

ASSIGNMENT – 3.3

NAME : BOMMAKANTI SRINITHI

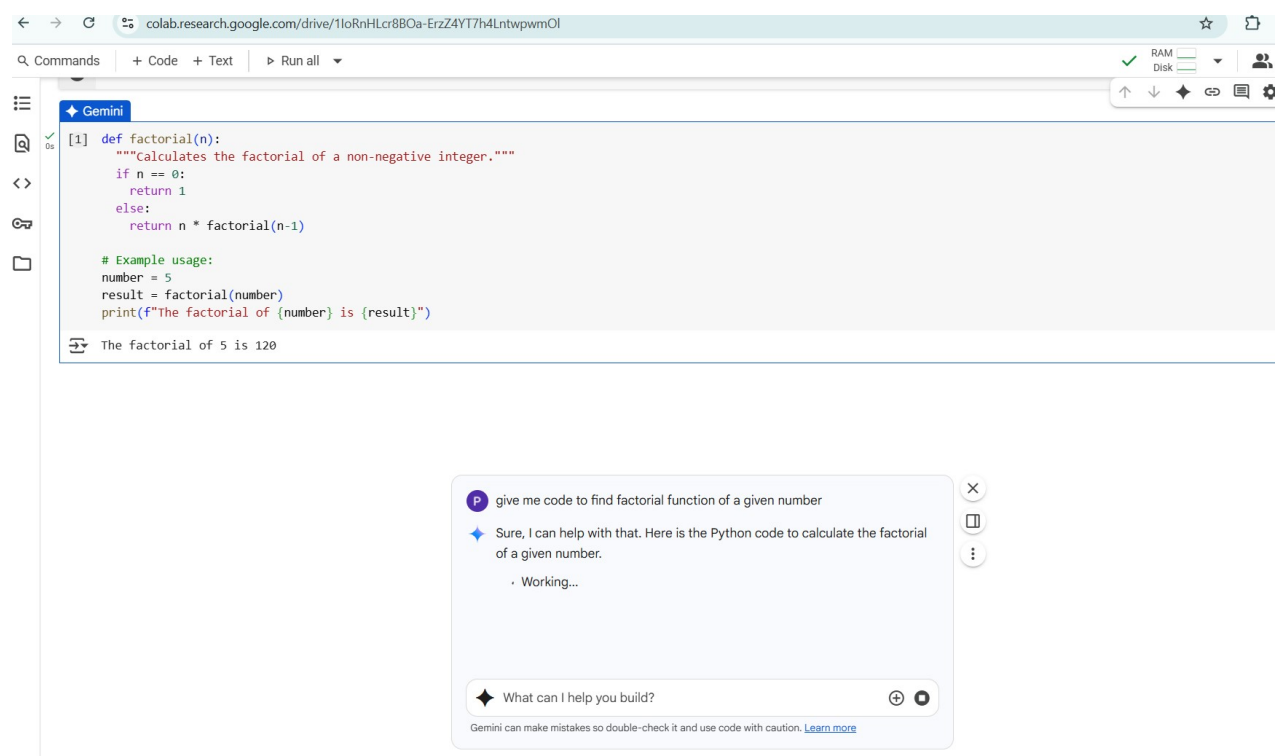
BATCH : 01

ROLL NO : 2403A51413

COURSE : AI ASSISTED CODING

TASK 1:

PROMPT 1:



The screenshot displays a Google Colab notebook interface. The top toolbar includes navigation icons, a search bar, and tabs for 'Commands', '+ Code', '+ Text', and 'Run all'. On the right, there are icons for RAM, Disk, and user profile. The left sidebar shows a file explorer with a 'Gemini' tab selected. The main code cell contains a Python function definition for factorial, an example usage with number=5, and a print statement. Below the code, the output shows 'The factorial of 5 is 120'. At the bottom, a Gemini chat window is open, showing a user prompt 'give me code to find factorial function of a given number', a Gemini response 'Sure, I can help with that. Here is the Python code to calculate the factorial of a given number.', and a 'Working...' status. The chat window also includes a text input field with the placeholder 'What can I help you build?' and a disclaimer: 'Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)'.

```
[1] def factorial(n):  
    """Calculates the factorial of a non-negative integer."""  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n-1)  
  
# Example usage:  
number = 5  
result = factorial(number)  
print(f"The factorial of {number} is {result}")
```

The factorial of 5 is 120

P give me code to find factorial function of a given number

◆ Sure, I can help with that. Here is the Python code to calculate the factorial of a given number.

