A
Lab Record of
Python Programming with Problem Solving

Master of Computer Application - I Sem

RUNGTA INTERNATIONAL SKILLS UNIVERSITY

SESSION: 2025-26

Guided By:	**Submitted By:-**
Mrs. Kavita Kanwar	**Sumit Kumar**
(Assistant Professor)	**Ref No. 11485**

Submitted To:

RUNGTA INTERNATIONAL SKILLS UNIVERSITY, CG

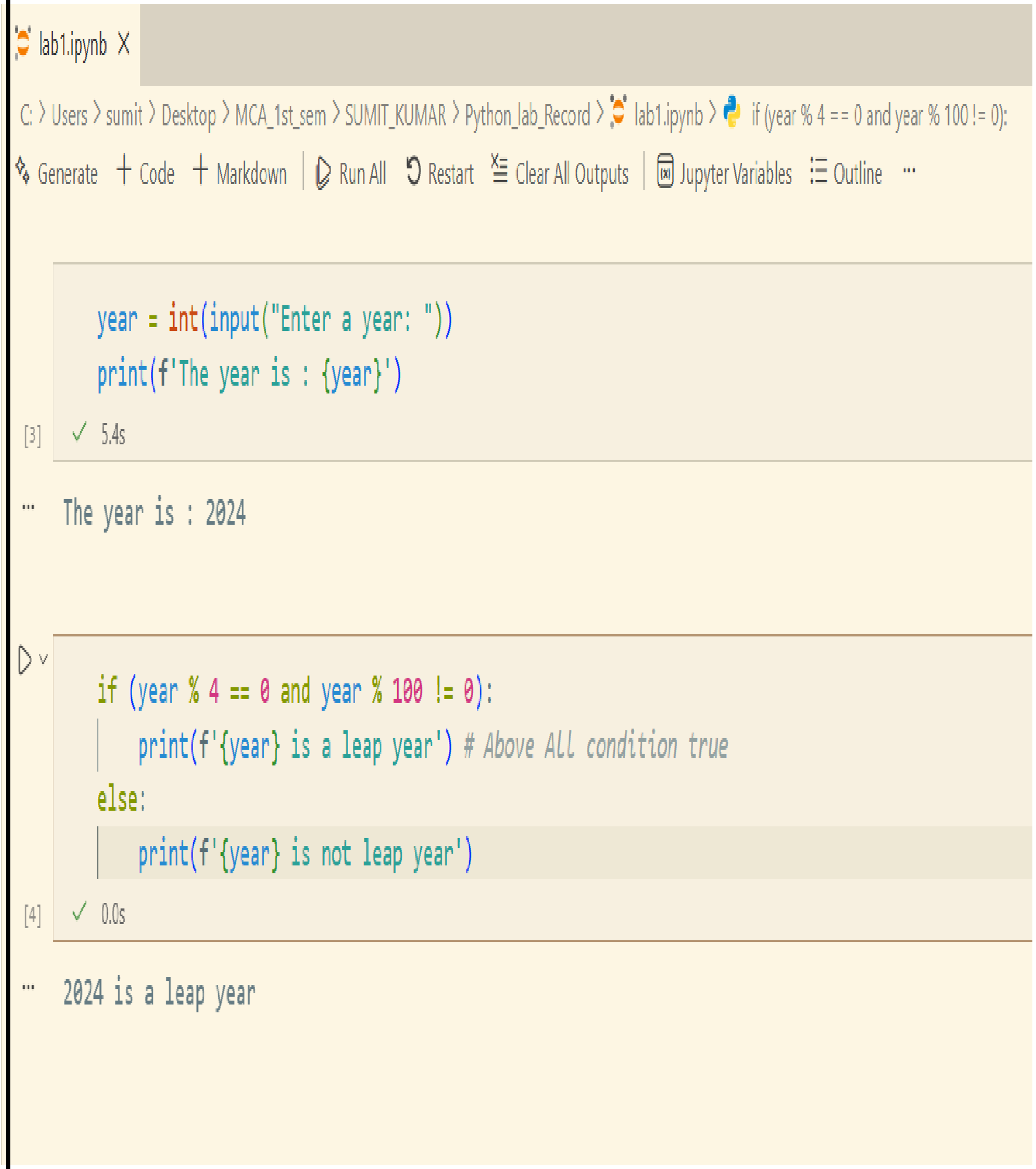
SCHOOL OF INFORMATION TECHNOLOGY

Index

9.	Read data from a CSV file containing employee details (name, department, salary) and display the average salary by department.		

PRACTICAL RECORD

AIM 1 - Write a Python program to to check whether a year is leap year or not.


lab1.ipynb X
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab1.ipynb > if (year % 4 == 0 and year % 100 != 0):
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline ...

```
year = int(input("Enter a year: "))
print(f'The year is : {year}')
```

