

# **ASSIGNMENT – 22.1**

**NAME : BHAWWANA SHRE**

**HALL TICKET NO : 2403A52311**

**BATCH NUMBER : 01**

**COURSE CODE : 24CS002PC215**

**PROGRAM NAME : B.TECH**

**YEAR/SEM : 2ND AND 3<sup>RD</sup>**

## ***Task 1 – Bias Detection in AI Suggestions***

- *Provide the same coding task to an AI tool multiple times.*
- *Check if AI outputs different results depending on prompt phrasing.*
- *Discuss:*

*Are there hidden assumptions or biases in the responses?*

*How could biased AI code impact fairness or accessibility?*

**PROMT :** Write a single Python solution for the task: ‘Create a function that sorts a list of numbers.’ After giving the code, also explain:

- *how changing the phrasing of this same task might lead the AI to generate different styles of code,*
- *whether any hidden assumptions or biases may appear in such responses,*
- *and how biased AI-generated code could impact fairness, accessibility, or user experience.”*

The screenshot shows a Google Colab interface with two cells. The first cell, titled 'Reasoning', contains text explaining the task: generating Python code for a sorting function. It includes a code block for 'sort\_numbers' and an example usage. The second cell, titled 'Subtask:', contains a list item under 'Explain Phrasing Impact on AI Code Generation'.

```
Reasoning: The subtask requires generating Python code for a sorting function based on the provided instructions. I will use a code_block to define the function, implement the sorting logic, and include an example of its usage.

Gemini
def sort_numbers(numbers_list):
    """
    Sorts a list of numbers in ascending order and returns a new sorted list.
    The original list remains unchanged.
    """
    return sorted(numbers_list)

# Example usage:
sample_list = [5, 2, 8, 1, 9, 4]
print(f"Original list: {sample_list}")
sorted_list = sort_numbers(sample_list)
print(f"Sorted list: {sorted_list}")
print(f"Original list after sorting: {sample_list} (should be unchanged)")

...
*** Original list: [5, 2, 8, 1, 9, 4]
sorted list: [1, 2, 4, 5, 8, 9]
Original list after sorting: [5, 2, 8, 1, 9, 4] (should be unchanged)
```

- ✓ Explain Phrasing Impact on AI Code Generation

Subtask:

### OBSERVATION :

The selected cell is a 'Reasoning' text cell. It explains that the purpose of the next step was to generate Python code for a sorting function, define the function, implement the sorting logic, and include an example of its usage. This reasoning served as the prelude to the Python code cell that followed, which contains the actual sorting function implementation.

### Task 2 – Privacy and Sensitive Data Handling

- Ask AI to write a script that processes user personal data (e.g., names, phone numbers).

Discuss:

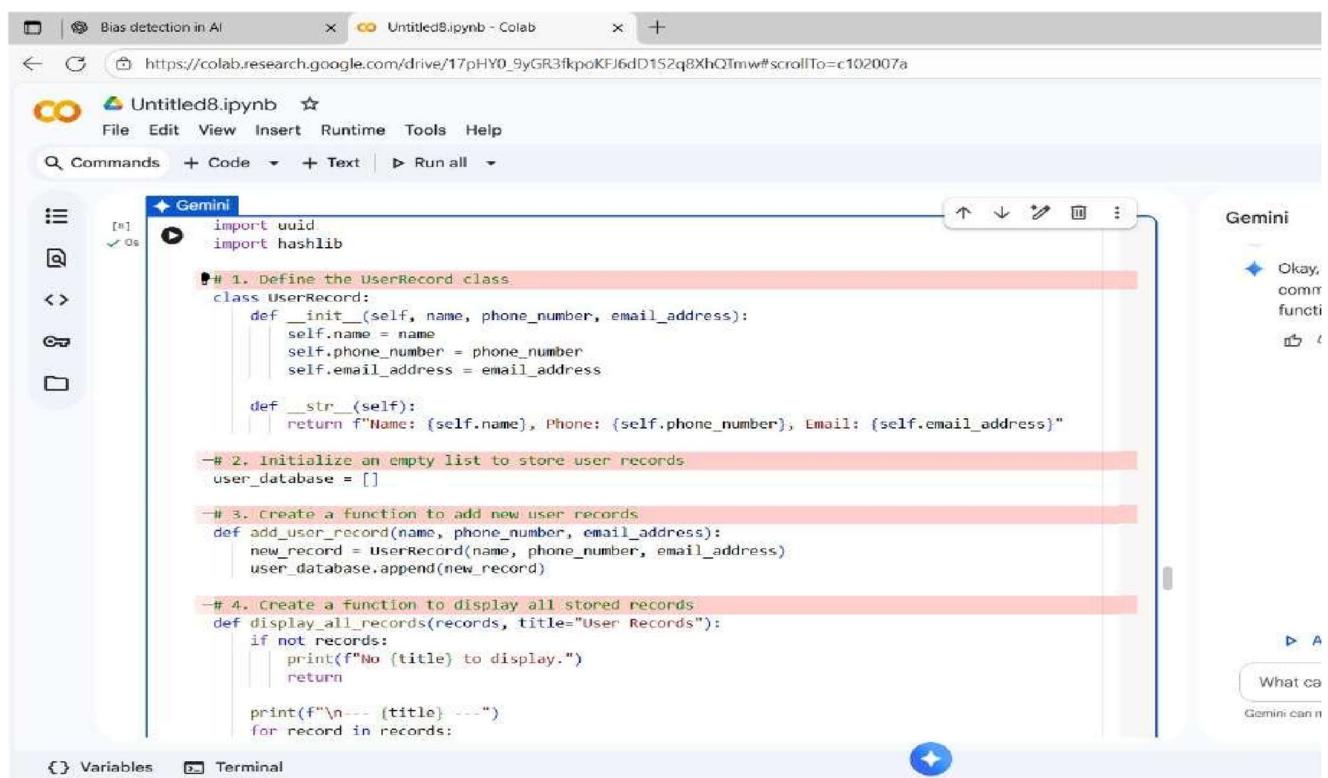
- o How should sensitive data be anonymized?
- o What ethical/legal issues could arise if AI-generated code mishandles it? o What role does developer responsibility play here?