• Working...

◆ What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

PROMPT 2:

```
✓ [2] def factorial_loop(n):
0s """Calculates the factorial of a non-negative integer using a loop."""
    if n < 0:
        return "Factorial is not defined for negative numbers"
    elif n == 0:
        return 1
    else:
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result

# Example usage:
number = 5
result = factorial_loop(number)
print(f"The factorial of {number} using a loop is {result}")
```

The factorial of 5 using a loop is 120

give me code to create a factorial program using a loop example usage take 5

Okay, here is a version of the factorial function that uses a loop, along with an example using the number 5.

I've provided two different ways to calculate the factorial of a number. Would you like to:

- Compare the performance of the two methods?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal 1:53 PM Python 3

PROMPT 3:

```
✓ [3] import math
0s
number = 5
result = math.factorial(number)
print(f"The factorial of {number} using math.factorial() is {result}")
```

The factorial of 5 using math.factorial() is 120

give me code to find factorial function using the built in math module take example as 5

You can indeed use Python's built-in `math` module to calculate factorials. Here's how to do it for the number 5:

I've shown you three different ways to calculate factorials in Python: using recursion, a loop, and the built-in `math` module.

What would you like to do next? We could:

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal 2:06 PM Python 3

TASK 2:

The code editor displays a Python function `sort_descending()` that takes a list of integers as input and sorts them in descending order. The function uses a `while` loop to repeatedly prompt the user for input until a valid list of integers is provided. It then sorts the list and prints the result.

```
def sort_descending():
    """
    Takes a list of integers as input from the user and sorts them in descending order.
    """
    while True:
        input_str = input("Enter a list of integers separated by spaces: ")
        try:
            # Convert the input string to a list of integers
            numbers = [int(x) for x in input_str.split()]
            break # Exit the loop if conversion is successful
        except ValueError:
            print("Invalid input. Please enter only integers separated by spaces.")

    # Sort the list in descending order
    numbers.sort(reverse=True)

    print("Sorted list in descending order:", numbers)

# Call the function to get input and sort
sort_descending()
```

The terminal output shows the user entering the list `55 99 76 34 22 11` and the function outputting the sorted list `[99, 76, 55, 34, 22, 11]`.

The Gemini chat interface on the right shows a user prompt: "give me code to write a function that sorts a list of integers in descending order by taking input from the user". The Gemini response provides a detailed explanation of the function and offers suggestions for further exploration, such as modifying the function to sort in ascending order, adding error handling for non-integer input, and exploring different sorting algorithms.

TASK 3:

The code editor displays a Python function `calculate_power_bill(consumption_kwh)` that calculates the power bill based on electricity consumption in kWh. The function uses a tiered pricing system with three tiers: Tier 1 (0-100 kWh at \$0.12/kWh), Tier 2 (100-300 kWh at \$0.15/kWh), and Tier 3 (above 300 kWh at \$0.20/kWh). It prompts the user for consumption and calculates the bill amount.

```
def calculate_power_bill(consumption_kwh):
    """
    Calculates the power bill based on consumption in kWh.
    """
    # Define billing tiers and rates (example rates, these would vary by location/provider)
    tier1_limit = 100 # kWh
    tier1_rate = 0.12 # per kWh

    tier2_limit = 300 # kWh
    tier2_rate = 0.15 # per kWh

    tier3_rate = 0.20 # per kWh (for consumption above tier2_limit)

    bill_amount = 0

    if consumption_kwh <= tier1_limit:
        bill_amount = consumption_kwh * tier1_rate
    elif consumption_kwh <= tier2_limit:
        bill_amount = (tier1_limit * tier1_rate) + ((consumption_kwh - tier1_limit) * tier2_rate)
    else:
        bill_amount = (tier1_limit * tier1_rate) + ((tier2_limit - tier1_limit) * tier2_rate) + ((consumption_kwh - tier2_limit) * tier3_rate)

    return bill_amount

# Get user input for electricity consumption
while True:
    try:
        user_consumption = float(input("Enter your electricity consumption in kWh: "))
        if user_consumption >= 0:
            break
        else:
            print("Consumption cannot be negative. Please enter a valid number.")
    except ValueError:
        print("Invalid input. Please enter a numerical value for consumption.")

# Calculate and display the bill
power_bill = calculate_power_bill(user_consumption)
print(f"Your power bill for {user_consumption} kWh is: ${power_bill:.2f}")
```

