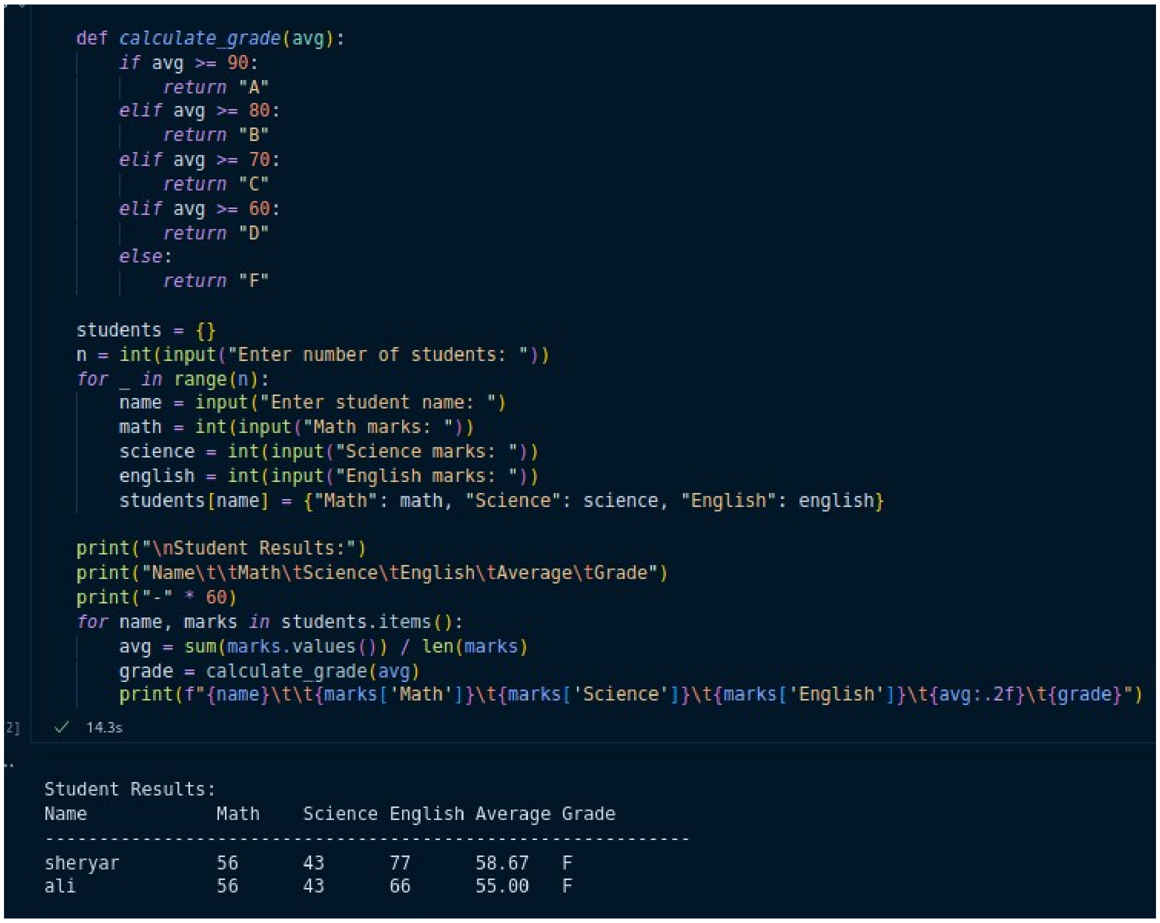
Python Assignment

Task 1: Student Grades Management (Lists + Dictionaries + Functions)

Write a program that takes input for multiple students (name and marks in 3 subjects).
- Store the data in a dictionary of dictionaries:
  {"Ali": {"Math": 85, "Science": 90, "English": 78}, ...}
- Calculate each student's average score and assign a grade (A, B, C, D, F).
- Print the results in a neat table format.