[3] ✓ 5.4s

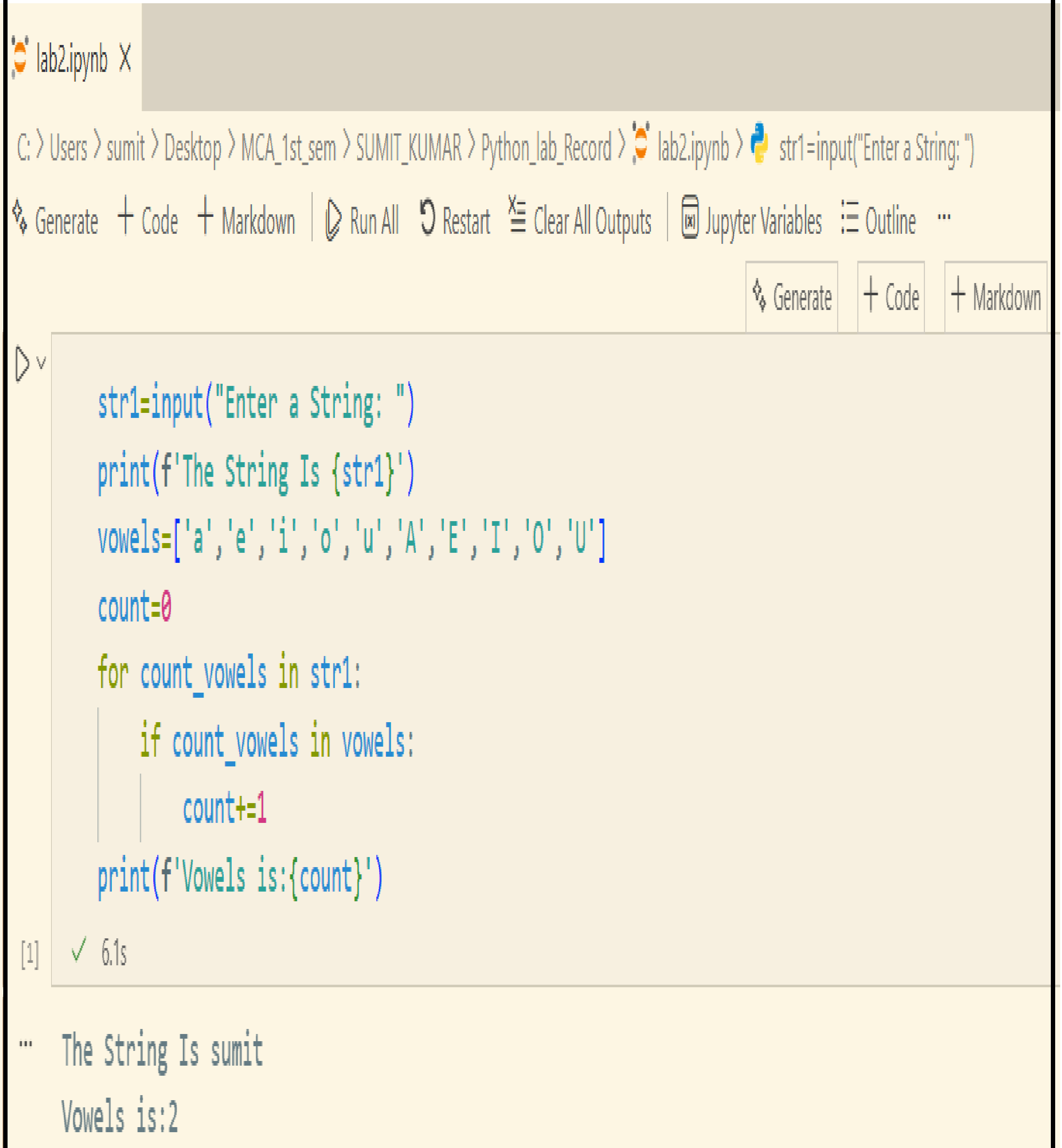
```
The year is : 2024
```

```
if (year % 4 == 0 and year % 100 != 0):
    print(f'{year} is a leap year') # Above All condition true
else:
    print(f'{year} is not leap year')
```

[4] ✓ 0.0s

```
2024 is a leap year
```

AIM 2 - Write a Python program to to count number of vowels in a string (Take User I/O)

