

2303A51563

batch =10

import hashlib

```
def hash_email(email):  
    # Hashing email to protect user identity  
    return hashlib.sha256(email.encode()).hexdigest()
```

```
name = input("Enter your name: ")
```

```
age = int(input("Enter your age: "))
```

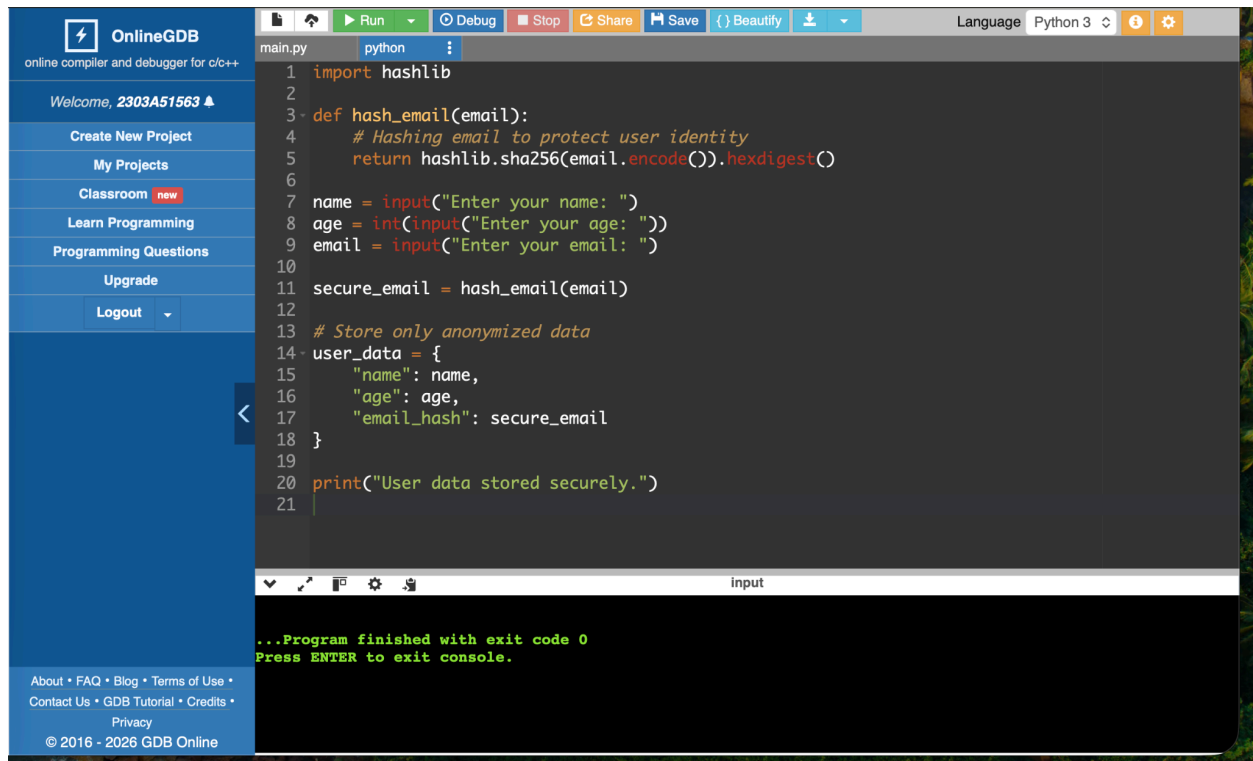
```
email = input("Enter your email: ")
```

```
secure_email = hash_email(email)
```

```
# Store only anonymized data
```

```
user_data = {  
    "name": name,  
    "age": age,  
    "email_hash": secure_email  
}
```

```
print("User data stored securely.")
```



The screenshot displays the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area shows a Python script with the following code:

```
1 import hashlib  
2  
3 def hash_email(email):  
4     # Hashing email to protect user identity  
5     return hashlib.sha256(email.encode()).hexdigest()  
6  
7 name = input("Enter your name: ")  
8 age = int(input("Enter your age: "))  
9 email = input("Enter your email: ")  
10  
11 secure_email = hash_email(email)  
12  
13 # Store only anonymized data  
14 user_data = {  
15     "name": name,  
16     "age": age,  
17     "email_hash": secure_email  
18 }  
19  
20 print("User data stored securely.")  
21
```

At the bottom, the console output shows: "...Program finished with exit code 0" and "Press ENTER to exit console."

```
def sentiment_analysis(text):
```

```
# Simple keyword-based sentiment analysis
positive_words = ["good", "happy", "excellent"]
negative_words = ["bad", "sad", "terrible"]
```

```
text = text.lower()
```

```
# Bias mitigation: neutral language, no offensive terms
score = 0
```

```
for word in positive_words:
```

```
    if word in text:
```

```
        score += 1
```

```
for word in negative_words:
```

```
    if word in text:
```

```
        score -= 1
```

```
if score > 0:
```

```
    return "Positive"
```

