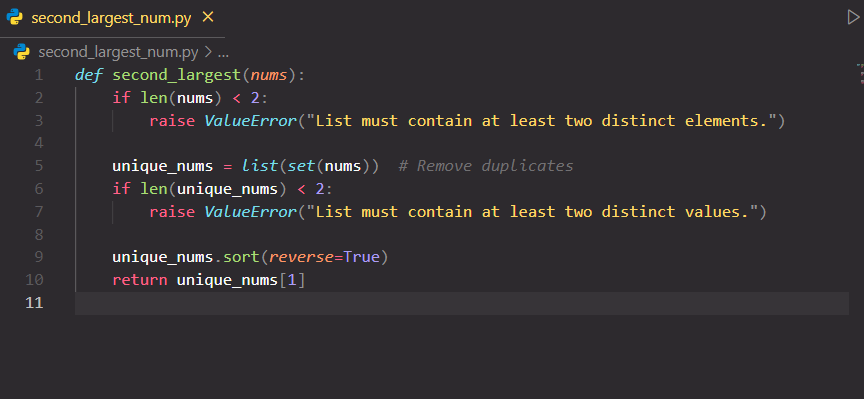
Name :Osuru Lawrence

Email:osurulawrence@gmail.com

**Question 1:** Write a function that returns the second-largest number in a given list of integers.  
 *(Provide your code and a short explanation of your approach.)*



Remove duplicates using set() — this ensures repeated values don’t affect the second-largest selection.

Check list size — after removing duplicates, there should be at least two elements. Otherwise, a second-largest value doesn’t exist.

Sort the list in descending order — so the largest is first, and the second-largest is second.

Return the second element (index 1) of the sorted list.

**Question 2:** Explain how you would optimize a page that loads too slowly. Mention at least **three causes** and how you’d fix each.

**Large or Unoptimized Images**

**Cause:** Images that are too large in file size or dimensions can significantly slow down a page, especially on mobile or low-bandwidth connections.

Fix:

Compress images using tools like TinyPNG, ImageOptim, or WebP formats to reduce file size without noticeable quality loss.

Use responsive images so devices load appropriately sized images.

Lazy-load offscreen images using the loading="lazy" attribute so they’re only loaded when needed.

**Too Many HTTP Requests**

C**ause:** Each asset (CSS, JS, fonts, icons, images) requires a separate request. Too many requests slow down initial rendering.

Fix:

Combine CSS and JS files where possible to reduce the number of requests.

Use bundling and minification tools

Leverage browser caching and set appropriate cache headers for static assets.

**Unoptimized JavaScript or Render-Blocking Resources**

**Cause:** Large JavaScript files or scripts that block rendering (especially in the <head>) can delay the display of content.

Fix:

Defer or async JS scripts using defer or async attributes so the HTML can load first.

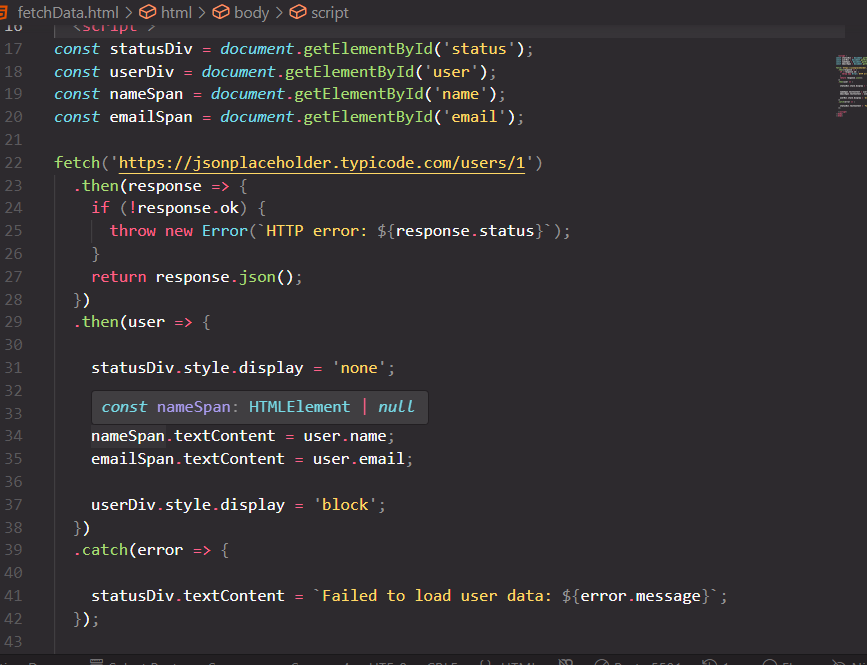
Minify JS to reduce file size.