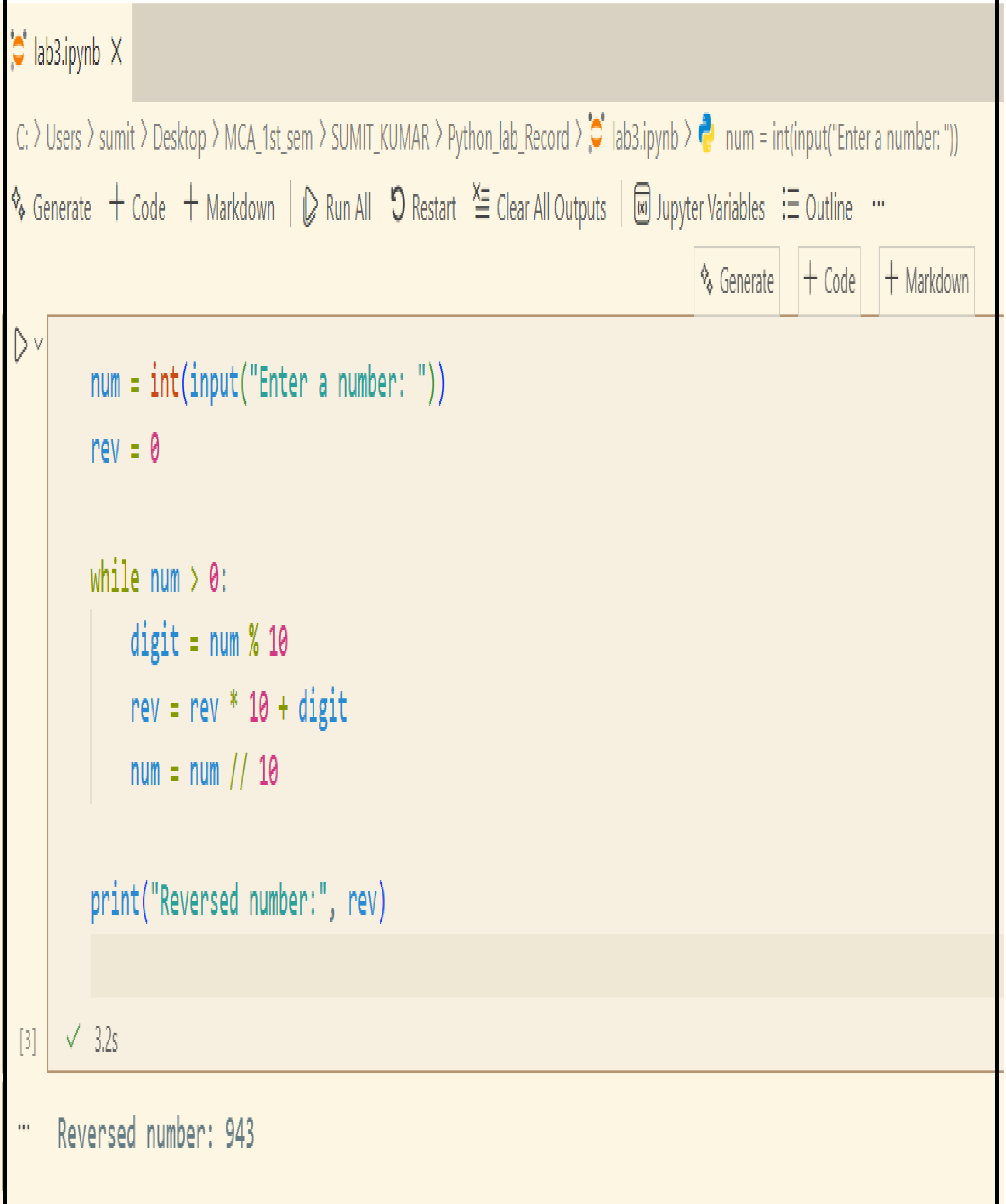

lab2.ipynb X
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab2.ipynb > str1=input("Enter a String: ")
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Generate + Code + Markdown

```python
str1=input("Enter a String: ")
print(f'The String Is {str1}')
vowels=['a','e','i','o','u','A','E','I','O','U']
count=0
for count_vowels in str1:
    if count_vowels in vowels:
        count+=1
print(f'Vowels is:{count}')
```

[1] ✓ 6.1s

```
The String Is sumit
Vowels is:2
```

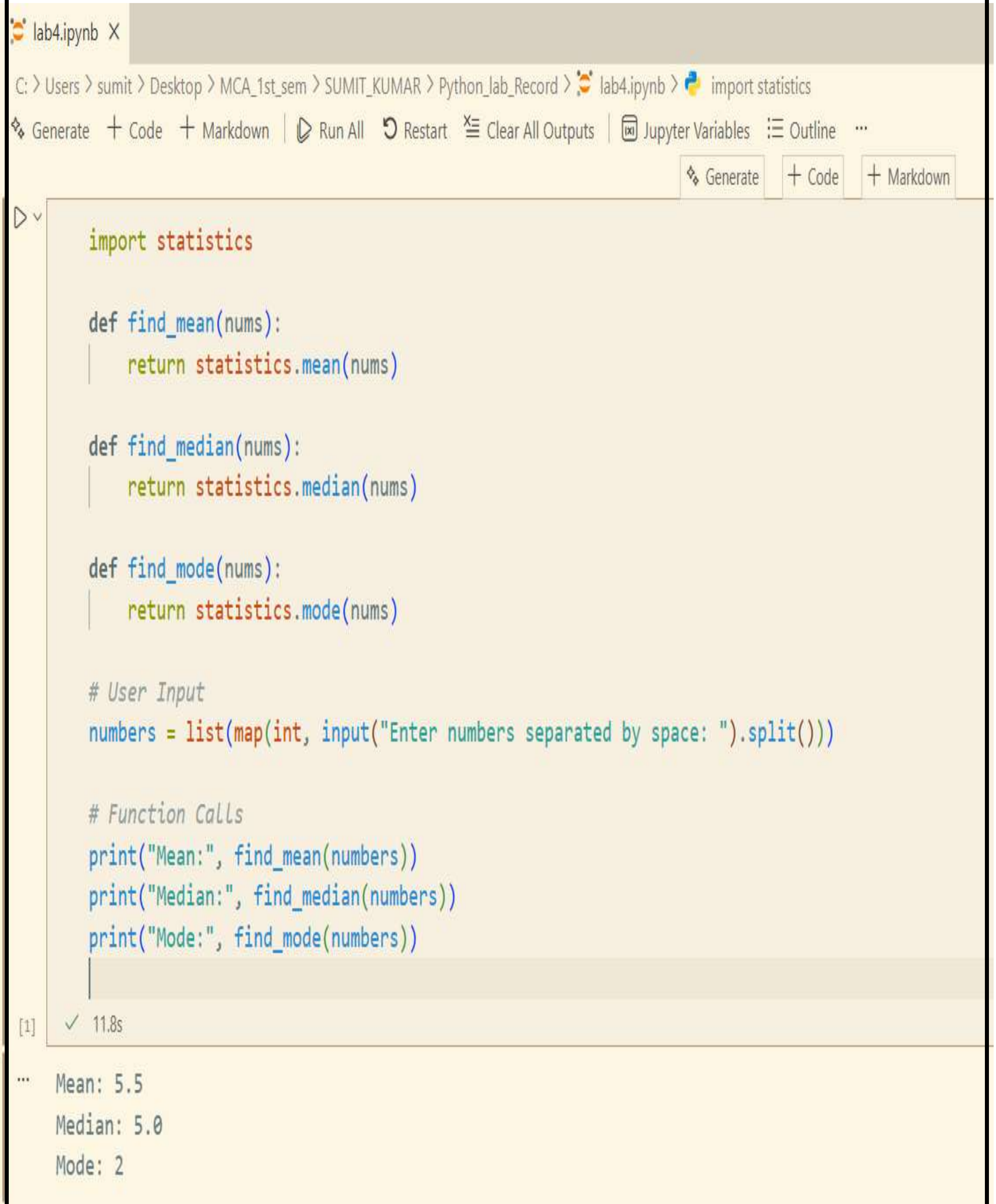
AIM 3 - Write a Python program to Reverse a Number(e.g. 123456 -> 654321)


lab3.ipynb X
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab3.ipynb > num = int(input("Enter a number: "))
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Generate + Code + Markdown
[3] ✓ 3.2s

```python
num = int(input("Enter a number: "))
rev = 0

while num > 0:
    digit = num % 10
    rev = rev * 10 + digit
    num = num // 10

print("Reversed number:", rev)
```

```
Reversed number: 943
```