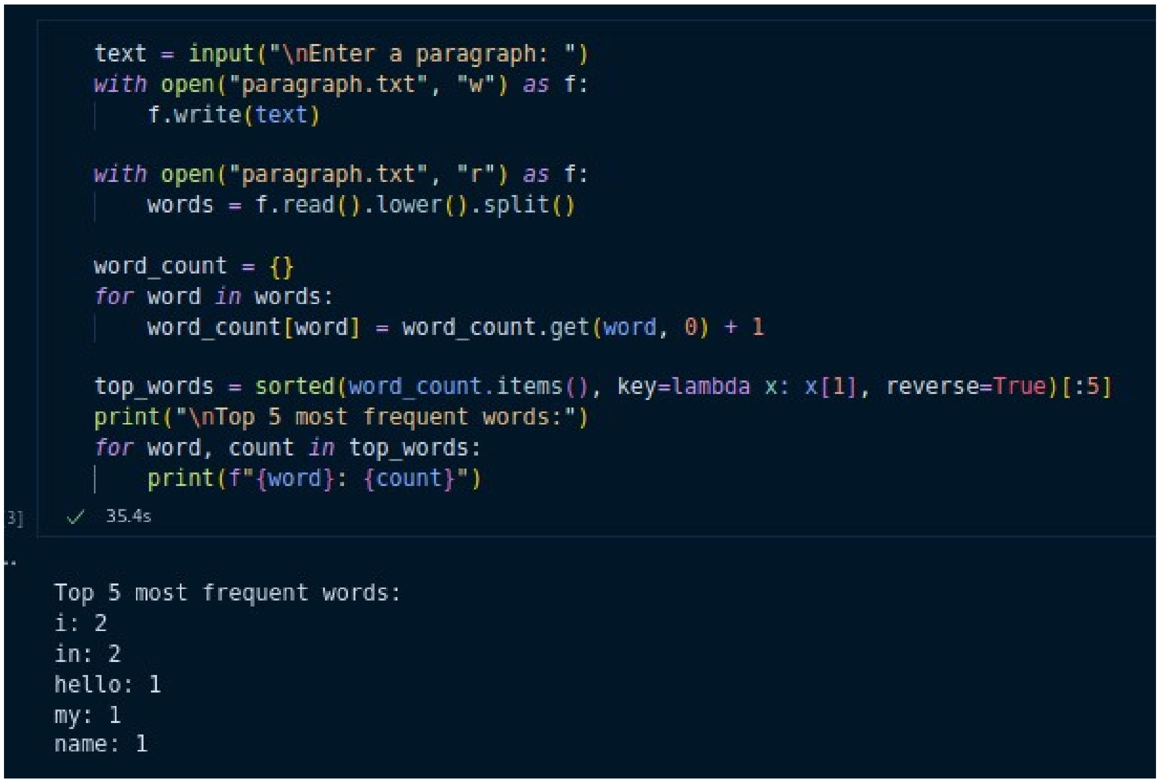
```python
def calculate_grade(avg):
    if avg >= 90:
        return "A"
    elif avg >= 80:
        return "B"
    elif avg >= 70:
        return "C"
    elif avg >= 60:
        return "D"
    else:
        return "F"

students = {}
n = int(input("Enter number of students: "))
for _ in range(n):
    name = input("Enter student name: ")
    math = int(input("Math marks: "))
    science = int(input("Science marks: "))
    english = int(input("English marks: "))
    students[name] = {"Math": math, "Science": science, "English": english}

print("\nStudent Results:")
print("Name\t\tMath\tScience\tEnglish\tAverage\tGrade")
print("-" * 60)
for name, marks in students.items():
    avg = sum(marks.values()) / len(marks)
    grade = calculate_grade(avg)
    print(f"{name}\t\t{marks['Math']}\t{marks['Science']}\t{marks['English']}\t{avg:.2f}\t{grade}")
```

✓ 14.3s

```
Student Results:
Name            Math    Science English Average Grade
------------------------------------------------------------
sheryar         56      43      77      58.67   F
ali             56      43      66      55.00   F
```

Task 2: Word Frequency Counter (Strings + Dictionaries + File Handling)

- Ask the user to enter a paragraph and save it into a text file.
- Read the file and count how many times each unique word appears.
- Ignore case sensitivity (The and the should be treated the same).
- Display the top 5 most frequent words.

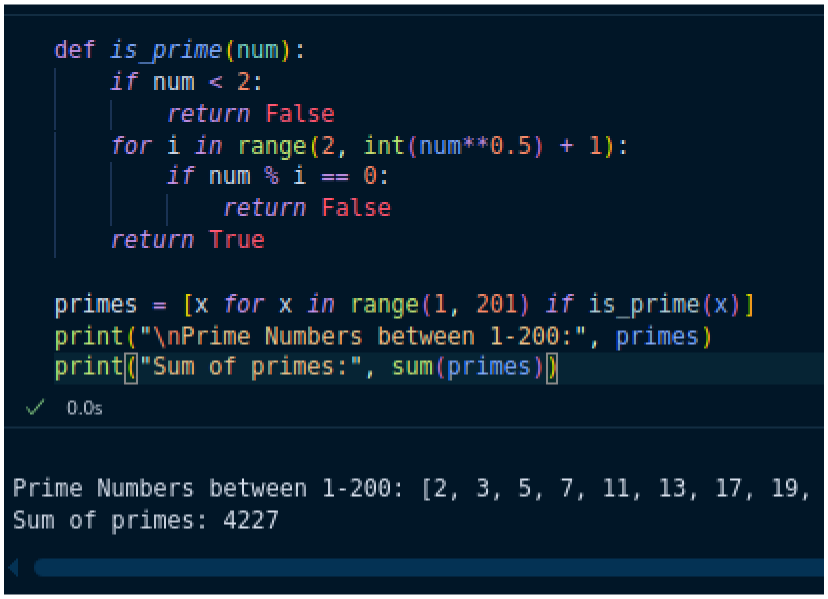


```python
text = input("\nEnter a paragraph: ")
with open("paragraph.txt", "w") as f:
    f.write(text)

with open("paragraph.txt", "r") as f:
    words = f.read().lower().split()

word_count = {}
for word in words:
    word_count[word] = word_count.get(word, 0) + 1

top_words = sorted(word_count.items(), key=lambda x: x[1], reverse=True)[:5]
print("\nTop 5 most frequent words:")
for word, count in top_words:
    print(f"{word}: {count}")
```

```
Top 5 most frequent words:
i: 2
in: 2
hello: 1
my: 1
name: 1
```

