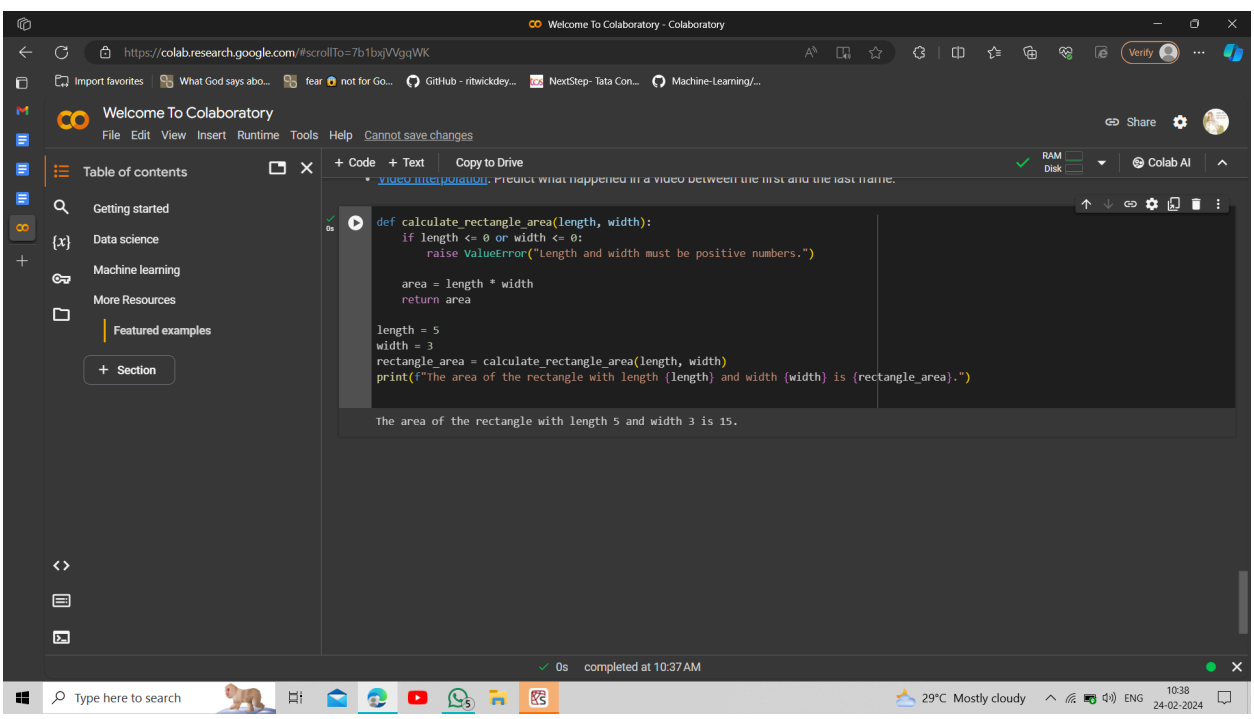
SMART INTERNZ - APSCHE

AI / ML Training

Assessment

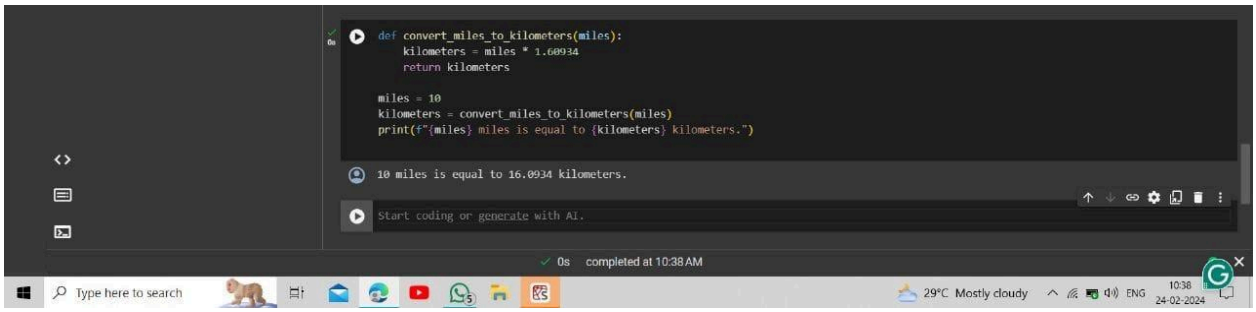
1. **Write a Python program to calculate the area of a rectangle given its length and width.**

```
def calculate_rectangle_area(length, width):
    if length <= 0 or width <= 0:
        raise ValueError("Length and width must be positive numbers.")

    area = length * width
    return area

length = 5
width = 3
rectangle_area = calculate_rectangle_area(length, width)
print(f"The area of the rectangle with length {length} and width {width} is {rectangle_area}.")
```