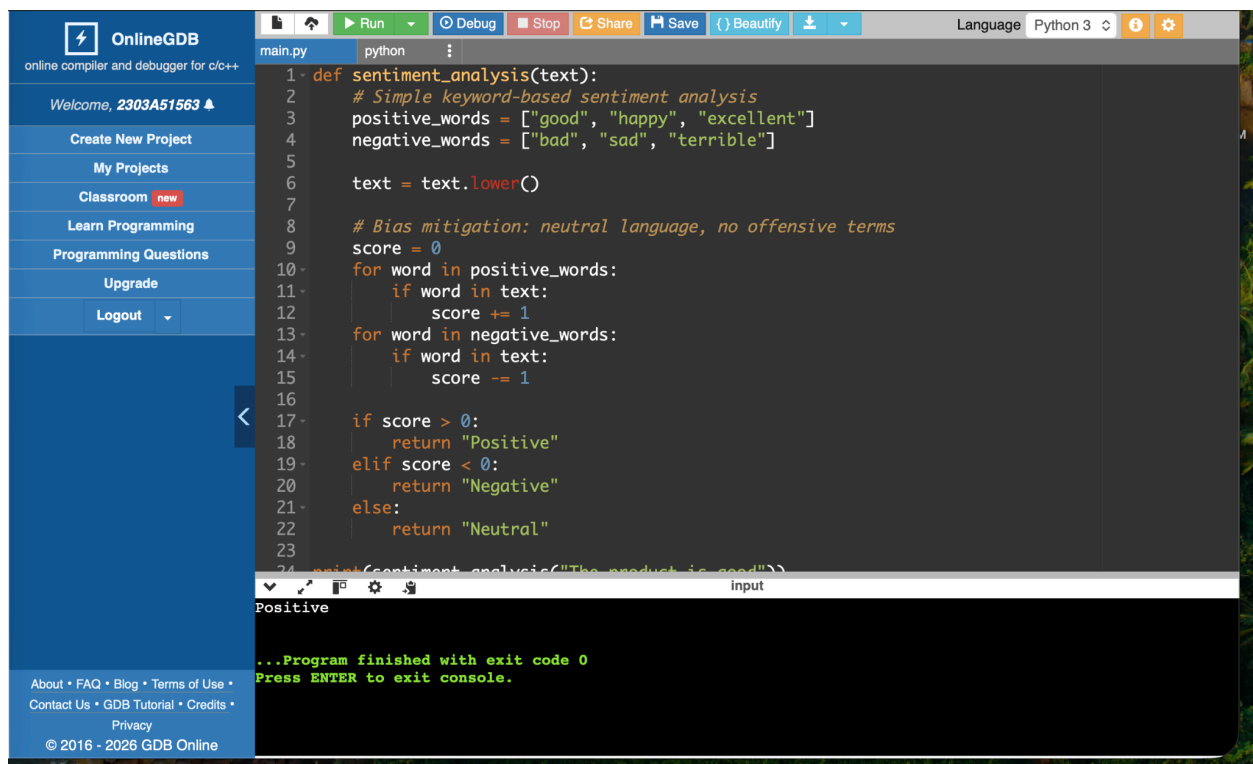
```
elif score < 0:
```

```
    return "Negative"
```

```
else:
```

```
    return "Neutral"
```

```
print(sentiment_analysis("The product is good"))
```



The screenshot displays the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area shows a Python script for sentiment analysis. The script defines two lists of words, 'positive\_words' and 'negative\_words', and a function 'sentiment\_analysis(text)'. The function iterates through the words in the text, incrementing the score for positive words and decrementing for negative words. It then returns 'Positive', 'Negative', or 'Neutral' based on the score. The script is executed, and the output shows 'Positive'. The bottom of the interface shows the program finished with exit code 0 and a prompt to press ENTER to exit the console.

```
1 def sentiment_analysis(text):
2     # Simple keyword-based sentiment analysis
3     positive_words = ["good", "happy", "excellent"]
4     negative_words = ["bad", "sad", "terrible"]
5
6     text = text.lower()
7
8     # Bias mitigation: neutral language, no offensive terms
9     score = 0
10    for word in positive_words:
11        if word in text:
12            score += 1
13    for word in negative_words:
14        if word in text:
15            score -= 1
16
17    if score > 0:
18        return "Positive"
19    elif score < 0:
20        return "Negative"
21    else:
22        return "Neutral"
23
24 print(sentiment_analysis("The product is good"))
```

input

Positive

...Program finished with exit code 0  
Press ENTER to exit console.