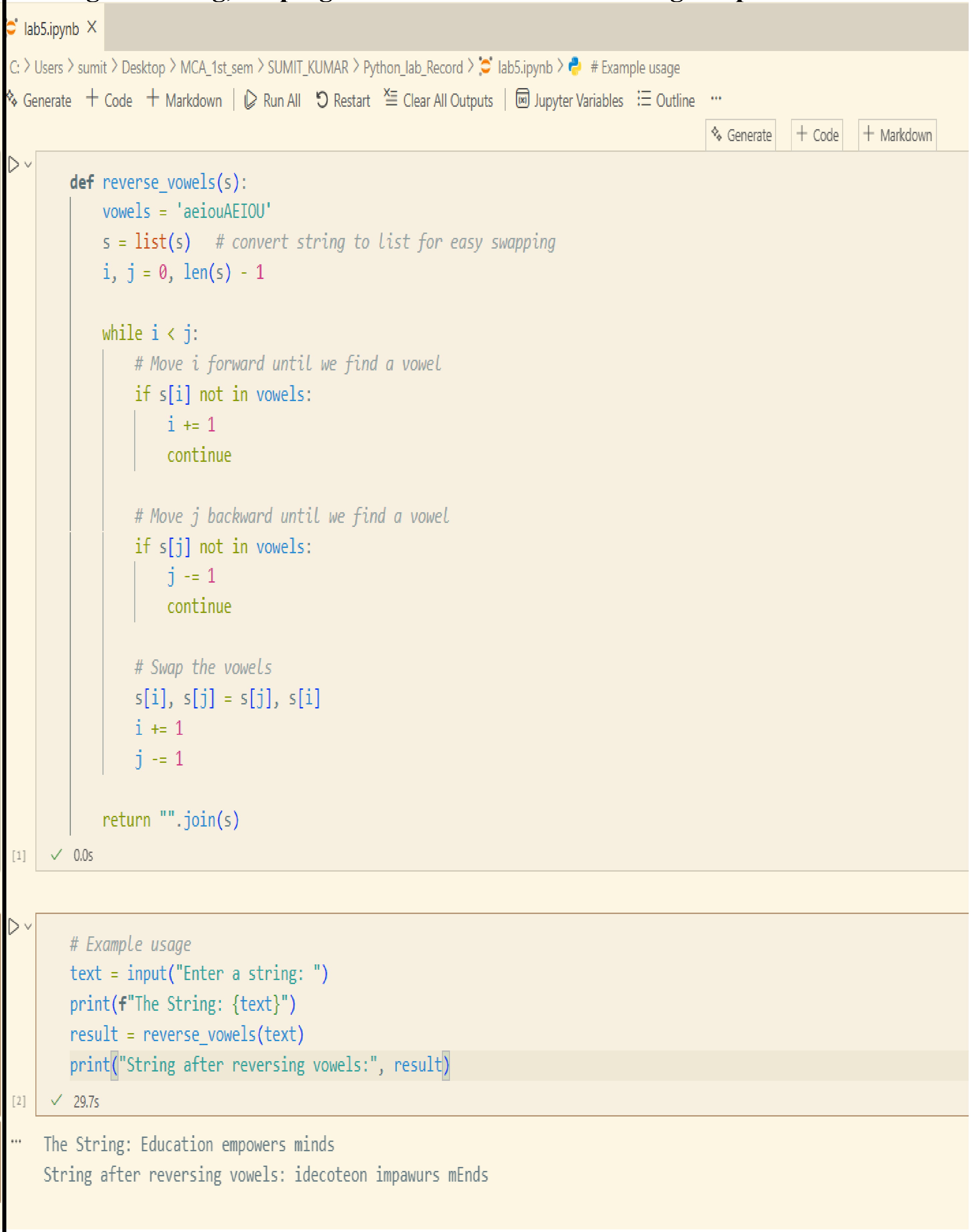
AIM 4 - Write a Python Program to Find Mean, Median & Mode of a given Number (Take User I/O)


lab4.ipynb X
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab4.ipynb > import statistics
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Generate + Code + Markdown
[1] ✓ 11.8s

```python
import statistics

def find_mean(nums):
    return statistics.mean(nums)

def find_median(nums):
    return statistics.median(nums)

def find_mode(nums):
    return statistics.mode(nums)

# User Input
numbers = list(map(int, input("Enter numbers separated by space: ").split()))

# Function Calls
print("Mean:", find_mean(numbers))
print("Median:", find_median(numbers))
print("Mode:", find_mode(numbers))
```

```
Mean: 5.5
Median: 5.0
Mode: 2
```

AIM 5 - Write a Python program to reverse only the vowels in a given string, keeping other characters in their original positions.


lab5.ipynb
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab5.ipynb > # Example usage
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Generate + Code + Markdown

```python
def reverse_vowels(s):
    vowels = 'aeiouAEIOU'
    s = list(s)   # convert string to list for easy swapping
    i, j = 0, len(s) - 1

    while i < j:
        # Move i forward until we find a vowel
        if s[i] not in vowels:
            i += 1
            continue

        # Move j backward until we find a vowel
        if s[j] not in vowels:
            j -= 1
            continue

        # Swap the vowels
        s[i], s[j] = s[j], s[i]
        i += 1
        j -= 1

    return "".join(s)
```