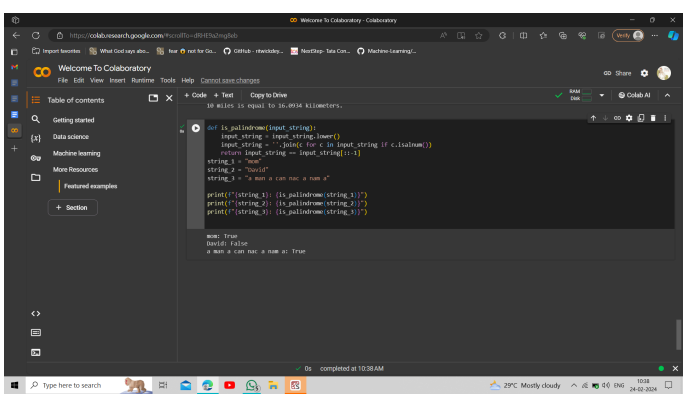
2. **Write a program to convert miles to kilometer.**

```
def convert_miles_to_kilometers(miles):
    kilometers = miles * 1.60934
    return kilometers

miles = 10
kilometers = convert_miles_to_kilometers(miles)
print(f"{miles} miles is equal to {kilometers} kilometers.")
```



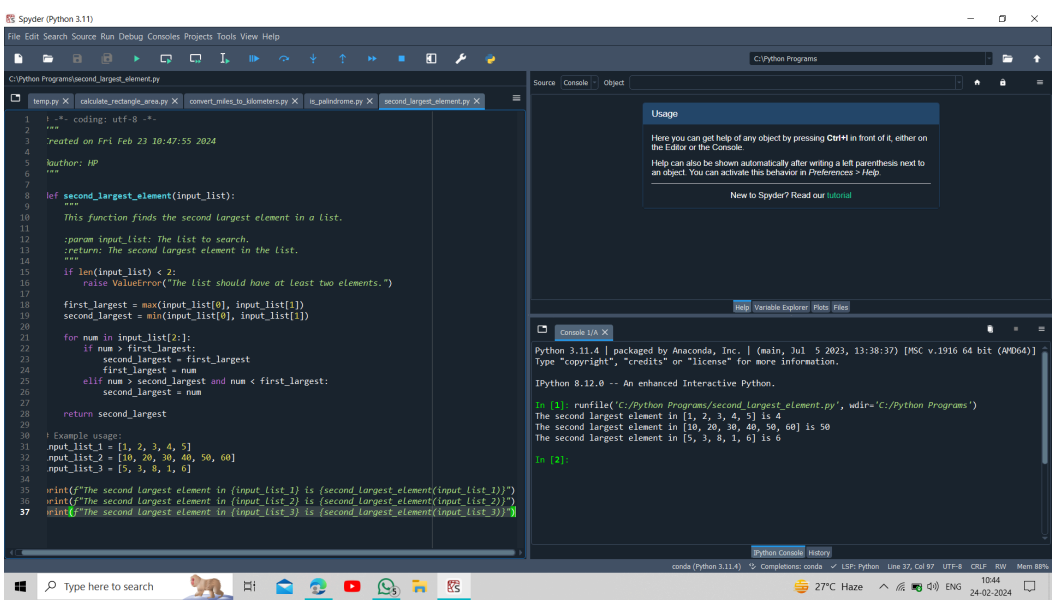
3. **Write a function to check if a given string is a palindrome.**

```
def is_palindrome(input_string):
    input_string = input_string.lower()
    input_string = ''.join(c for c in input_string if c.isalnum())
    return input_string == input_string[::-1]
string_1 = "mom"
string_2 = "David"
string_3 = "a man a can nac a nam a"

print(f"{string_1}: {is_palindrome(string_1)}")
print(f"{string_2}: {is_palindrome(string_2)}")
print(f"{string_3}: {is_palindrome(string_3)}")
```



4. **Write a Python program to find the second largest element in a list.**

```
def second_largest_element(input_list):
    if len(input_list) < 2:
      raise ValueError("The list should have at least two elements.")

  first_largest = max(input_list[0], input_list[1])
  second_largest = min(input_list[0], input_list[1])

  for num in input_list[2:]:
      if num > first_largest:
          second_largest = first_largest
          first_largest = num
      elif num > second_largest and num < first_largest:
          second_largest = num

  return second_largest

input_list_1 = [1, 2, 3, 4, 5]
input_list_2 = [10, 20, 30, 40, 50, 60]
input_list_3 = [5, 3, 8, 1, 6]

print(f"The second largest element in {input_list_1} is
{second_largest_element(input_list_1)}")
print(f"The second largest element in {input_list_2} is
{second_largest_element(input_list_2)}")
print(f"The second largest element in {input_list_3} is
{second_largest_element(input_list_3)}")
```