The terminal output shows the user entering `60` kWh and the function outputting a power bill of `$7.20`.

The Gemini chat interface on the right shows a user prompt: "generate python code to calculate power bill". The Gemini response provides a detailed plan for generating the Python code, including defining the function, implementing the billing logic, adding user input, and calculating the bill. It also offers suggestions for further exploration, such as adding more billing tiers, including fixed charges or taxes, saving the bill calculation results to a file, and working on a different coding problem.

TASK 4:

P generate python code using if and else for login user and registraion user

✦ I can definitely help you with that. Here is a plan to create Python code using if/else statements for user login and registration.

- **Simulate a user database:** Create a simple way to store usernames and passwords, perhaps using a dictionary.
- **Implement registration logic:**
 - Prompt the user for a new username and password.
 - Check if the username already exists in the database.
 - If the username is unique, add the new user to the database.
 - If the username exists, inform the user.
- **Implement login logic:**
 - Prompt the user for their username and password.
 - Check if the username exists in the database.
 - If the username exists, verify the password.
 - If the username and password match, grant access.
 - If the username doesn't exist or the password is incorrect, inform the user.
- **Add a main program loop:** Create a loop that allows the user to choose between registration and login.
- **Finish task:** Provide the complete Python code for the login and registration system.



Simulate a user database using a dictionary

user_database = {}

def register_user():

"""Registers a new user."""

username = input("Enter a new username: ")

if username in user_database:

print("Username already exists. Please choose a different username.")

else:

password = input("Enter a password: ")

user_database[username] = password

print("Registration successful!")

def login_user():

"""Logs in an existing user."""

username = input("Enter your username: ")

if username in user_database:

password = input("Enter your password: ")

if user_database[username] == password:

print("Login successful!")

else:

print("Incorrect password.")

else:

print("Username not found.")

Main program loop

while True:

print("\nChoose an option:")

print("1. Register")

print("2. Login")

print("3. Exit")

choice = input("Enter your choice: ")

if choice == '1':

register_user()

elif choice == '2':

login_user()

elif choice == '3':

break

Gemini X

and passwords, perhaps using a dictionary.

- Implement registration logic:
 - Prompt the user for a new username and password.
 - Check if the username already exists in the database.
 - If the username is unique, add the new user to the database.
 - If the username exists, inform the user.
- Implement login logic:
 - Prompt the user for their username and password.
 - Check if the username exists in the database.
 - If the username exists, verify the password.
 - If the username and password match, grant access.
 - If the username doesn't exist or the password is incorrect, inform the user.
- Add a main program loop: Create a loop that allows the user to choose between registration and login.
- Finish task: Provide the complete Python code for the login and registration system.

P

using this plan give me code

Great! Here is the Python code that implements the login and registration system based on the plan we created. It uses a dictionary to simulate a user database and if/else statements for the login and registration logic.

Working...

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables

Terminal

Execution (1m 12s)

Python 3

if user_database[username] == password:

print("Login successful!")

else:

print("Incorrect password.")

else:

print("Username not found.")

Main program loop

while True:

print("\nChoose an option:")

print("1. Register")

print("2. Login")

print("3. Exit")

choice = input("Enter your choice: ")

if choice == '1':

register_user()

elif choice == '2':

login_user()

elif choice == '3':

print("Exiting program.")

break

else:

print("Invalid choice. Please try again.")

Gemini X

and passwords, perhaps using a dictionary.