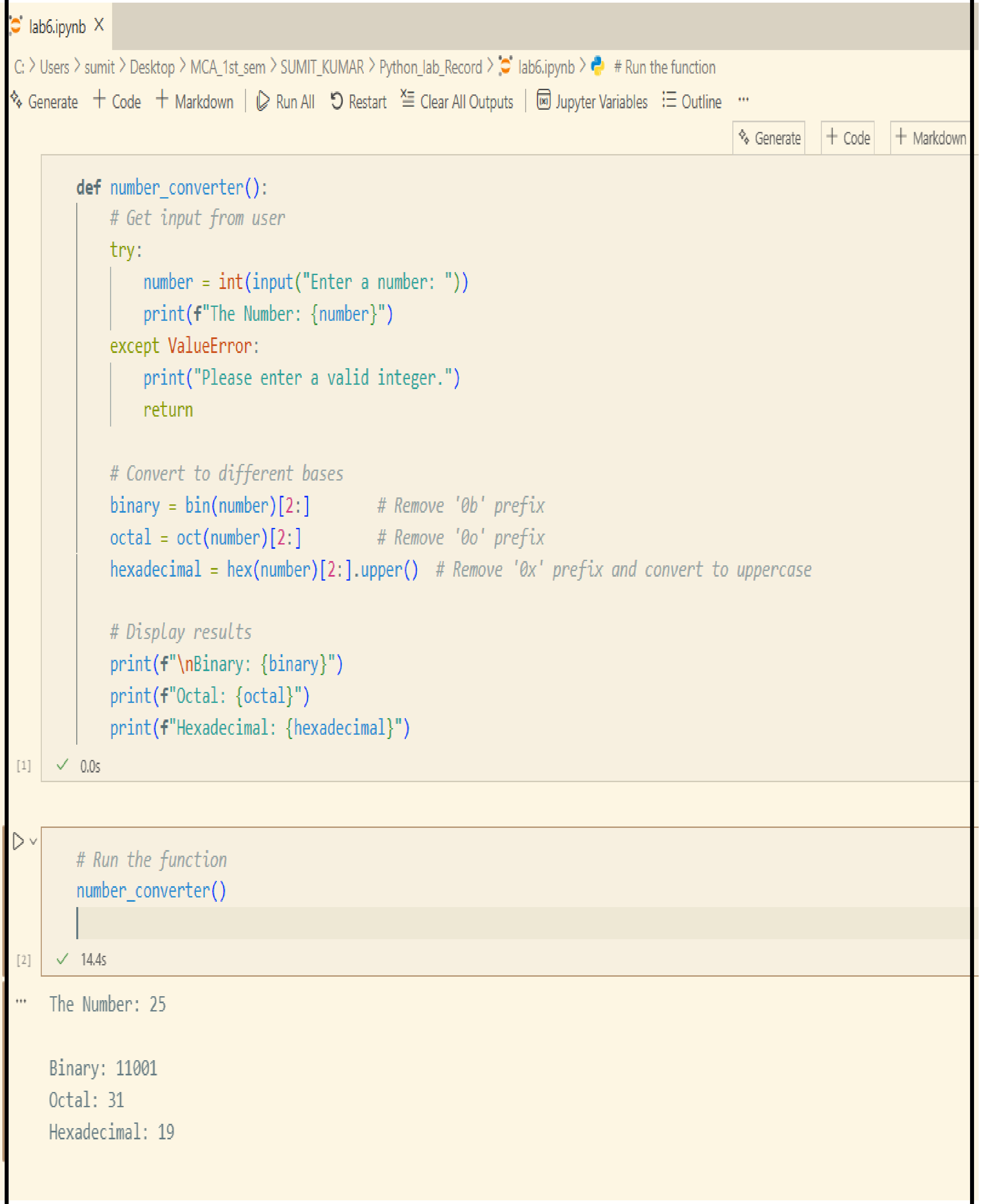
[1] ✓ 0.0s

```python
# Example usage
text = input("Enter a string: ")
print(f"The String: {text}")
result = reverse_vowels(text)
print("String after reversing vowels:", result)
```

[2] ✓ 29.7s

```
The String: Education empowers minds
String after reversing vowels: idecoteon impawurs mEnds
```

AIM 6 - Create a script that takes an integer and displays its binary, octal, and hexadecimal representations neatly formatted.


lab6.ipynb X
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab6.ipynb > # Run the function
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Generate + Code + Markdown

```python
def number_converter():
    # Get input from user
    try:
        number = int(input("Enter a number: "))
        print(f"The Number: {number}")
    except ValueError:
        print("Please enter a valid integer.")
        return

    # Convert to different bases
    binary = bin(number)[2:]        # Remove '0b' prefix
    octal = oct(number)[2:]         # Remove '0o' prefix
    hexadecimal = hex(number)[2:].upper()  # Remove '0x' prefix and convert to uppercase

    # Display results
    print(f"\nBinary: {binary}")
    print(f"Octal: {octal}")
    print(f"Hexadecimal: {hexadecimal}")
```