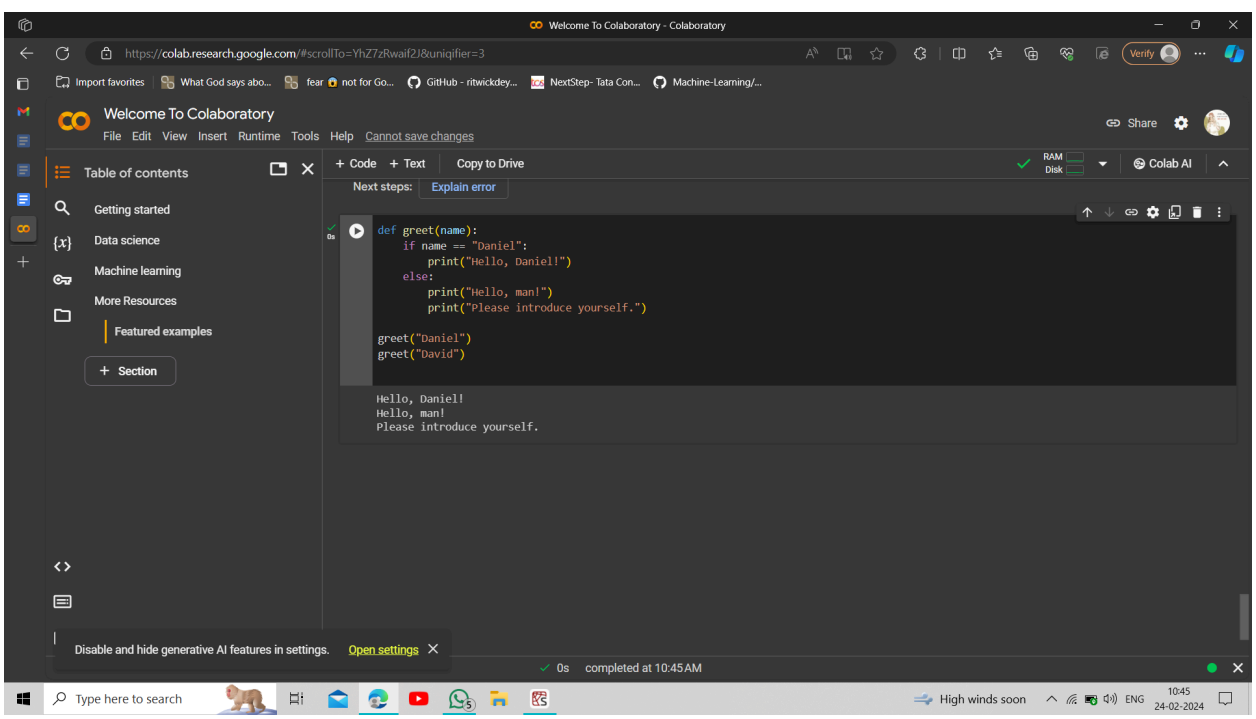
5. **Explain what indentation means in Python.**

In Python, indentation is used to define the scope and structure of the code. It is a crucial part of the Python syntax, as it helps the interpreter understand the logical grouping of statements and the relationship between them. In other programming languages, curly braces {} or keywords like begin and end are used to define blocks of code. However, Python uses indentation to indicate the beginning and end of a block.

Here's an example of how indentation is used in Python:

```
def greet(name):
    if name == "Daniel":
        print("Hello, Daniel!")
    else:
        print("Hello, man!")
        print("Please introduce yourself.")

greet("Daniel")
```

greet("David")



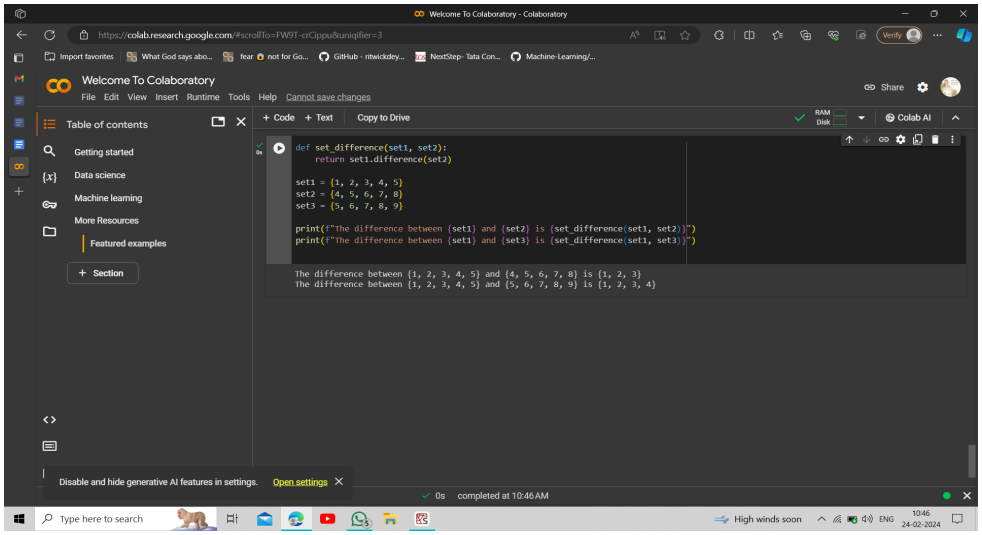
In this example, the** greet **function definition,** if **statement, and** else **block are all indented. The first** print **statement is indented further, indicating that it is part of the** if **block. The second** print **statement in the** else **block is indented even further, showing that it is nested within the** else **block.