Audit performance with tools like Chrome DevTools or Lighthouse to find long-running scripts or inefficient DOM manipulation.

**Question 3 (Front-end):** You are creating a simple profile page that fetches user data from an API (https://jsonplaceholder.typicode.com/users/1).  
 Explain or show code for:

Fetching and displaying the user’s name and email.  
Handling the loading and error states.

HTML  


Javascript  


**Question 4 (Back-end / Logic):** A small store wants to calculate total sales from this dataset:

[

  {"item": "Pen", "price": 20, "quantity": 3},

  {"item": "Book", "price": 200, "quantity": 2},

  {"item": "Bag", "price": 800, "quantity": 1}

]

Write a short function to calculate the **total revenue**.



**Question 5:** You’ve been given this code snippet:

numbers = [1, 2, 3, 4, 5]

for i in range(len(numbers)):

    if i % 2 == 0:

        numbers.remove(i)

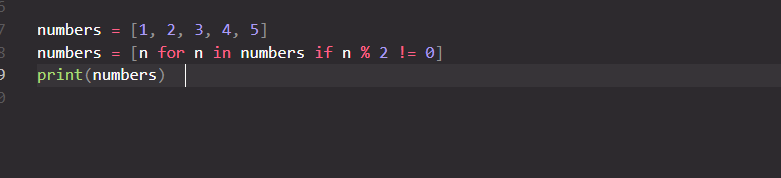
print(numbers)

1. What’s wrong with the code?  
   **This loop removes elements from the list while iterating over it by index, which changes the list length and shifts the remaining elements, leading to unexpected behavior or skipping elements.**

**It mistakenly uses the index i as a value, not the value itself.**

1. What will it output?

**The output is: [1, 3, 5]**

1. How would you fix it to remove even numbers correctly?  
     
   the code should be rewritten like this  
   

**Question 6:** Explain how you would use Git to collaborate on a team project with other developers.  
 Mention at least:

* One common Git command you use often.
* One problem you’ve faced while using Git and how you solved it.

Collaborating on a team project using Git involves using version control to track changes, coordinate work, and avoid conflicts. Here's how I’d approach it, with the required points addressed:

1. Clone the repository from a shared remote like GitHub:

**git clone https://github.com/your-team/project.git**

1. Create a new branch for your feature or bug fix:

**git checkout -b feature/login-form**

1. Make changes locally, then stage and commit:

**git add .**

**git commit -m "Add login form UI"**

1. Pull latest changes from the main branch to stay updated:

**git pull origin main**

1. Push your branch to the remote:

**git push origin feature/login-form**

1. Create a pull request (PR) so teammates can review and merge your code.

**One Common Git Command:**

git status

I use this constantly to check what files have changed, what's staged, and whether I'm on the correct branch.

**One Problem I've Faced with Git and How I Solved It:**

Problem: Merge conflict during a pull from main.

git pull origin main

Git responds with:

CONFLICT : Merge conflict in src/App.js

Why it happened: I changed the same lines in App.js that someone else changed and already pushed to main.

**Solution:**

Git marked the conflicting parts like:

<<<<<<< HEAD

my changes

=======

their changes

>>>>>>> origin/main

I manually edited the file to keep or merge the right changes.

Then I staged and committed the resolved conflict:

git add src/App.js

git commit -m "Resolve merge conflict in App.js"