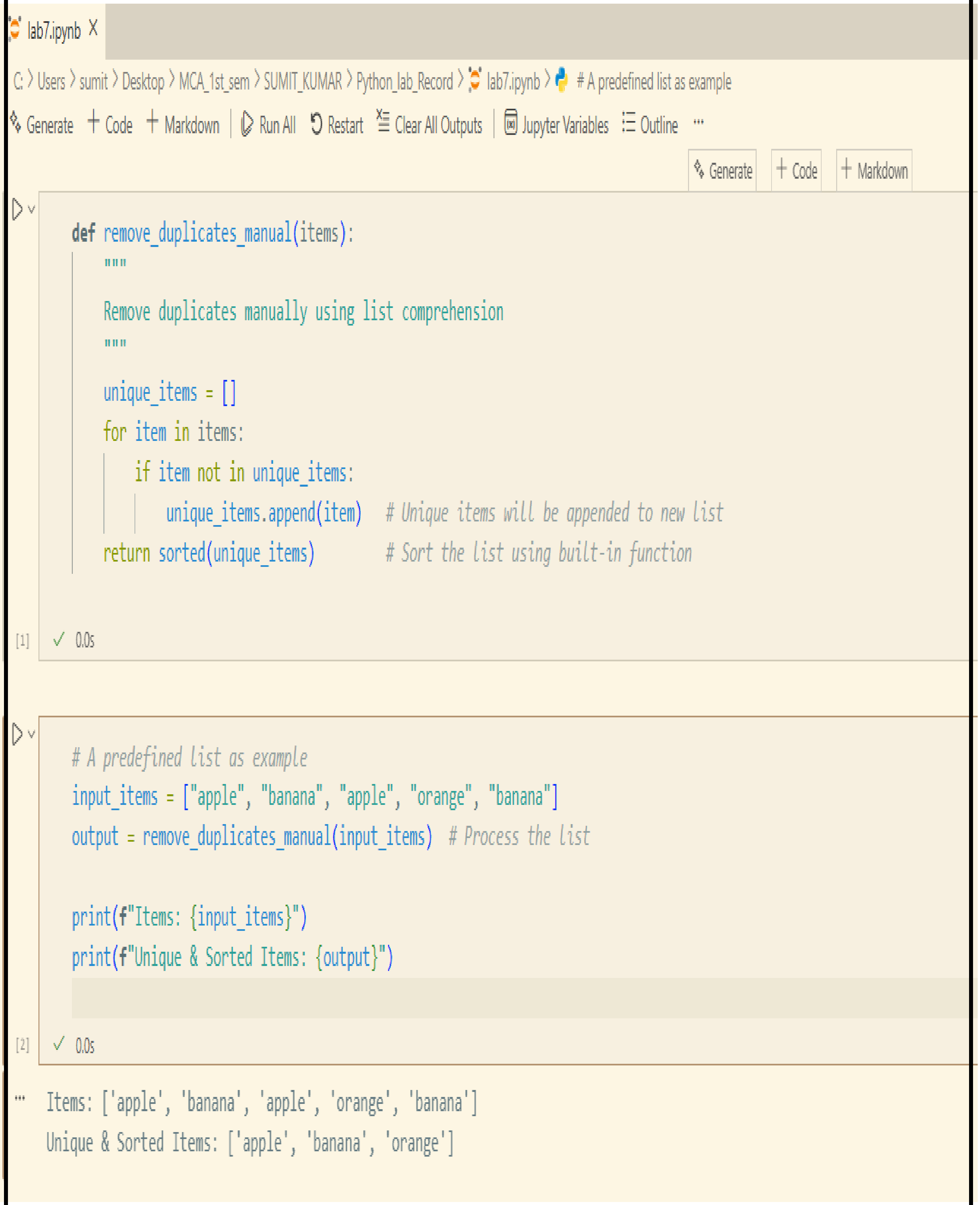
[1] ✓ 0.0s

```python
# Run the function
number_converter()
```

[2] ✓ 14.4s

```
The Number: 25

Binary: 11001
Octal: 31
Hexadecimal: 19
```

AIM 7 - Given a list of items (possibly with duplicates), write a program that removes duplicates and displays the sorted list.