Task 3: Prime Number Analyzer (Functions + Loops + List Comprehension)

- Write a function to check whether a number is prime.
- Generate a list of prime numbers between 1 and 200 using list comprehension.
- Find the sum of these prime numbers.



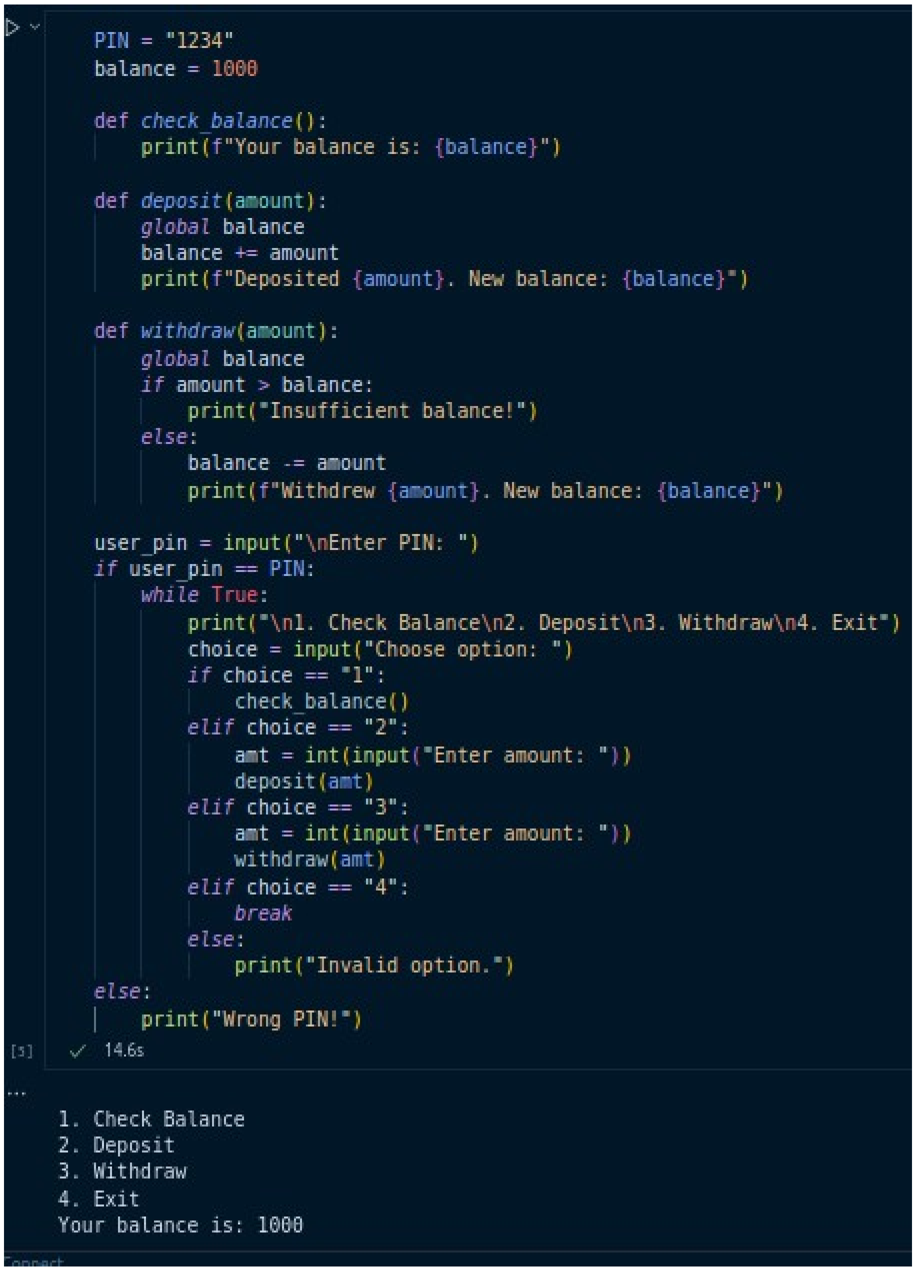
```python
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

primes = [x for x in range(1, 201) if is_prime(x)]
print("\nPrime Numbers between 1-200:", primes)
print("Sum of primes:", sum(primes))
```