**PROMT :** Write a Python script that processes user personal data such as names, phone numbers, and email addresses. The script should read a list of user records, store them, and print them in a structured format.

After generating the code, also explain:

- how this sensitive data should be anonymized or masked (with examples),
- what ethical or legal issues could occur if the code mishandles personal data (GDPR, privacy risks, misuse),
- and what responsibilities developers have when building systems involving personal information.”

### CODE :



The screenshot shows a Google Colab notebook titled "Untitled8.ipynb - Colab". The code cell contains the following Python script:

```

import uuid
import hashlib

# 1. Define the UserRecord class
class UserRecord:
    def __init__(self, name, phone_number, email_address):
        self.name = name
        self.phone_number = phone_number
        self.email_address = email_address

    def __str__(self):
        return f"Name: {self.name}, Phone: {self.phone_number}, Email: {self.email_address}"

# 2. Initialize an empty list to store user records
user_database = []

# 3. Create a function to add new user records
def add_user_record(name, phone_number, email_address):
    new_record = UserRecord(name, phone_number, email_address)
    user_database.append(new_record)

# 4. Create a function to display all stored records
def display_all_records(records, title="User Records"):
    if not records:
        print(f"No {title} to display.")
        return

    print(f"\n--- {title} ---")
    for record in records:

```

The code is annotated with comments explaining each step: defining the UserRecord class, initializing an empty list for user records, creating a function to add new records, and creating a function to display all stored records.

Bias detection in AI

Untitled8.ipynb - Colab

https://colab.research.google.com/drive/17pHY0\_9yGR3fkpoKFJ6dD1S2q8XhQTmw#scrollTo=c102007a

Untitled8.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▶ Run all

```
print("\n--- {title} ---")
for record in records:
    print(record)
print("-----")

# Populate user_database with sample records
add_user_record("Alice Smith", "111-222-3333", "alice.s@example.com")
add_user_record("Bob Johnson", "444-555-6666", "bob.j@example.com")
add_user_record("Charlie Brown", "777-888-9999", "charlie.b@example.com")
add_user_record("Diana Prince", "123-456-7890", "diana.p@example.com")

display_all_records(user_database, "Original User Records")

# 5. Define the pseudonymize_records function
def pseudonymize_records(records):
    pseudonymized_records = []
    name_to_pseudonym = {}
    for record in records:
        # Create a copy to avoid modifying the original record directly in the loop
        new_record = UserRecord(record.name, record.phone_number, record.email_address)
        if new_record.name not in name_to_pseudonym:
            name_to_pseudonym[new_record.name] = f"user_{uuid.uuid4().hex[:8]}"
        new_record.name = name_to_pseudonym[new_record.name]
        pseudonymized_records.append(new_record)
    return pseudonymized_records

# 6. Define the hash_records function
def hash_records(records):
    hashed_records = []
    for record in records:
```

Bias detection in AI

Untitled8.ipynb - Colab

https://colab.research.google.com/drive/17pHY0\_9yGR3fkpoKFJ6dD1S2q8XhQTmw#scrollTo=c102007a