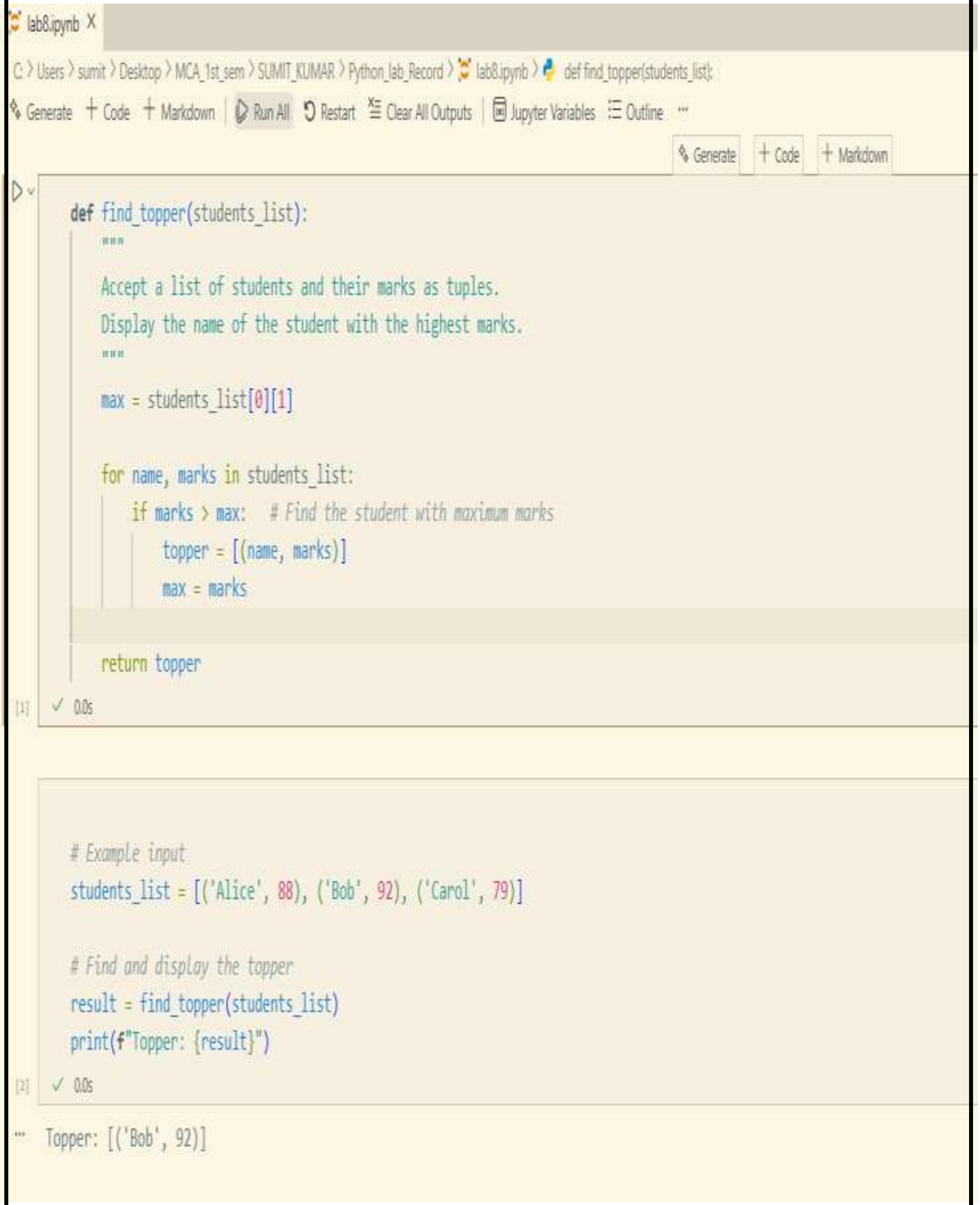

lab7.ipynb X
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab7.ipynb > # A predefined list as example
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Generate + Code + Markdown
[1] ✓ 0.0s
[2] ✓ 0.0s

```python
def remove_duplicates_manual(items):
    """
    Remove duplicates manually using list comprehension
    """
    unique_items = []
    for item in items:
        if item not in unique_items:
            unique_items.append(item)   # Unique items will be appended to new list
    return sorted(unique_items)         # Sort the list using built-in function
```

```python
# A predefined list as example
input_items = ["apple", "banana", "apple", "orange", "banana"]
output = remove_duplicates_manual(input_items)  # Process the list

print(f"Items: {input_items}")
print(f"Unique & Sorted Items: {output}")
```

```
Items: ['apple', 'banana', 'apple', 'orange', 'banana']
Unique & Sorted Items: ['apple', 'banana', 'orange']
```

AIM 8 - Accept a list of students and their marks as tuples. Display the name of the student with the highest marks.


lab8.ipynb X
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab8.ipynb > def find_topper(students_list):
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline ...
Generate + Code + Markdown

```python
def find_topper(students_list):
    """
    Accept a list of students and their marks as tuples.
    Display the name of the student with the highest marks.
    """
    max = students_list[0][1]

    for name, marks in students_list:
        if marks > max:   # Find the student with maximum marks
            topper = [(name, marks)]
            max = marks

    return topper
```

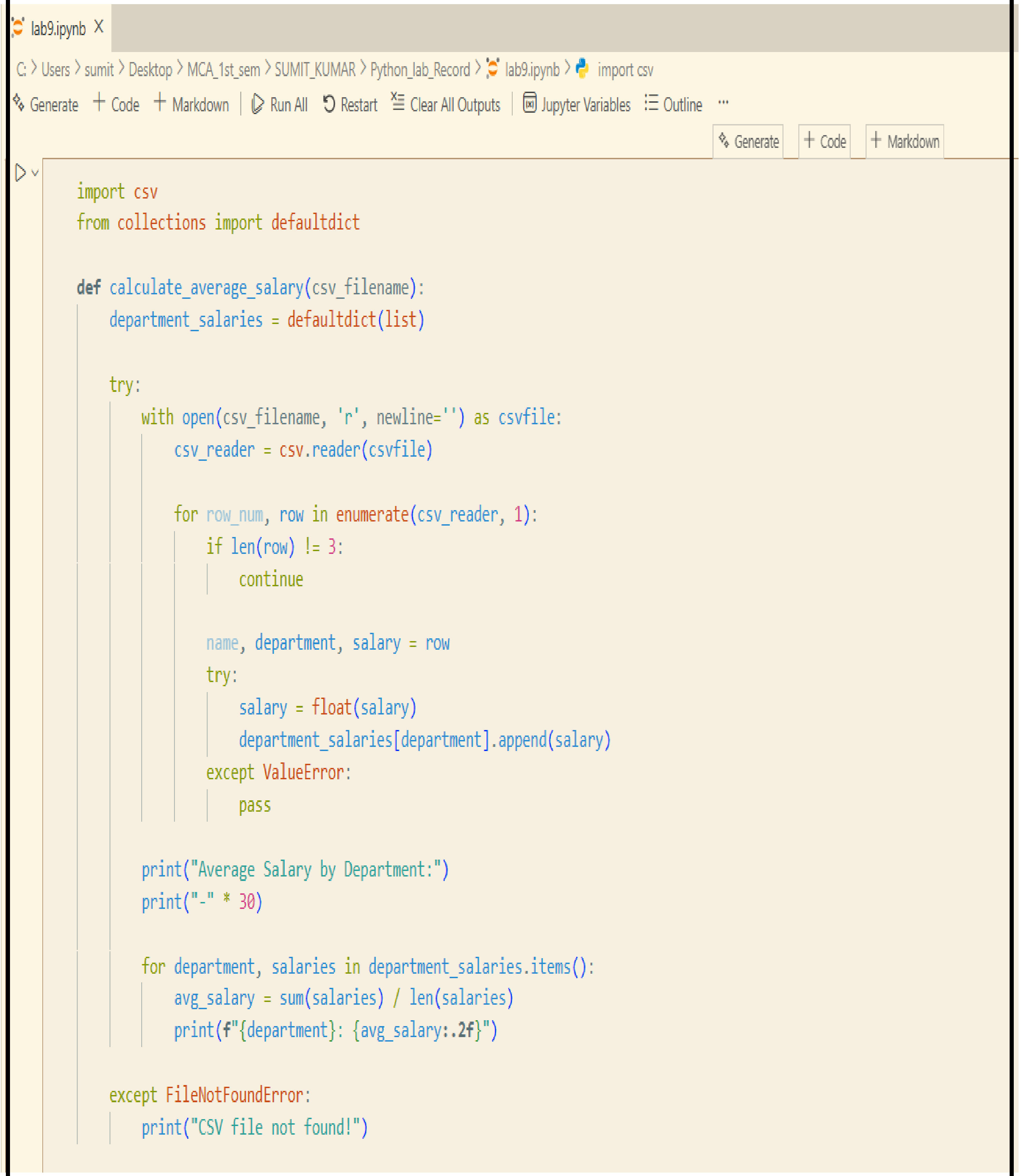
[1] ✓ 0.0s

```python
# Example input
students_list = [('Alice', 88), ('Bob', 92), ('Carol', 79)]

# Find and display the topper
result = find_topper(students_list)
print(f"Topper: {result}")
```