✓ 0.0s

```
Prime Numbers between 1-200: [2, 3, 5, 7, 11, 13, 17, 19,
Sum of primes: 4227
```

Task 4: Mini ATM Simulation (Conditional Statements + Loops + Functions)

Create a mini ATM program with the following features:
- User must enter PIN (predefined).
- Menu: Check Balance, Deposit, Withdraw, Exit.
- Ensure withdrawals cannot exceed the balance.
- Use functions to implement each feature.



```python
PIN = "1234"
balance = 1000

def check_balance():
    print(f"Your balance is: {balance}")

def deposit(amount):
    global balance
    balance += amount
    print(f"Deposited {amount}. New balance: {balance}")

def withdraw(amount):
    global balance
    if amount > balance:
        print("Insufficient balance!")
    else:
        balance -= amount
        print(f"Withdrew {amount}. New balance: {balance}")

user_pin = input("\nEnter PIN: ")
if user_pin == PIN:
    while True:
        print("\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Exit")
        choice = input("Choose option: ")
        if choice == "1":
            check_balance()
        elif choice == "2":
            amt = int(input("Enter amount: "))
            deposit(amt)
        elif choice == "3":
            amt = int(input("Enter amount: "))
            withdraw(amt)
        elif choice == "4":
            break
        else:
            print("Invalid option.")
else:
    print("Wrong PIN!")
```

```
[5]  ✓ 14.6s
```

```
...
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Your balance is: 1000
```

Task 5: Data Filtering with Lambda + Map + Filter

Given a list of dictionaries of employees:

employees = [

{"name": "Sara", "age": 25, "salary": 50000},

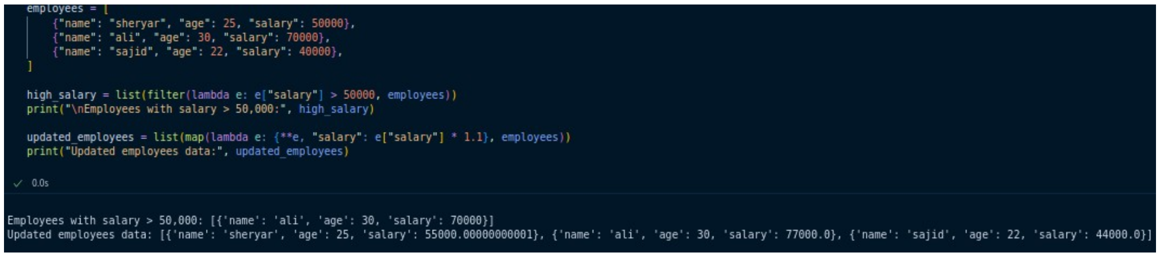
{"name": "Ali", "age": 30, "salary": 70000},

{"name": "John", "age": 22, "salary": 40000},

]

- Use filter to find employees with salary > 50,000.

- Use map to increase all employees’ salary by 10%.
- Print the updated data.



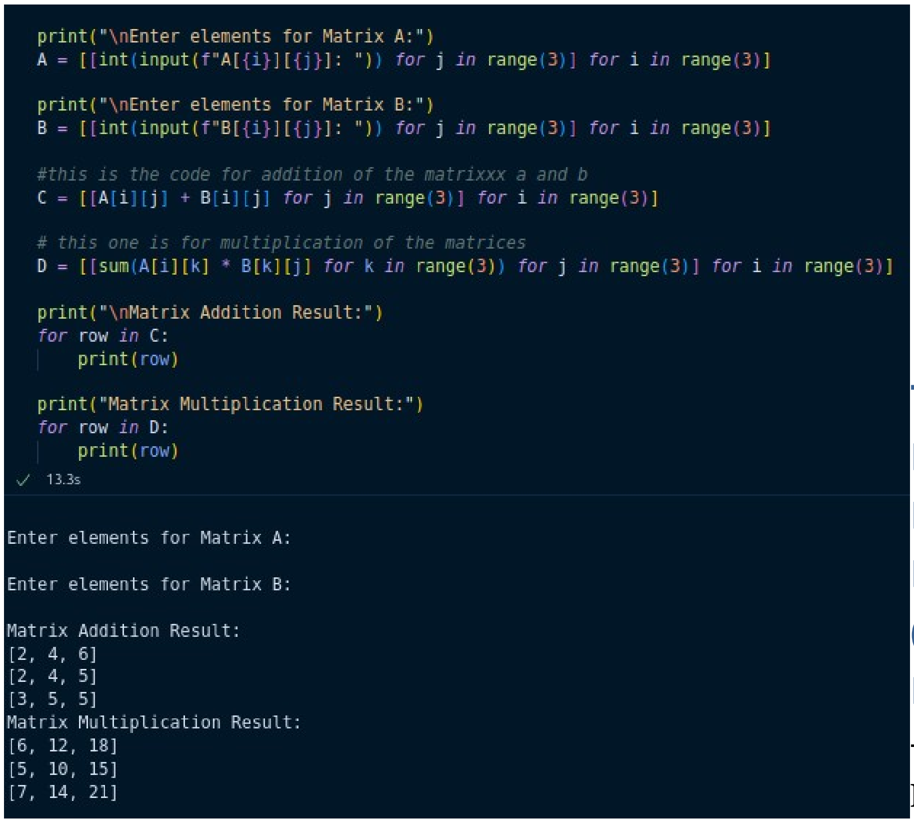
```
employees = [
    {"name": "sheryar", "age": 25, "salary": 50000},
    {"name": "ali", "age": 30, "salary": 70000},
    {"name": "sajid", "age": 22, "salary": 40000},
]

high_salary = list(filter(lambda e: e["salary"] > 50000, employees))
print("\nEmployees with salary > 50,000:", high_salary)

updated_employees = list(map(lambda e: {**e, "salary": e["salary"] * 1.1}, employees))
print("Updated employees data:", updated_employees)
```