When the Python interpreter encounters an indented block, it considers the statements within that block to be related to the statement that precedes it. In this example, the** if **statement and its corresponding** print **statement form a block, and the** else **block and its corresponding** print **statements from another block.

Proper indentation is essential in Python, as it directly affects the program's logic and behavior. Incorrect indentation can lead to syntax errors or unexpected results. It is a best practice to use consistent indentation throughout the code to make it more readable and maintainable.

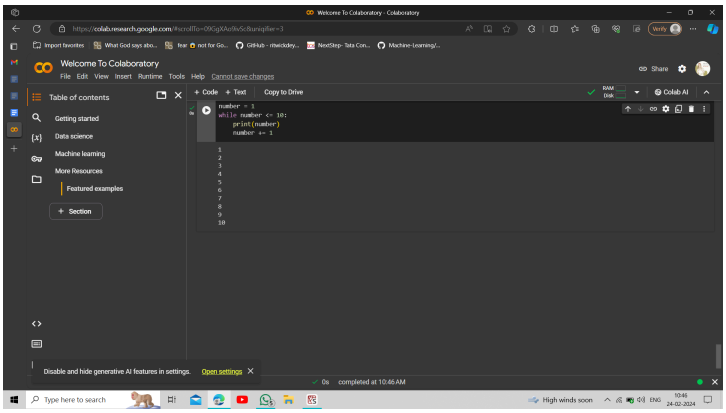
6. **Write a program to perform set difference operation.**

```
def set_difference(set1, set2):
    return set1.difference(set2)
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
set3 = {5, 6, 7, 8, 9}
print(f"The difference between {set1} and {set2} is {set_difference(set1, set2)}")
print(f"The difference between {set1} and {set3} is {set_difference(set1, set3)}")
```



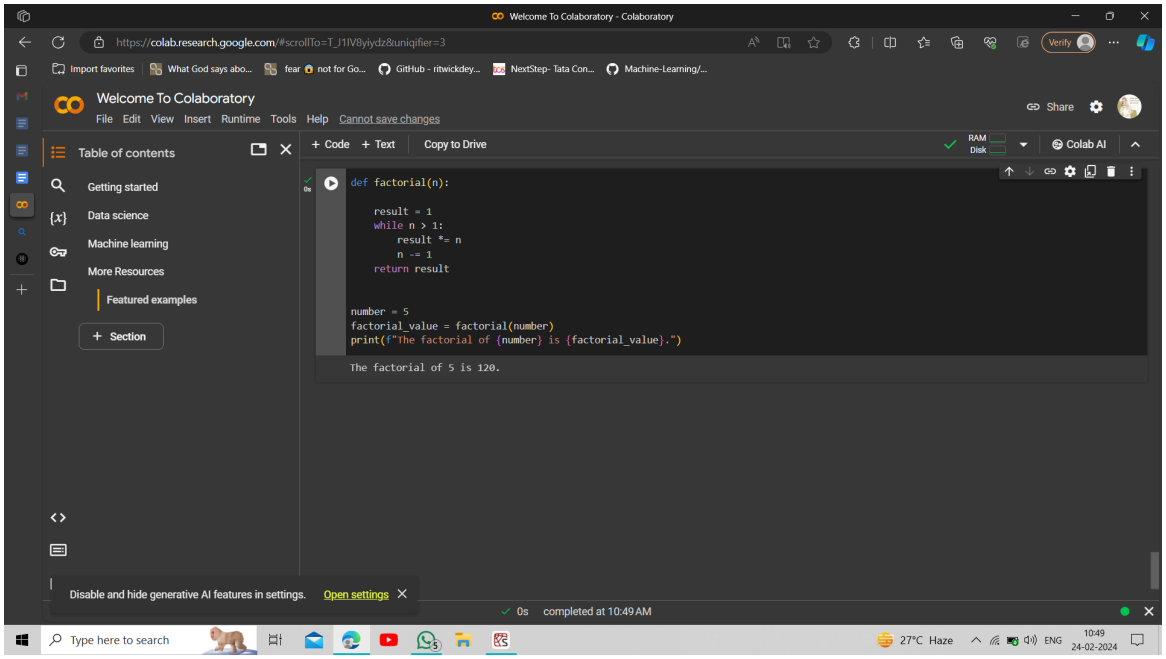
7. **Write a Python program to print numbers from 1 to 10 using a while loop.**

```
number = 1
while number <= 10:
    print(number)
    number += 1
```