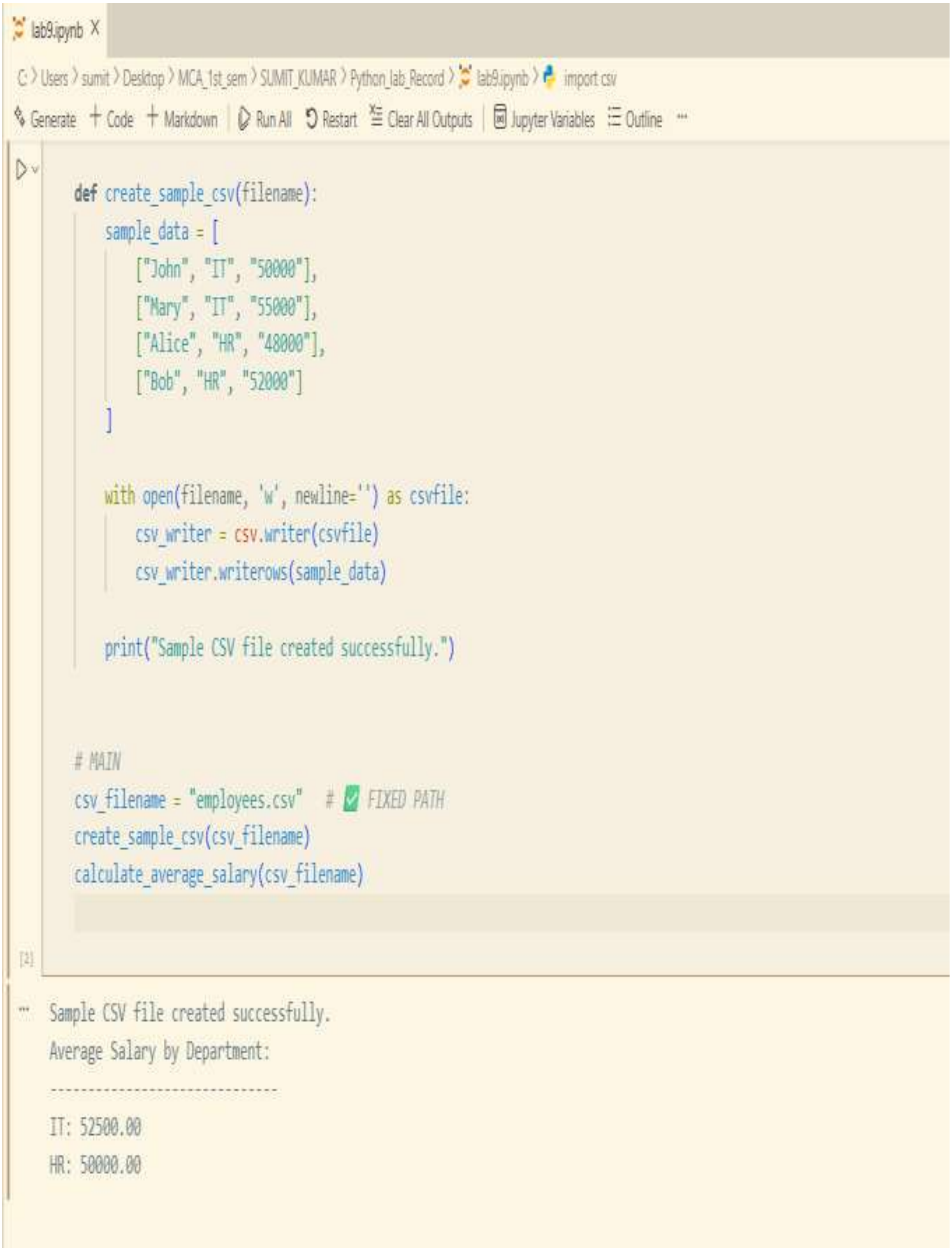
[2] ✓ 0.0s

```
Topper: [('Bob', 92)]
```

AIM 9 - Read data from a CSV file containing employee details (name, department, salary) and display the average salary by department.


lab9.ipynb X
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab9.ipynb > import csv
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Generate + Code + Markdown

```python
import csv
from collections import defaultdict

def calculate_average_salary(csv_filename):
    department_salaries = defaultdict(list)

    try:
        with open(csv_filename, 'r', newline='') as csvfile:
            csv_reader = csv.reader(csvfile)

            for row_num, row in enumerate(csv_reader, 1):
                if len(row) != 3:
                    continue

                name, department, salary = row
                try:
                    salary = float(salary)
                    department_salaries[department].append(salary)
                except ValueError:
                    pass

        print("Average Salary by Department:")
        print("-" * 30)

        for department, salaries in department_salaries.items():
            avg_salary = sum(salaries) / len(salaries)
            print(f"{department}: {avg_salary:.2f}")

    except FileNotFoundError:
        print("CSV file not found!")
```


lab9.ipynb
C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > Python_lab_Record > lab9.ipynb > import csv
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline

```python
def create_sample_csv(filename):
    sample_data = [
        ["John", "IT", "50000"],
        ["Mary", "IT", "55000"],
        ["Alice", "HR", "48000"],
        ["Bob", "HR", "52000"]
    ]

    with open(filename, 'w', newline='') as csvfile:
        csv_writer = csv.writer(csvfile)
        csv_writer.writerows(sample_data)

    print("Sample CSV file created successfully.")


# MAIN
csv_filename = "employees.csv"   # ✅ FIXED PATH
create_sample_csv(csv_filename)
calculate_average_salary(csv_filename)
```

[1]

```
Sample CSV file created successfully.
Average Salary by Department:
------------------------------
IT: 52500.00
HR: 50000.00
```