219114/115 Programming I

OOP programming laboratory 2

To do this lab, you must have first completed the OOP programming lab. 1. Click on the link below to do that lab before proceeding to do this lab.

oop_lab_1

https://gist.github.com/parujr/d598417e006c26c1504276913b5e2316

Commit 3
We want to turn the code in Commit 2 from a procedural-style code into an OOP-style code

- Let's think what you have to do:
  - Probably need a "DataLoader" class to load a csv file and convert them into a Table
  - Probably need a "Table" class
  - What operations are needed in that class
  - How do we define __init__ in that class

The starting code is provided via this link:
https://gist.github.com/parujr/acdbe501e3a327d6297c46ee726c0242

Your job is to complete the body of the Table class and test your implementation with the provided test code.

Once you are done with the implementation, create a README.md file that includes:

- Lab Overview - What is this lab all about?
- Project Structure - How files are organized, e.g,

```
oop_lab_1/
|
├── README.md              # This file
├── Cities.csv             # The dataset
|---- data_processing.py   # The analysis code
```

- Design Overview - Detailed explanation of each class, detailing attributes and key methods
- How to test and run your code

Make the final commit with a meaningful message and push your changes to the remote Github repository.

What to submit

- Show your final commit at your remote Github repository to the TAs so they can check you off
- Put the link to the Github repository to Google Classroom for grading

Aside note

Why lambda?
Q: Why anonymous functions like lambdas in Python? It seems that they can always be substituted by real non-anonymous functions.

A (by ChatGPT): You're right that **every lambda function in Python can be replaced with a regular function**. So why do lambdas exist?

The short answer: **convenience, readability in small cases, and functional-style programming.**

Why Lambda Functions?

Reason	Explanation	Example
Conciseness	Write small throwaway functions without naming them	`sorted(items, key=lambda x: x.age)`
Inline usage	Useful when passing a function as an argument	`map(lambda x: x*2, nums)`
Functional style	Works naturally with `map`, `filter`, `reduce`	`filter(lambda x: x>0, nums)`
Avoid clutter	Don’t pollute namespace with tiny helper functions	No need to define `def get_age(p):`

Rule of Thumb

Use `lambda` when	Use `def` when
Tiny function inline	Function needs a name
One expression	Multiple statements
Used once or twice	Reused many times
Improves readability	Hurts readability

Bottom Line

Yes, lambdas are not necessary — but they make code:

- cleaner in small contexts
- more expressive in functional patterns
- less cluttered with helper functions

Think of them like **sticky notes** in code: useful, temporary, and lightweight.