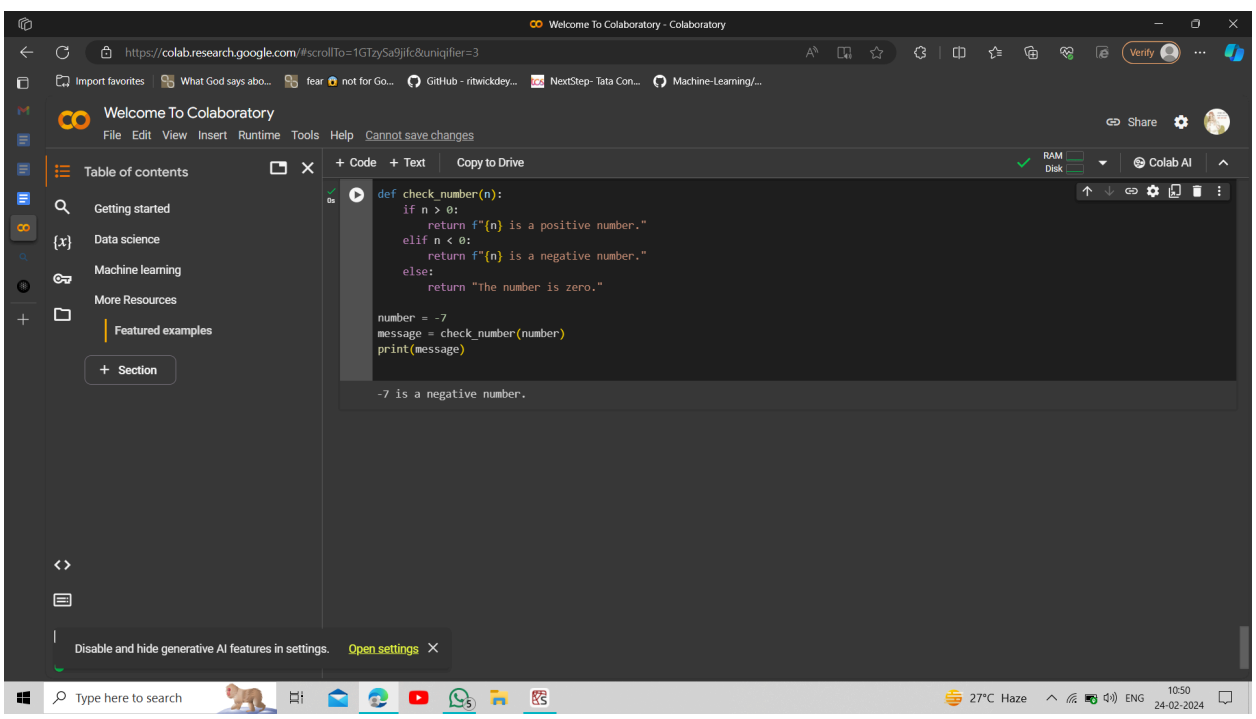
8. **Write a program to calculate the factorial of a number using a while loop.**

```
def factorial(n):
    result = 1
    while n > 1:
        result *= n
        n -= 1
    return result
number = 5
factorial_value = factorial(number)
print(f"The factorial of {number} is {factorial_value}.")
```



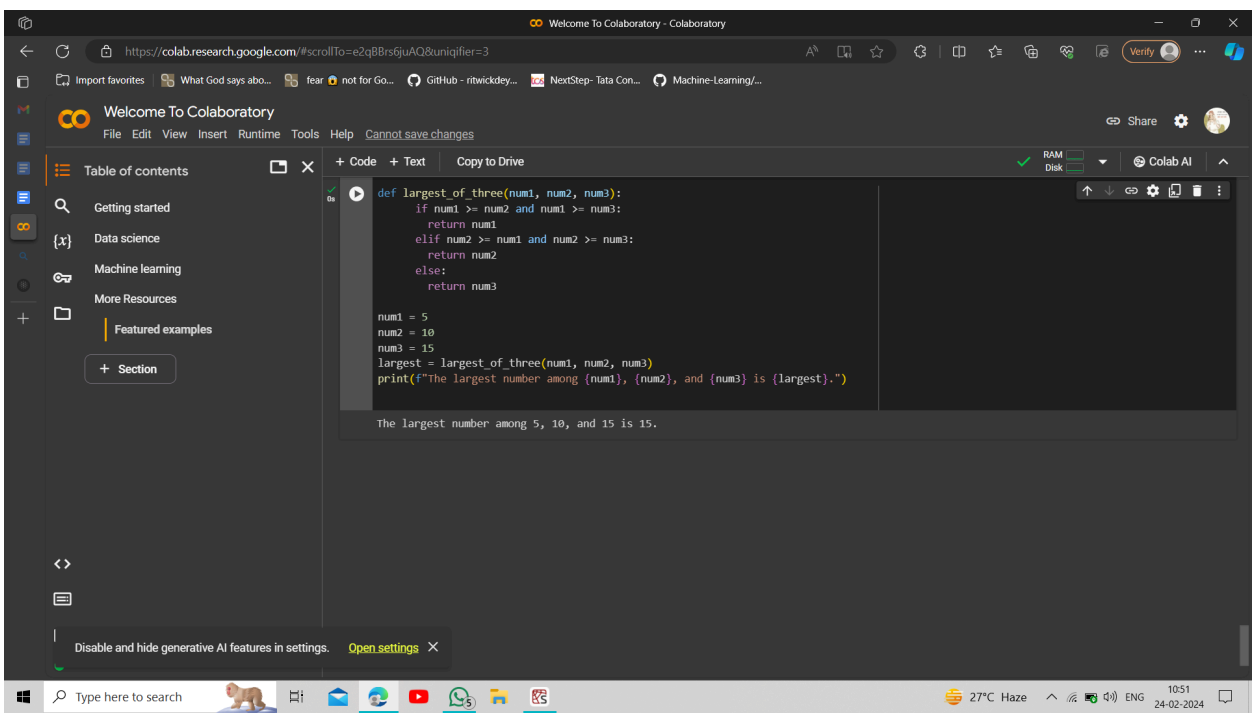
9. **Write a Python program to check if a number is positive, negative, or zero using if-elif-else statements.**

```
def check_number(n):
    if n > 0:
        return f"{n} is a positive number."
    elif n < 0:
        return f"{n} is a negative number."
    else:
        return "The number is zero."

number = 5
message = check_number(number)
print(message)
```