✓ 0.0s

```
Employees with salary > 50,000: [{'name': 'ali', 'age': 30, 'salary': 70000}]
Updated employees data: [{'name': 'sheryar', 'age': 25, 'salary': 55000.00000000001}, {'name': 'ali', 'age': 30, 'salary': 77000.0}, {'name': 'sajid', 'age': 22, 'salary': 44000.0}]
```

Task 6: Matrix Operations (Nested Loops + List Comprehension)

- Take two 3x3 matrices (lists of lists) as input.
- Perform matrix addition and matrix multiplication manually (without NumPy).
- Display the results.



```
print("\nEnter elements for Matrix A:")
A = [[int(input(f"A[{i}][{j}]: ")) for j in range(3)] for i in range(3)]

print("\nEnter elements for Matrix B:")
B = [[int(input(f"B[{i}][{j}]: ")) for j in range(3)] for i in range(3)]

#this is the code for addition of the matrixxx a and b
C = [[A[i][j] + B[i][j] for j in range(3)] for i in range(3)]

# this one is for multiplication of the matrices
D = [[sum(A[i][k] * B[k][j] for k in range(3)) for j in range(3)] for i in range(3)]

print("\nMatrix Addition Result:")
for row in C:
    print(row)

print("Matrix Multiplication Result:")
for row in D:
    print(row)
```

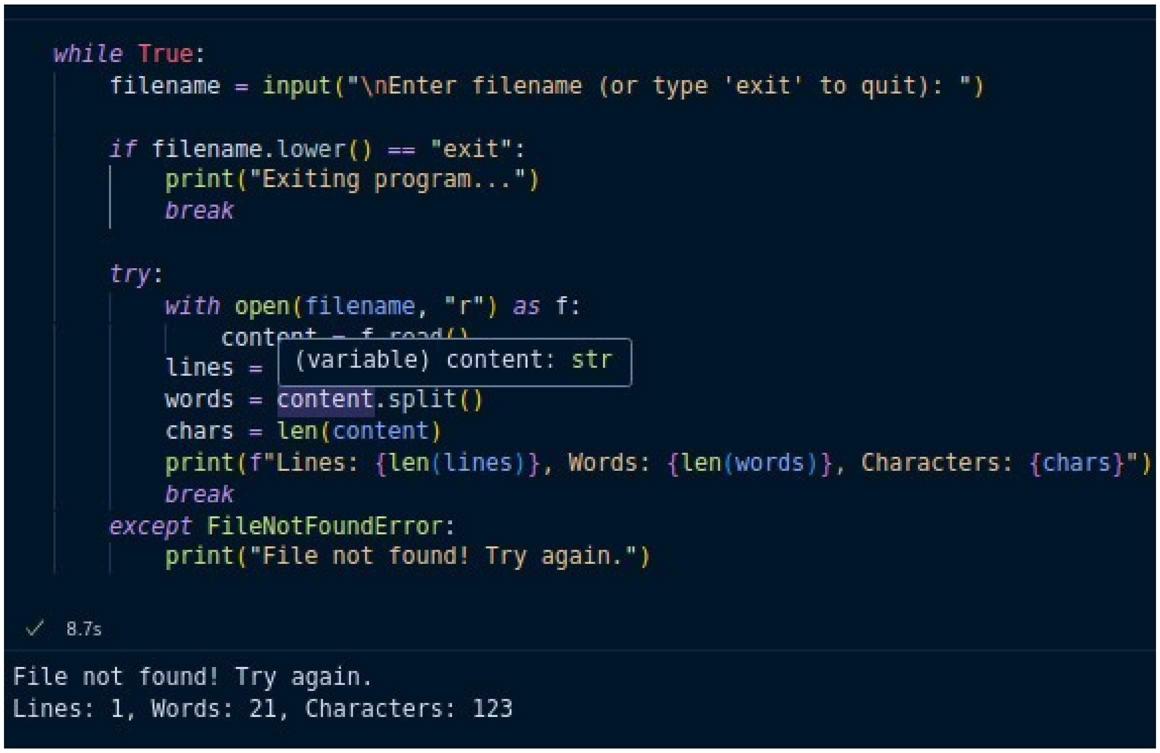
✓ 13.3s

```
Enter elements for Matrix A:

Enter elements for Matrix B:

Matrix Addition Result:
[2, 4, 6]
[2, 4, 5]
[3, 5, 5]
Matrix Multiplication Result:
[6, 12, 18]
[5, 10, 15]
[7, 14, 21]
```

