	313
313	314
314	315
315	316
316	317
317	318
318	319
319	320
320	321
321	322
322	323
323	324
324	325
325	326
326	327
327	328
328	329
329	330
330	331
331	332
332	333
333	334
334	335
335	336
336	337
337	338
338	339
339	340
340	341
341	342
342	343
343	344
344	345
345	346
346	347
347	348
348	349
349	350
350	351
351	352
352	353
353	354
354	355
355	356
356	357
357	358
358	359
359	360
360	361
361	362
362	363
363	364
364	365
365	366
366	367
367	368
368	369
369	370
370	371
371	372
372	373
373	374
374	375
375	376
376	377
377	378
378	379
379	380
380	381
381	382
382	383
383	384
384	385
385	386
386	387
387	388
388	389
389	390
390	391
391	392
392	393
393	394
394	395
395	396
396	397
397	398
398	399
399	400
400	401
401	402
402	403
403	404
404	405
405	406
406	407
407	408
408	409
409	410
410	411
411	412
412	413
413	414
414	415
415	416
416	417
417	418
418	419
419	420
420	421
421	422
422	423
423	424
424	425
425	426
426	427
427	428
428	429
429	430
430	431
431	432
432	433
433	434
434	435
435	436
436	437
437	438
438	439
439	440
440	441
441	442
442	443
443	444
444	445
445	446
446	447
447	448
448	449
449	450
450	451
451	452
452	453
453	454
454	455
455	456
456	457
457	458
458	459
459	460
460	461
461	462
462	463
463	464
464	465
465	466
466	467
467	468
468	469
469	470
470	471
471	472
472	473
473	474
474	475
475	476
476	477
477	478
478	479
479	480
480	481
481	482
482	483
483	484
484	485
485	486
486	487
487	488
488	489
489	490
490	491
491	492
492	493
493	494
494	495
495	496
496	497
497	498
498	499
499	500
500	501
501	502
502	503
503	504
504	505
505	506
506	507
507	508
508	509
509	510
510	511
511	512
512	513
513	514
514	515
515	516
516	517
517	518
518	519
519	520
520	521
521	522
522	523
523	524
524	525
525	526
526	527
527	528
528	529
529	530
530	531
531	532
532	533
533	534
534	535
535	536
536	537
537	538
538	539
539	540
540	541
541	542
542	543
543	544
544	545
545	546
546	547
547	548
548	549
549	550
550	551
551	552
552	553
553	554
554	555
555	556
556	557
557	558
558	559
559	560
560	561
561	562
562	563
563	564
564</td	

# DATA IMPORT

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

```
# 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.)")
```

```
# 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.

```
# 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.

```
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)

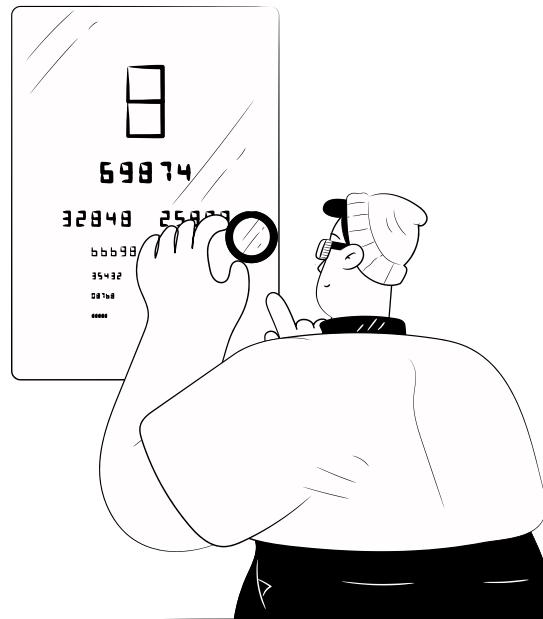
def main():
    """
    The main function of the script. It prompts the user to enter five numbers, calculates their average, and prints the result.