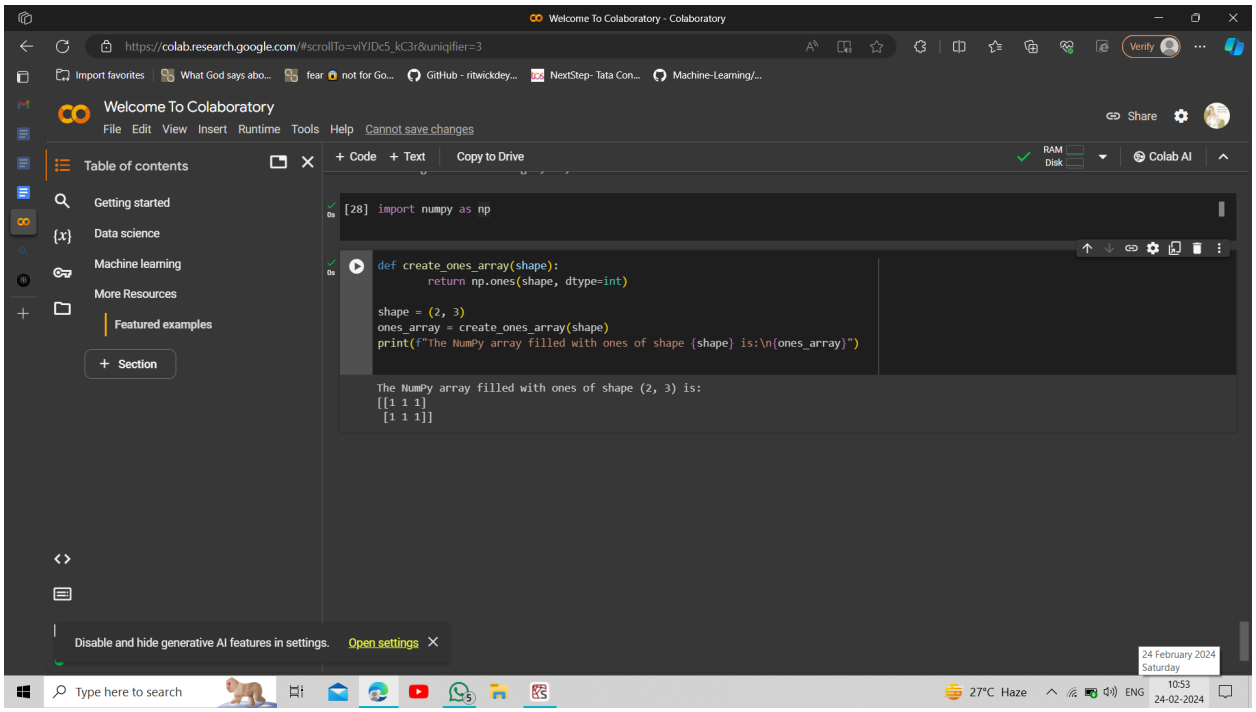
10. **Write a program to determine the largest among three numbers using conditional statements.**

```
def largest_of_three(num1, num2, num3):
    if num1 >= num2 and num1 >= num3:
      return num1
  elif num2 >= num1 and num2 >= num3:
      return num2
  else:
      return num3

num1 = 5
num2 = 10
num3 = 15
largest = largest_of_three(num1, num2, num3)
print(f"The largest number among {num1}, {num2}, and {num3} is {largest}.")
```



11. **Write a Python program to create a numpy array filled with ones of given shape.**

```
//bash
pip install numpy
//python
import numpy as np
def create_ones_array(shape):
    return np.ones(shape, dtype=int)

shape = (2, 3)
ones_array = create_ones_array(shape)
print(f"The NumPy array filled with ones of shape {shape} is:\n{ones_array}")
```