Task 7: File Handling + Exception Handling (Error-Proof Program)

- Write a program that asks the user for a filename and tries to read it.

- If the file doesn’t exist, handle the exception gracefully and ask again.
- Count how many lines, words, and characters the file contains.

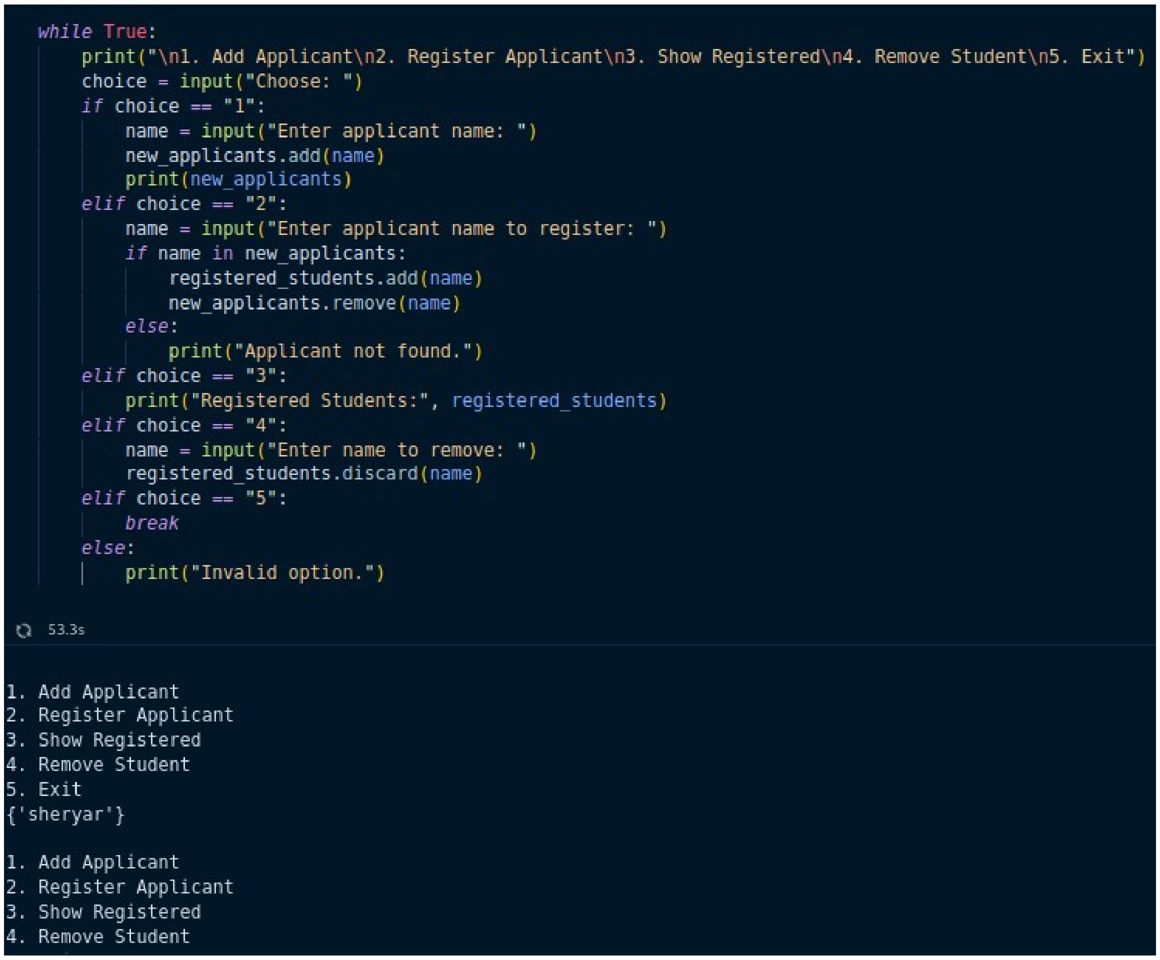


```python
while True:
    filename = input("\nEnter filename (or type 'exit' to quit): ")

    if filename.lower() == "exit":
        print("Exiting program...")
        break

    try:
        with open(filename, "r") as f:
            content = f.read()
        lines = 
        words = content.split()
        chars = len(content)
        print(f"Lines: {len(lines)}, Words: {len(words)}, Characters: {chars}")
        break
    except FileNotFoundError:
        print("File not found! Try again.")
```

```
File not found! Try again.
Lines: 1, Words: 21, Characters: 123
```

