**Week 2 Python Project: Object-Oriented Programming, Functions, and CLI with Git Collaboration**

**Introduction**

Welcome to Week 2!  
This week we will dive deeper into **Object-Oriented Programming (OOP)**, **writing reusable functions**, and building **Command-Line Interfaces (CLI)** using the argparse/click, and logging libraries. You will also learn how to effectively use **Git** for version control and collaboration on GitHub.  
The focus this week will be on writing Python classes, transforming data using custom functions, and implementing basic command-line functionality.

**PART A: Video Links**

Here are some helpful links that will introduce or reinforce the concepts you'll be applying this week:

**[OOP in Python](https://youtu.be/JeznW_7DlB0?si=Q-C43tV0q0UOrUQl)**

**[Typing in Python](https://youtu.be/7Chd5gPHlDg?si=p2wvENuowP4HiaaT)**

**[Args and Kwargs](https://youtu.be/Vh__2V2tXUM?si=DGe-9RoAL4A91jOE)**

**[Functions in Python](https://youtu.be/89cGQjB5R4M?si=GRKeAfrEbbQjUo8f)**

**[Pandas apply() Function](https://youtu.be/DsjvCKxOdgI?si=Q4TTb-gBY5mAsCWo)**

**[argparse Library](https://youtu.be/OxpBMNalsDM?si=EwOr3nVY-h42zCmq)**

**[click Library](https://youtu.be/GnSKhetBa48?si=Nx9YV9ACbby5EwoU)**

**[Logging in Python](https://youtu.be/urrfJgHwIJA?si=ftBJ6HgvGU1YkKZO)**

**[Git Basics and GitHub Collaboration](https://youtu.be/USjZcfj8yxE?si=U41qmoEKj2EV6qwP)**

**PART B: Assignment Tasks and Directions**

This week, you will extend your TitanicCleaner class by creating custom functions, adding

command-line interface (CLI) capabilities using argparse and click, and implementing

logging. You will also explore the use of Git for version control and collaboration via GitHub.

Task 1: Object-Oriented Programming (OOP) and Writing Functions

1. Create a class for the Titanic data cleaner:

- Create a class named `TitanicCleaner` to encapsulate all your data cleaning functions.

- Add methods for loading data, filling in missing values, removing duplicates, and applying

transformations.

2. Add data transformation functions to your class:

- Write a method to bin the Age column into categories `<18`, `18-40`, `40-60`, and `60+`.

- Create a method to calculate Family Size\*\* by summing the `SibSp` and `Parch` columns. Store

the result in a new column called `FamilySize`.

- Write a method to map the Embarked column\*\* values (e.g., `S -> Southampton`, `C ->

Cherbourg`, `Q -> Queenstown`) and update the column accordingly.

Sample Directions:

- Create your `TitanicCleaner` class in a Python script.

- Define each of these functions as methods within the class.

- After cleaning, ensure that your methods return the modified DataFrame.

Task 2: Add Command-Line Interface (CLI) Features

1. Add CLI functionality using `argparse`:

- Implement `argparse` in your Python script to allow users to specify the Titanic data file location

from the command line.

- Define arguments for loading the file, specifying output locations, and running specific data

transformations.

**OR**

2. Add CLI functionality using `click`:

- Create an alternative CLI interface using the `click` library.

- Ensure the user can run data transformations through simple CLI commands like `--bin-age` or `--map-embarked`.

Sample Directions:

- Modify your Python script to accept command-line arguments using `argparse` or `click`.

- Test either CLI implementations to ensure that they work as expected.

Task 3: Add Logging to Your Script

1. Replace print statements with logging:

- Instead of using `print()` statements, use the `logging` library to log important steps like data

loading, missing value handling, and transformation processes.

- Set up your logger to log information to both the console and a file.

Sample Directions:

- Add logging for steps like loading the CSV, filling missing values, or dropping duplicates.

- Set the logging level to `INFO` or `DEBUG` for detailed output.

Task 4: Write 3 Functions for Data Transformation Using `apply()`

1. Create a function for binning the Age column:

- Use the `apply()` function to transform the `Age` column into bins `<18`, `18-40`, `40-60`, and

`60+`.

2. Create a function to calculate Family Size:

- Define a function that sums the `SibSp` and `Parch` columns to create a new column called

`FamilySize`.

3. Create a function to map Embarked column:

- Write a function that maps the values of the `Embarked` column (e.g., `S -> Southampton`, `C

-> Cherbourg`, `Q -> Queenstown`) and apply it using `apply()`.

Sample Directions:

- Add these transformation functions to your `TitanicCleaner` class and test them.

- Use the `apply()` method in Pandas to apply these functions to the corresponding DataFrame columns.

Task 5: Git Version Control and Collaboration

1. Stage and commit new changes to your local repo:

- After writing your Python code, use `git add` to stage your files.

- Commit your changes with `git commit -m "Added TitanicCleaner class with transformations"`.

2. Push your changes to GitHub:

- push your changes with `git push -u origin main`.

3. Branching for collaboration:

- Create a new branch with `git branch new-feature` and switch to it with `git checkout new-

feature`. Make changes on this branch by using **print statements instead of logger library** and

push this branch to GitHub

- What benefits do branches offer in collaborative development?

Sample Directions:

- After writing the code, stage your changes using `git add .`, then commit the changes with a

meaningful message.

- Push the project to a new repository on GitHub using `git push`.

**PART C - Submission Instructions**

You are expected to submit your assignment by the end of the week. Submission will be done via Google Forms. You

are encouraged to put your work online to help build your portfolio and show your learning

For Twitter Submission:

• Tag the official TDI page: @TDataImmersed

• Tag Annie @DabereNnamani

• Tag the project coordinator @The\_Jonathaan

* Tag @python
* Tag @JOloganj
* Use the hashtag TDI

For LinkedIn Submission:

• Tag the TDI page: @TheDataImmersed

* Tag Annie @AnneNnamani
* Tag @josephologunja

GitHub Submission Instructions:

For this assignment, you will be required to submit your work via GitHub. Please follow the instructions below to submit your project:

1. Use the same repository as last week.

1. Project Structure:

* In your repository, create a new folder for this project:

Titanic\_Data\_Scripting/

├── README.md

├── scripts/

│ └── titanic.py

├── data/

│ └── titanic.csv # (Optional - Kaggle link can be referenced here)

├── notebooks/

│ └── titanic\_analysis.ipynb # Jupyter notebook for analysis

3. The README.md should briefly describe the project and instructions for running the code.

4. Add your Jupyter notebook that contains the code for loading, cleaning, exploring, and visualising the Titanic dataset.

Push the Code:

* Add and commit your files to the repository.
* Push the repository to GitHub

PART D - Correction Class

Correction classes will be held every Saturday from 4 pm to 6 pm Nigerian Time on the TDI official Discord or a

Google Meet link will be shared before the class.

Good luck with your assignment! We hope you find it both challenging and rewarding. If you have any questions, feel

free to reach out to the mentors or the community on the TDI platform.