- Implement registration logic:
 - Prompt the user for a new username and password.
 - Check if the username already exists in the database.
 - If the username is unique, add the new user to the database.
 - If the username exists, inform the user.
- Implement login logic:
 - Prompt the user for their username and password.
 - Check if the username exists in the database.
 - If the username exists, verify the password.
 - If the username and password match, grant access.
 - If the username doesn't exist or the password is incorrect, inform the user.
- Add a main program loop: Create a loop that allows the user to choose between registration and login.
- Finish task: Provide the complete Python code for the login and registration system.

P

using this plan give me code

Great! Here is the Python code that implements the login and registration system based on the plan we created. It uses a dictionary to simulate a user database and if/else statements for the login and registration logic.

Working...

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables

Terminal

Executing (2m 1s)

Python 3

TASK 5:

PROMPT 1 :

Here are the formulas for converting between Fahrenheit and Celsius:

Fahrenheit to Celsius: $C = (F - 32) \times \frac{5}{9}$

Celsius to Fahrenheit: $F = (C \times \frac{9}{5}) + 32$

Gemini

```
def fahrenheit_to_celsius(fahrenheit):  
    """Converts Fahrenheit to Celsius."""  
    celsius = (fahrenheit - 32) * 5/9  
    return celsius  
  
def celsius_to_fahrenheit(celsius):  
    """Converts Celsius to Fahrenheit."""  
    fahrenheit = (celsius * 9/5) + 32  
    return fahrenheit  
  
# Example usage:  
fahrenheit_temp = 68  
celsius_temp = fahrenheit_to_celsius(fahrenheit_temp)  
print(f"{fahrenheit_temp}°F is equal to {celsius_temp:.2f}°C")  
  
celsius_temp_2 = 20  
fahrenheit_temp_2 = celsius_to_fahrenheit(celsius_temp_2)  
print(f"{celsius_temp_2}°C is equal to {fahrenheit_temp_2:.2f}°F")
```

Gemini X

P give me code to convert temperature

Sure, I can provide you with code to convert temperatures between Fahrenheit and Celsius. I'll also include the formulas for your reference.

👍 🗨

PROMPT 2:

```
def celsius_to_fahrenheit(celsius):  
    """Converts Celsius to Fahrenheit."""  
    fahrenheit = (celsius * 9/5) + 32  
    return fahrenheit  
  
# Example usage:  
celsius_temp = 25  
fahrenheit_temp = celsius_to_fahrenheit(celsius_temp)  
print(f"{celsius_temp}°C is equal to {fahrenheit_temp:.2f}°F")
```

25°C is equal to 77.00°F

P give me code for a python function that converts celsius to fahrenheit

I can provide you with that. Here is the function to convert Celsius to Fahrenheit:

I hope that was helpful! Do you want to try converting Fahrenheit to Celsius, or perhaps explore other types of unit conversions?

👍 🗨

PROMPT 3:

```
def fahrenheit_to_celsius(fahrenheit):
    """converts Fahrenheit to Celsius."""
    celsius = (fahrenheit - 32) * 5/9
    return celsius

def celsius_to_fahrenheit(celsius):
    """converts Celsius to Fahrenheit."""
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

while True:
    try:
        temperature = float(input("Enter the temperature value: "))
        unit = input("Enter the unit of temperature (C for Celsius, F for Fahrenheit): ").upper()

        if unit == 'C':
            fahrenheit_result = celsius_to_fahrenheit(temperature)
            print(f"{temperature}°C is equal to {fahrenheit_result:.2f}°F")
            break
        elif unit == 'F':
            celsius_result = fahrenheit_to_celsius(temperature)
            print(f"{temperature}°F is equal to {celsius_result:.2f}°C")
            break
        else:
            print("Invalid unit. Please enter 'C' or 'F'.")
    except ValueError:
        print("Invalid input. Please enter a numerical value for temperature.")

Enter the temperature value: 25
Enter the unit of temperature (C for Celsius, F for Fahrenheit): C
25.0°C is equal to 77.00°F
```

Gemini X

give me code that converts temperature from celsius to fahrenheit and fahrenheit to celsius, take input from user and return the converted value

Certainly! Here is the code that takes temperature input from the user and converts it between Celsius and Fahrenheit.

Great! The temperature conversion code is ready. Would you like to explore other unit conversions, like weight or distance? Or perhaps you'd like to add more error handling to the current code?