    """
    nums = collect_numbers() # Call function to collect numbers from user
    avg = calculate_average(nums) # Call function to calculate average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

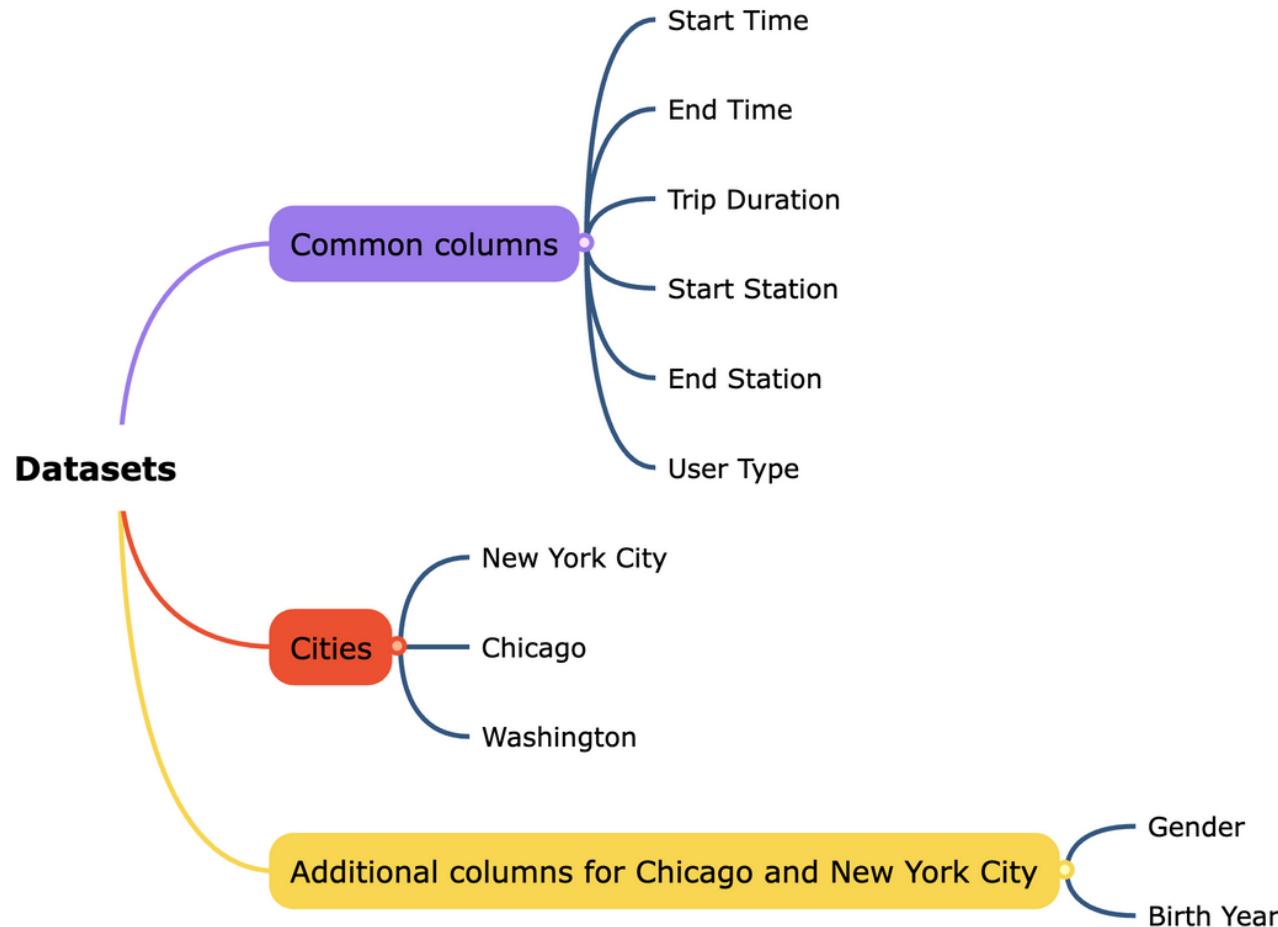


# PROJECT DETAILS

---

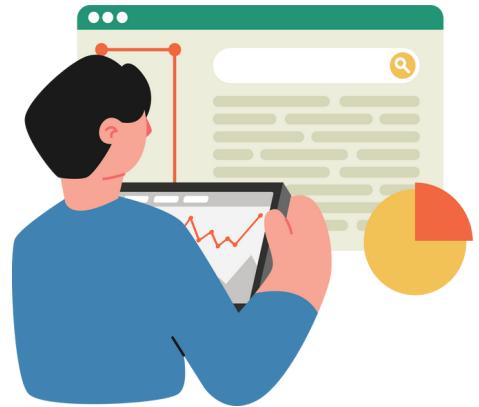
# DATASETS

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

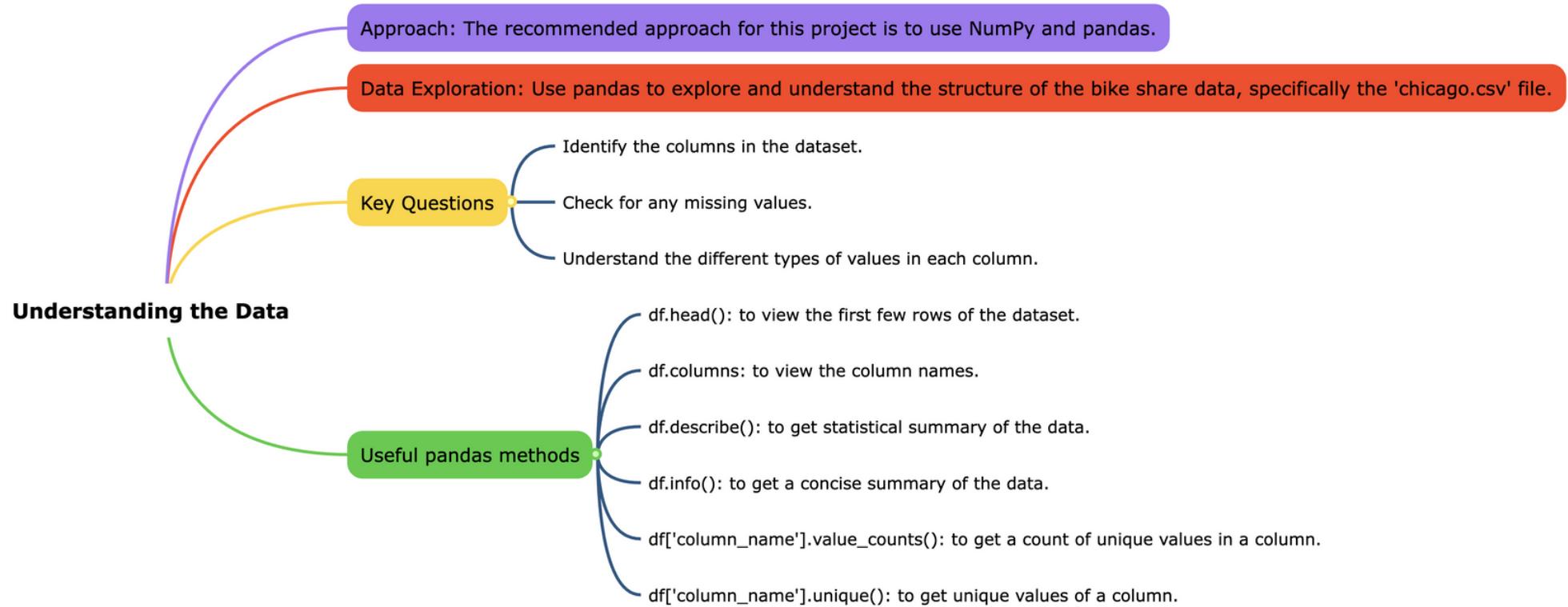
---

## Code Walkthrough

- Template: Use and modify the bikeshare.py file as needed.
- Error Handling: Code should robustly manage unexpected user inputs.
- Data Display: If requested, the script should display raw data, five rows at a time.
- Customization: Changes to improve code efficiency or style are encouraged.

# UNDERSTANDING THE DATA

---



# COMPLETING AND SUBMITTING THIS PROJECT

---

## Completing and Submitting this Project

Option 1: Work and submit directly in the provided workspace environment.

Option 2: Download project files, work locally, and upload project files for submission.

# COMPLETING AND SUBMITTING THIS PROJECT

---

## Completing and Submitting this Project

Option 1: Work and submit directly in the provided workspace environment.