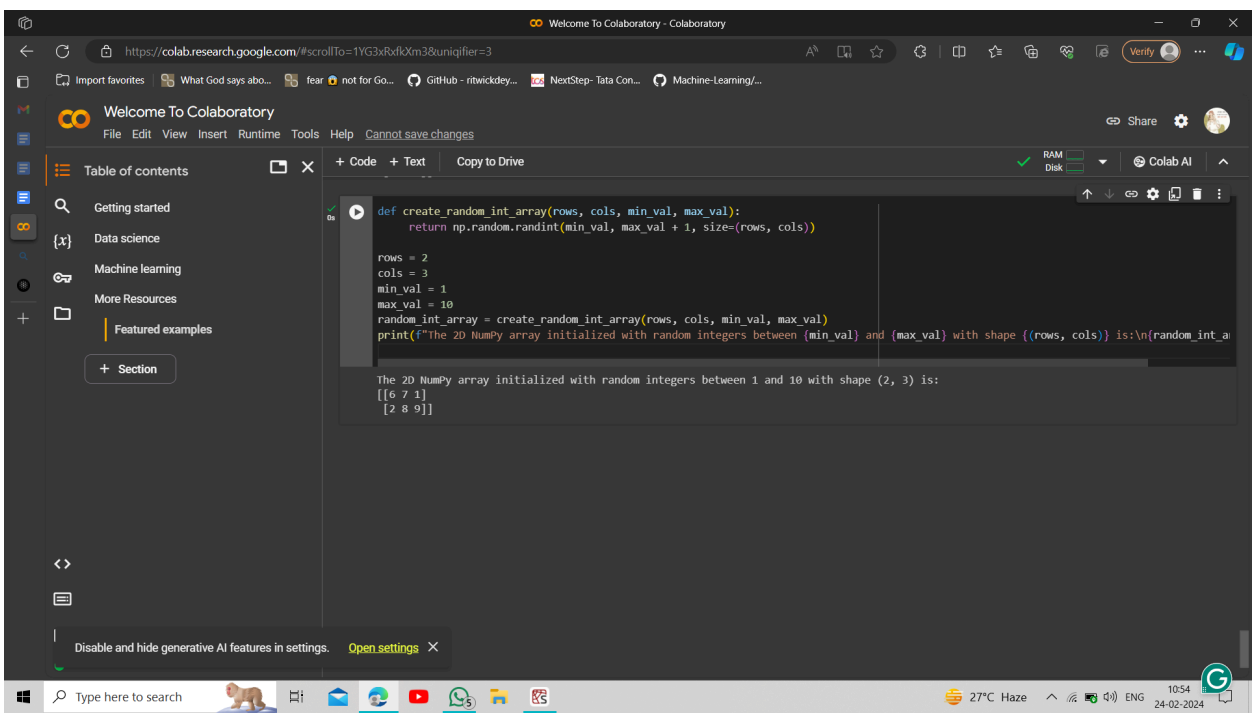
12. **Write a program to create a 2D numpy array initialized with random integers.**

//bash

pip install numpy

//python

```
import numpy as np
def create_random_int_array(rows, cols, min_val, max_val):
    return np.random.randint(min_val, max_val + 1, size=(rows, cols))

rows = 2
cols = 3
min_val = 1
max_val = 10
random_int_array = create_random_int_array(rows, cols, min_val, max_val)
print(f"The 2D NumPy array initialized with random integers between {min_val} and {max_val} with shape {(rows, cols)} is:\n{random_int_array}")
```



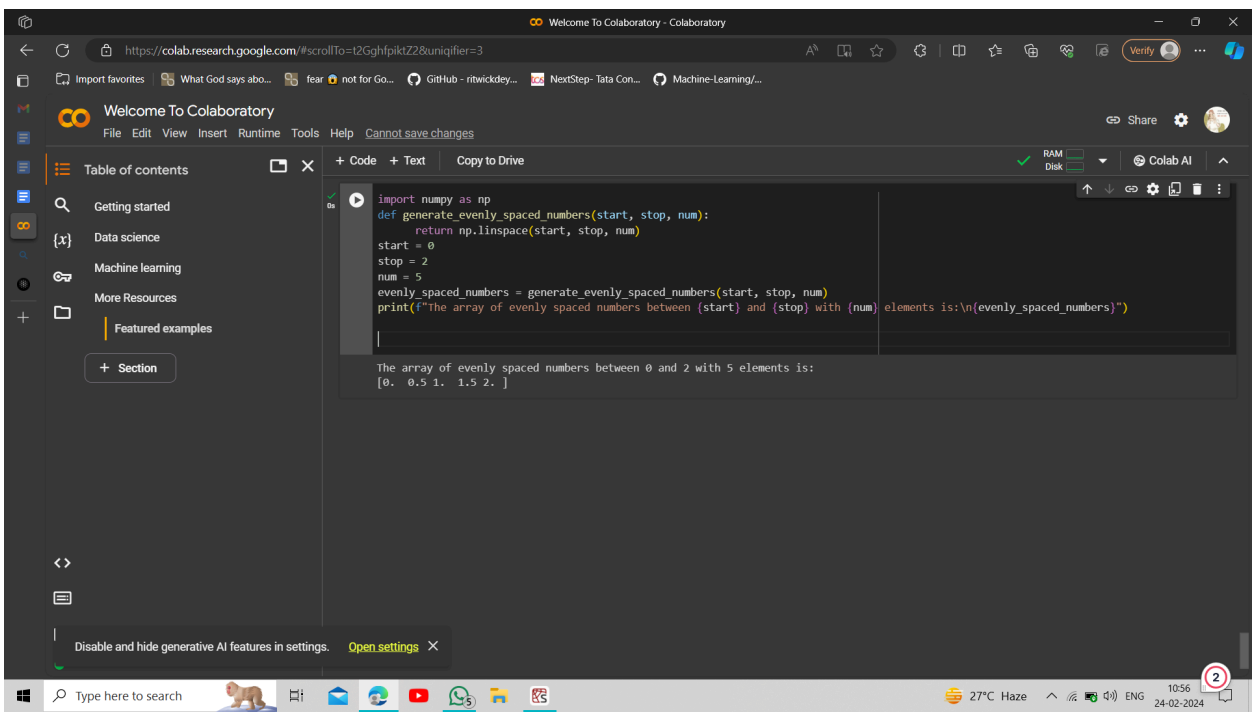
13. **Write a Python program to generate an array of evenly spaced numbers over a specified range using linspace.**