Task 8: Student Registration System (Sets + Dictionaries + Loops)

- Maintain two sets: registered_students and new_applicants.
- Allow user to:
 1. Add a new applicant
 2. Register an applicant
 3. Show all registered students
 4. Remove a student
- Ensure no duplicate registrations are possible.



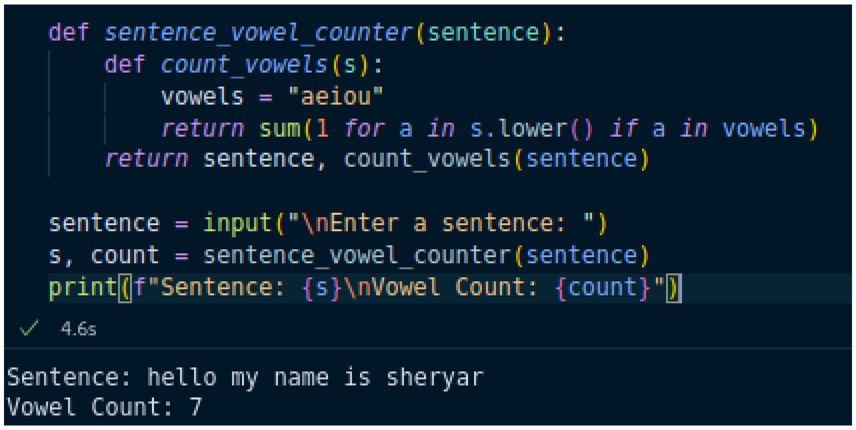
```
while True:
    print("\n1. Add Applicant\n2. Register Applicant\n3. Show Registered\n4. Remove Student\n5. Exit")
    choice = input("Choose: ")
    if choice == "1":
        name = input("Enter applicant name: ")
        new_applicants.add(name)
        print(new_applicants)
    elif choice == "2":
        name = input("Enter applicant name to register: ")
        if name in new_applicants:
            registered_students.add(name)
            new_applicants.remove(name)
        else:
            print("Applicant not found.")
    elif choice == "3":
        print("Registered Students:", registered_students)
    elif choice == "4":
        name = input("Enter name to remove: ")
        registered_students.discard(name)
    elif choice == "5":
        break
    else:
        print("Invalid option.")
```

53.3s

```
1. Add Applicant
2. Register Applicant
3. Show Registered
4. Remove Student
5. Exit
{'sheryar'}

1. Add Applicant
2. Register Applicant
3. Show Registered
4. Remove Student
```

Task 9: Nested Functions + Scope

- Write a program that contains a nested function.
- Outer function should take a sentence as input.
- Inner function should count vowels inside the sentence.
- The outer function should return both the original sentence and the vowel count.