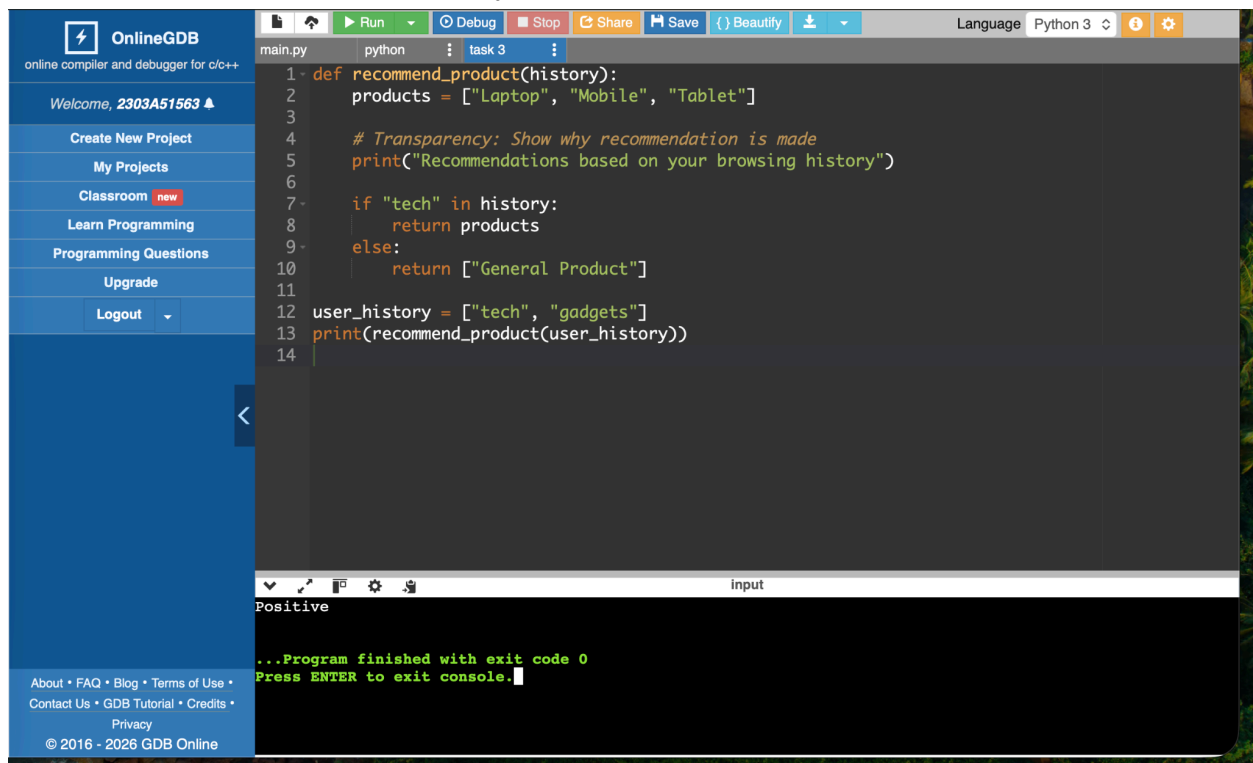
### Task 3

```
def recommend_product(history):
    products = ["Laptop", "Mobile", "Tablet"]

    # Transparency: Show why recommendation is made
    print("Recommendations based on your browsing history")

    if "tech" in history:
        return products
    else:
        return ["General Product"]

user_history = ["tech", "gadgets"]
print(recommend_product(user_history))
```



The screenshot displays the OnlineGDB web interface. On the left is a sidebar with navigation links like 'Create New Project', 'My Projects', and 'Classroom'. The main area shows a Python script being executed. The code defines a function `recommend_product` that returns a list of products based on the presence of 'tech' in the history. Below the code editor, the output console shows the word 'Positive' and a message indicating the program finished successfully with exit code 0.