Untitled8.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▶ Run all

```
for record in records:
    new_record = UserRecord(record.name, record.phone_number, record.email_address)
    hashed_email = hashlib.sha256(new_record.email_address.encode()).hexdigest()
    new_record.email_address = hashed_email
    hashed_records.append(new_record)
return hashed_records

# 7. Define the generalize_records function
def generalize_records(records):
    generalized_records = []
    for record in records:
        new_record = UserRecord(record.name, record.phone_number, record.email_address)

        if new_record.phone_number and len(new_record.phone_number) >= 4:
            new_record.phone_number = new_record.phone_number[:-4] + 'XXXX'
        else:
            new_record.phone_number = 'XXXX-XXXX'

        if new_record.email_address and '@' in new_record.email_address:
            username, domain = new_record.email_address.split('@')
            if len(username) > 3:
                new_record.email_address = username[:3] + '***@' + domain
            else:
                new_record.email_address = '***@' + domain
        else:
            new_record.email_address = '***@example.com'

        generalized_records.append(new_record)
    return generalized_records
```

Gemini

Okay, I've made the code skip comments, empty lines, and ; functionality remains the same.

Accept & R Screenshot Automatic

What can I help you with?

Gemini can make mistakes

The screenshot shows a Google Colab interface with two panes. The left pane displays Python code demonstrating four data anonymization techniques: Pseudonymization, Hashing, Generalization, and Redaction. The right pane shows a Gemini AI interface with a message from the AI and a screenshot of the AI's interface.

```

File Edit View Insert Runtime Tools Help
Commands + Code + Text | Run all
... original user records ...
Name: Alice Smith, Phone: 111-222-3333, Email: alice@example.com
Name: Bob Johnson, Phone: 444-555-6666, Email: bob@example.com
Name: Charlie Brown, Phone: 777-888-9999, Email: charlie@example.com
Name: Diana Prince, Phone: 123-456-7890, Email: diana.p@example.com
...
... Pseudonymized User Records ...
Name: User_f206e2fb, Phone: 111-222-3333, Email: alice.s@example.com
Name: User_f47fcfb, Phone: 444-555-6666, Email: bob.j@example.com
Name: User_c11685ce, Phone: 777-888-9999, Email: charlie.b@example.com
Name: User_098cfcf1, Phone: 123-456-7890, Email: diana.p@example.com
...
... Hashed User Records (Email) ...
Name: Alice Smith, Phone: 111-222-3333, Email: 0b11f4dad50cfb477dc2291479621492fe003786660022dc9ad8831c
Name: Bob Johnson, Phone: 444-555-6666, Email: a36e50d652da695ba385708be795576hd9eb8211095c5hec2a2987b
Name: Charlie Brown, Phone: 777-888-9999, Email: 0c8f6dbba950c8c54686170b688453629696727af2285ecb2c51f
Name: Diana Prince, Phone: 123-456-7890, Email: a3344a2ecf387cd73b3b280bd8d81e353b40082601fb56be3fafd
...
... Generalized User Records (Phone & Email) ...
Name: Alice Smith, Phone: 111-222-XXXX, Email: ali***@example.com
Name: Bob Johnson, Phone: 444-555-XXXX, Email: bob***@example.com
Name: Charlie Brown, Phone: 777-888-XXXX, Email: cha***@example.com
Name: Diana Prince, Phone: 123-456-XXXX, Email: dia***@example.com
...
... Redacted User Records (Email & Phone) ...
Name: Alice Smith, Phone: [REDACTED], Email: [REDACTED]
Name: Bob Johnson, Phone: [REDACTED], Email: [REDACTED]
Name: Charlie Brown, Phone: [REDACTED], Email: [REDACTED]

```

Gemini

Okay, I've made the code shorter by removing non-essential comments, empty lines, and some verbose output. The functionality remains the same, but it's more concise.

Accept & R Snipping Tool

What can I help you with? Gemini can make mistakes.

Screenshot copied to clipboard Automatically saved to screenshots folder.

Mark up and share

**OBSERVATION :** Certainly! This script defines a `UserRecord` class to manage user data (`name, phone, email`). It then demonstrates four data anonymization techniques:

*Pseudonymization: Replaces names with unique, artificial IDs (e.g., "User\_abc123").*

*Hashing: Transforms email addresses into irreversible hash values.*

*Generalization: Masks parts of phone numbers (e.g., "XXXX") and email usernames (e.g., "ali\*").*

*Redaction: Completely replaces sensitive fields (phone, email) with [REDACTED].*

*Each technique is applied to a copy of the original data, and the results are printed to show how sensitive information is protected.*

### Task 3 – AI-Generated Security Risks

- Use AI to generate login/authentication code.
- Identify possible security flaws (e.g., weak password storage, lack of encryption).
- Students must document risks and propose ethical guidelines for safe AI code usage.