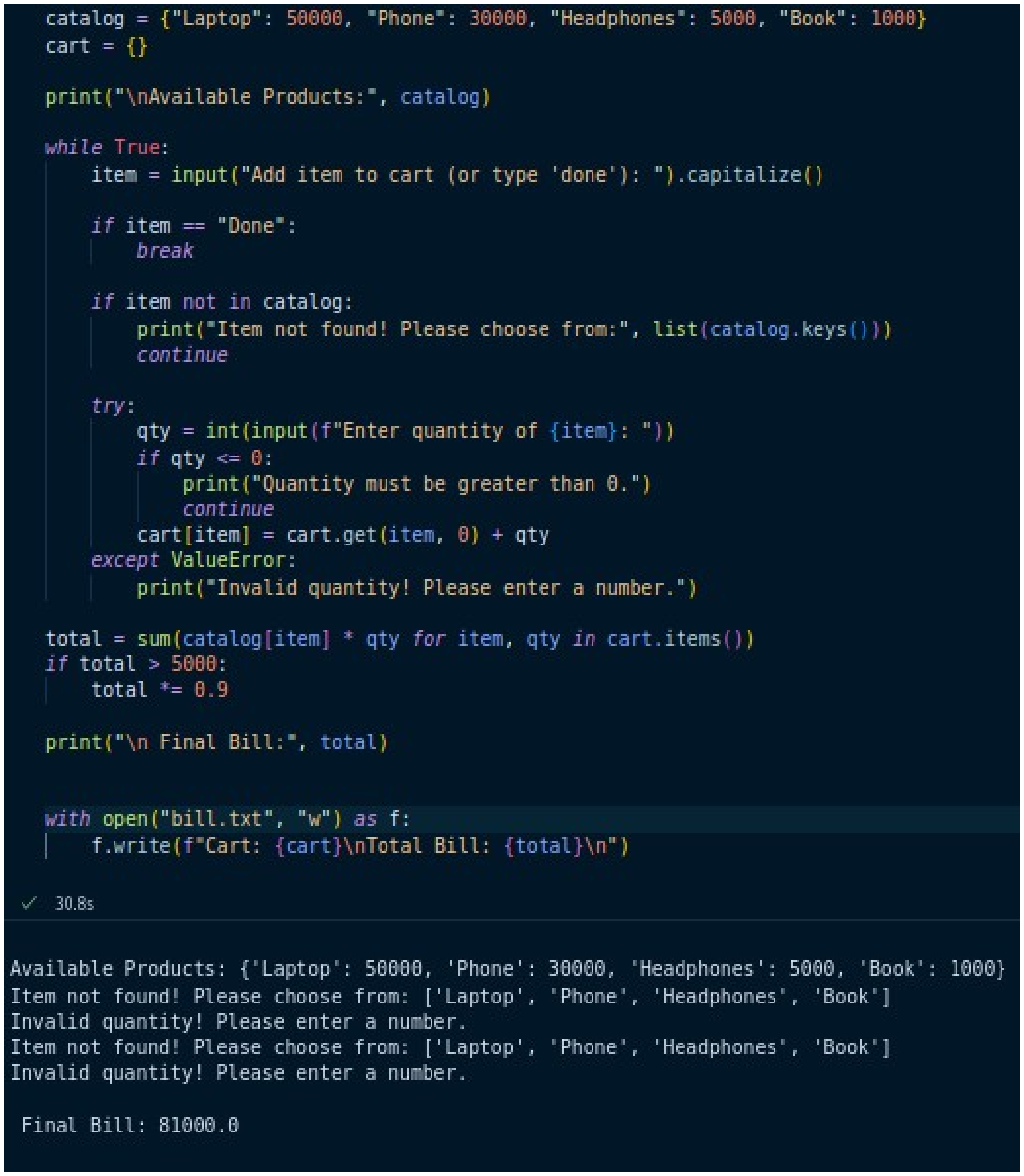
```
def sentence_vowel_counter(sentence):
    def count_vowels(s):
        vowels = "aeiou"
        return sum(1 for a in s.lower() if a in vowels)
    return sentence, count_vowels(sentence)

sentence = input("\nEnter a sentence: ")
s, count = sentence_vowel_counter(sentence)
print(f"Sentence: {s}\nVowel Count: {count}")
```

4.6s

```
Sentence: hello my name is sheryar
Vowel Count: 7
```

Task 10: Mini Shopping Cart (Dictionaries + Loops + Functions)

- Predefine a product catalog (dictionary with product name → price).
- Let the user add items to the cart until they type "done".
- Calculate total bill, apply 10% discount if total > 5000, and print the final bill.
- Save the bill in a text file.



```python
catalog = {"Laptop": 50000, "Phone": 30000, "Headphones": 5000, "Book": 1000}
cart = {}

print("\nAvailable Products:", catalog)

while True:
    item = input("Add item to cart (or type 'done'): ").capitalize()

    if item == "Done":
        break

    if item not in catalog:
        print("Item not found! Please choose from:", list(catalog.keys()))
        continue

    try:
        qty = int(input(f"Enter quantity of {item}: "))
        if qty <= 0:
            print("Quantity must be greater than 0.")
            continue
        cart[item] = cart.get(item, 0) + qty
    except ValueError:
        print("Invalid quantity! Please enter a number.")

total = sum(catalog[item] * qty for item, qty in cart.items())
if total > 5000:
    total *= 0.9

print("\n Final Bill:", total)


with open("bill.txt", "w") as f:
    f.write(f"Cart: {cart}\nTotal Bill: {total}\n")
```

✓ 30.8s

```
Available Products: {'Laptop': 50000, 'Phone': 30000, 'Headphones': 5000, 'Book': 1000}
Item not found! Please choose from: ['Laptop', 'Phone', 'Headphones', 'Book']
Invalid quantity! Please enter a number.
Item not found! Please choose from: ['Laptop', 'Phone', 'Headphones', 'Book']
Invalid quantity! Please enter a number.

 Final Bill: 81000.0
```