//bash

pip install numpy

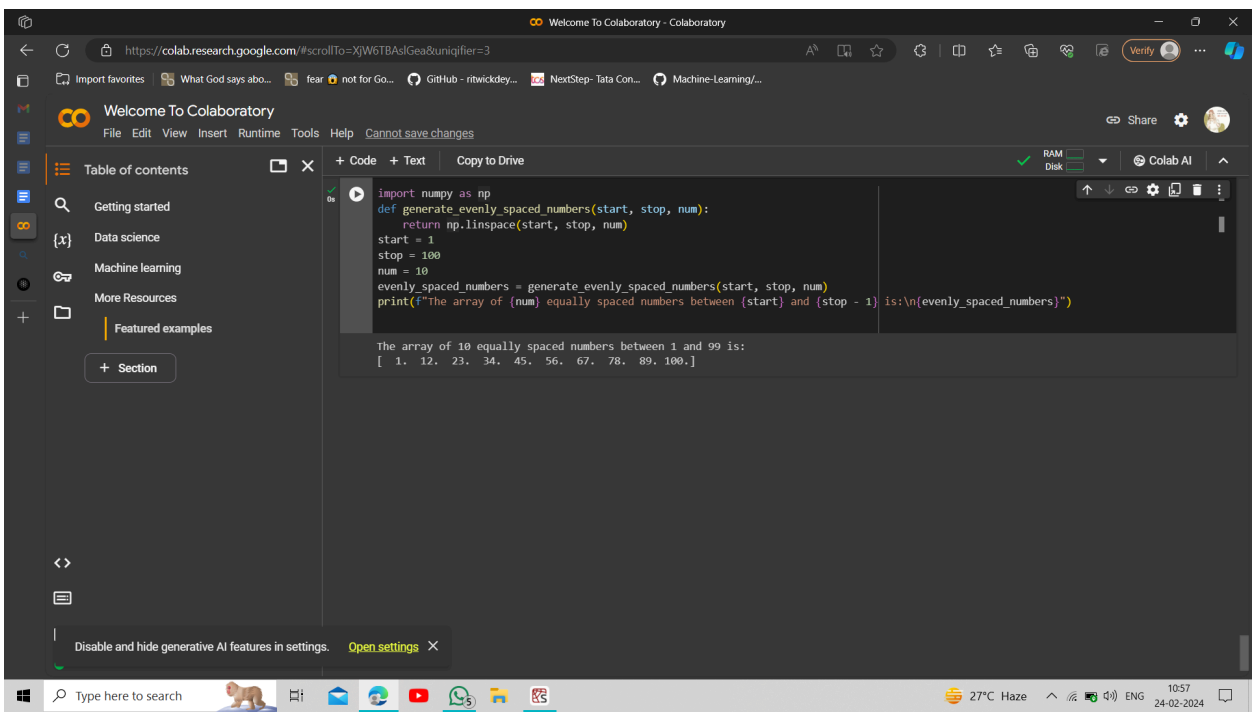
//python

```
import numpy as np
def generate_evenly_spaced_numbers(start, stop, num):
    return np.linspace(start, stop, num)
start = 0
stop = 2
num = 5
evenly_spaced_numbers = generate_evenly_spaced_numbers(start, stop, num)
print(f"The array of evenly spaced numbers between {start} and {stop} with {num} elements is:\n{evenly_spaced_numbers}")
```



14. **Write a program to generate an array of 10 equally spaced values between 1 and 100 using linspace.**

```
//bash
pip install numpy
//python
import numpy as np
def generate_evenly_spaced_numbers(start, stop, num):
    return np.linspace(start, stop, num)
start = 1
stop = 100
num = 10
evenly_spaced_numbers = generate_evenly_spaced_numbers(start, stop, num)
print(f"The array of {num} equally spaced numbers between {start} and {stop - 1}
is:\n{evenly_spaced_numbers}")
```



15. Write a Python program to create an array containing even numbers from 2 to 20 using arrange.

```
//bash
pip install numpy
//python
import numpy as np
def create_even_numbers(start, stop, step=1):
    numbers = np.arange(start, stop, step)
    even_numbers = numbers[numbers % 2 == 0]
    return even_numbers

start = 2
stop = 21
even_numbers = create_even_numbers(start, stop)
print(f"The array of even numbers between {start} and {stop - 1} is:\n{even_numbers}")
```