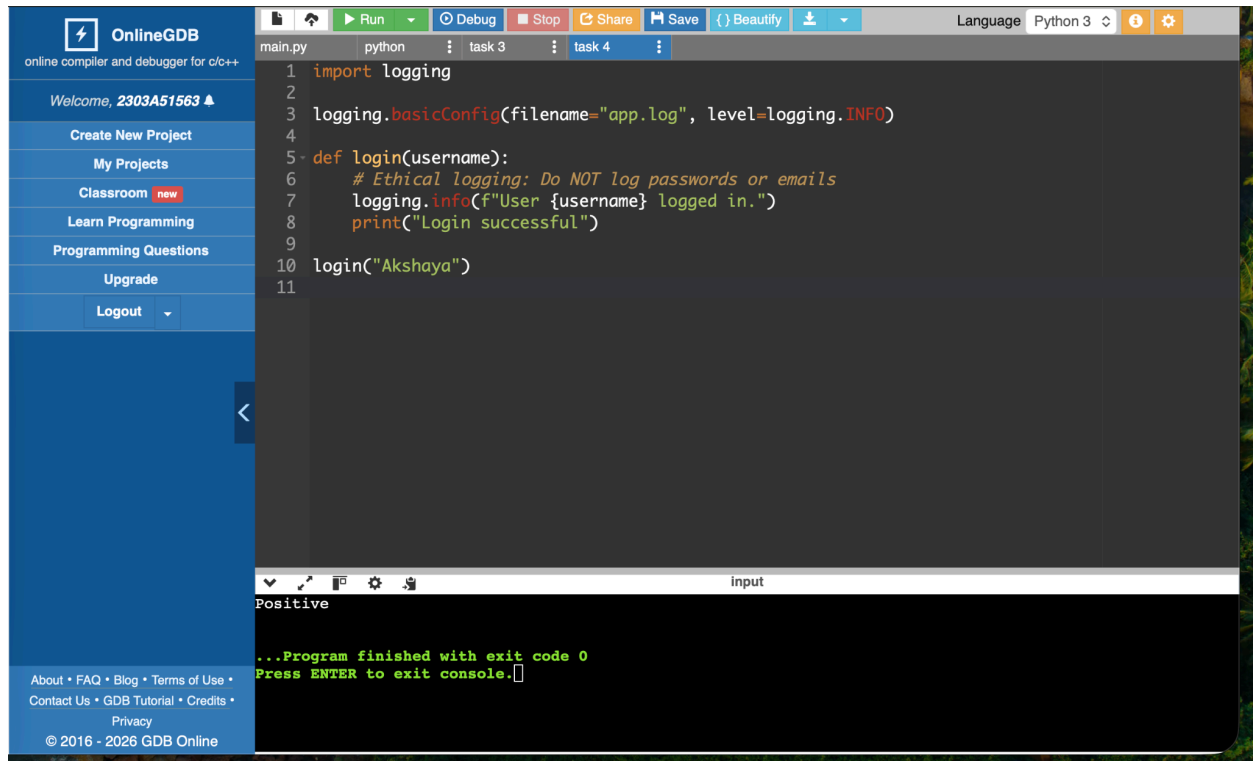
### Task 3

import logging

```
logging.basicConfig(filename="app.log", level=logging.INFO)
```

```
def login(username):
    # Ethical logging: Do NOT log passwords or emails
    logging.info(f"User {username} logged in.")
    print("Login successful")
```

login("harshavardhan")



The screenshot shows the OnlineGDB web interface. On the left is a blue sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom' (marked 'new'), 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area displays a Python script in a dark-themed editor. The script imports the 'logging' module, configures it to write to 'app.log' at the 'INFO' level, and defines a 'login' function. The function has a comment about ethical logging and prints a success message. It is called with the username 'Akshaya'. Below the editor, the console output shows 'Positive' and a message indicating the program finished with exit code 0. The top of the interface includes a toolbar with buttons for Run, Debug, Stop, Share, Save, and Beautify, along with a language selector set to 'Python 3'.

```
1 import logging
2
3 logging.basicConfig(filename="app.log", level=logging.INFO)
4
5 def login(username):
6     # Ethical logging: Do NOT log passwords or emails
7     logging.info(f"User {username} logged in.")
8     print("Login successful")
9
10 login("Akshaya")
11
```

input

Positive

...Program finished with exit code 0  
Press ENTER to exit console.

Tak 5

Responsible AI Model Usage:

- Model accuracy depends on data quality
  - Not for critical medical/legal decisions
  - Bias may exist if training data is biased
- """

```
from sklearn.linear_model import LinearRegression
```


```
X = [[1], [2], [3], [4]]
```

```
y = [2, 4, 6, 8]
```

```
model = LinearRegression()
```

```
model.fit(X, y)
```

```
print("Prediction:", model.predict([[5]]))
```

**OnlineGDB**  
online compiler and debugger for c/c++

Welcome, **2303A51563** ▲

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Upgrade

Logout ▼

About • FAQ • Blog • Terms of Use •  
Contact Us • GDB Tutorial • Credits •  
Privacy  
© 2016 - 2026 GDB Online

main.pypythontask 3task 4task 5

1"""

2Responsible AI Model Usage:

3- Model accuracy depends on data quality

4- Not for critical medical/legal decisions

5- Bias may exist if training data is biased

6"""

7

8from sklearn.linear\_model import LinearRegression

9

10X = [[1], [2], [3], [4]]

11y = [2, 4, 6, 8]

12

13model = LinearRegression()

14model.fit(X, y)

15

16print("Prediction:", model.predict([[5]]))

17

input

Positive

...Program finished with exit code 0  
Press ENTER to exit console.