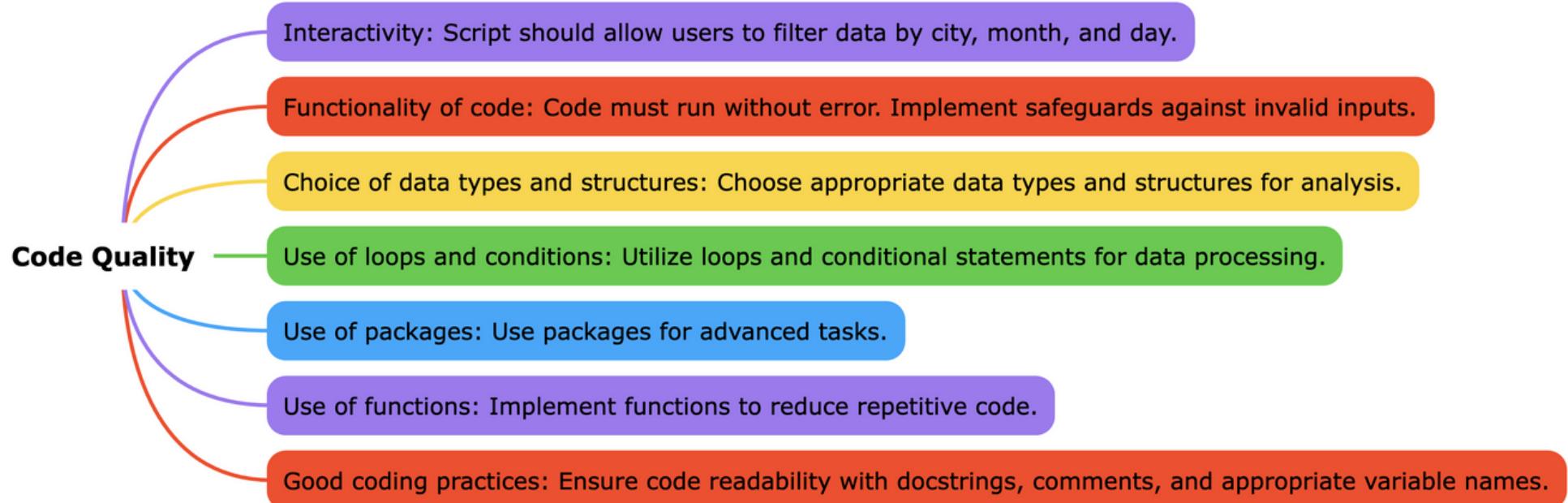
Option 2: Download project files, work locally, and upload project files for submission.

# PROJECT RUBRIC

---

# CODE QUALITY

---



# SCRIPT AND QUESTIONS

---

## Script and Questions

User input handling: Handle user input correctly, without errors for unexpected inputs.

Descriptive statistics: Use descriptive statistics to answer data-related questions.

Raw data display: Upon user request, display raw data in increments of 5 lines until the user opts out or data exhausts.

# SUGGESTIONS TO STAND OUT

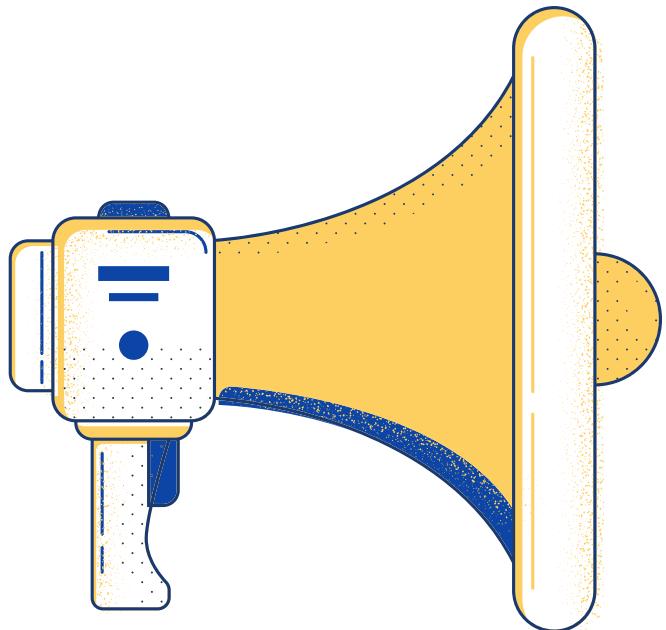
---

## Suggestions to Stand Out

Enhance the structure of bikeshare.py for better efficiency or style.

Ask and answer more questions about the data.

Improve interactive experience with added features like images, web app functionality, etc. Include clear instructions if needed.



**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!