**PROMT :** “Generate a simple Python login/authentication script that handles usernames and passwords. After giving the code, identify all possible security flaws in the AI-generated code (such as plain-text passwords, lack of hashing, weak validation, no encryption, or insecure storage). Then explain the risks these flaws create and propose ethical guidelines for safely using AI-generated code in authentication systems.” CODE :

Bias detection in AI

Untitled8.ipynb - Colab

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Gemini

```
[14]: registered_users = {}

def register_user(username, password):
    """
    Registers a new user by adding their username and password to the registered_users dictionary.
    """
    if username in registered_users:
        print(f"Username '{username}' already exists. Please choose a different one.")
    else:
        registered_users[username] = password
        print(f"User '{username}' registered successfully.")

def login_user(username, password):
    """
    Attempts to log in a user. Returns True for successful login, False otherwise.
    """
    if username in registered_users:
        if registered_users[username] == password:
            return True
        else:
            print("Incorrect password.")
            return False
    else:
        print("Username not found.")
        return False

# 4. Add sample users
print("--- Registering Users ---")
```

Generating...

Prepared

- Generate Python Login/Authentication Script
- Identify Security Flaws in the AI-Generated Code
- Explain Risks Created by Security Flaws
- Propose Ethical Guidelines for AI-Generated Authentication Code
- Final Task

What can I help you build?

RAM Disk

1:51PM Python 3

Bias detection in AI

Untitled8.ipynb - Colab

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Gemini

```
[14]: # 4. Add sample users
print("--- Registering Users ---")
register_user("admin", "admin123")
register_user("user1", "password123")
register_user("guest", "guestpass")
print("-----")

# 5. Demonstrate successful login
print("\n--- Demonstrating Successful Login ---")
if login_user("admin", "admin123"):
    print("Login successful for admin!")
else:
    print("Login failed for admin.")
print("-----")

# 6. Demonstrate failed login attempts
print("\n--- Demonstrating Failed Login Attempts ---")
# Incorrect password
if login_user("user1", "wrongpass"):
    print("Login successful for user1 (should fail).")
else:
    print("Login failed for user1 (as expected due to wrong password).")

# Non-existent username
if login_user("nonexistent", "anypass"):
    print("Login successful for nonexistent user (should fail).")
else:
    print("Login failed for nonexistent user (as expected).")
print("-----")
```

Generating...

Prepared

- Generate Python Login/Authentication Script
- Identify Security Flaws in the AI-Generated Code
- Explain Risks Created by Security Flaws
- Propose Ethical Guidelines for AI-Generated Authentication Code
- Final Task

What can I help you build?

RAM Disk

1:51PM Python 3

```

print("Login failed for nonexistent user (as expected).")
# Incorrect password for existing user
if login_user("guest", "wrongguesspass"):
    print("Login successful for guest (should fail).")
else:
    print("Login failed for guest (as expected).")
print("-----")

--- Registering Users ---
User 'admin' registered successfully.
User 'user1' registered successfully.
User 'guest' registered successfully.

--- Demonstrating Successful Login ---
Login successful for admin!
----- 

--- Demonstrating Failed Login Attempts ---
Incorrect password.
Login failed for user1 (as expected due to wrong password).
Username not found.
Login failed for nonexistent user (as expected).
Incorrect password.
Login failed for guest (as expected).
----- 

```

**Gemini**

- Propose Ethical Guidelines for AI-Generated Authentication Code: Outline ethical guidelines and best practices for responsibly using AI-generated code, particularly in critical systems like authentication, emphasizing security, privacy, and user trust.
- Final Task: Summarize the generated script, the identified security flaws, the associated risks, and the proposed ethical guidelines for using AI-generated code in authentication systems.

Generating... Prepared

Preparing Generating

Generate P Identify S

Identify S Explain R

Explain R Propose E

Propose E Final T

Final T Snipping Tool

What can I help you with? Screenshot copied to clipboard

Gemini can make mistakes. Automatically saved to screenshots folder.

Mark up and share

**OBSERVATION :** A key observation is that while the AI-generated login script fulfills the basic functional requirement, it severely lacks fundamental security measures essential for any real-world application, primarily by storing passwords in plain text. This highlights the critical need for developers to scrutinize AI-generated code for security vulnerabilities, especially in sensitive areas like authentication.

#### Task 4 – Accountability in Human–AI Collaboration

- Scenario: AI writes code for a hospital's patient monitoring system.
- If the code fails and causes harm, who is responsible?
  - o The AI tool? o The human developer? o The organization using it?
- Students must debate and submit a reflection report.

**PROMT :** “Write a simple Python code snippet for a basic patient-monitoring system that checks vital signs (e.g., heart rate and temperature) and prints alerts if values exceed safe limits.

After generating the code, explain who would be responsible if such AI-assisted code fails in a real hospital environment and causes harm — the AI tool, the human developer, or the organization using it. Provide a detailed reflection discussing accountability, shared responsibility, and ethical considerations in human–AI collaboration.”

### CODE :

```

[1]: import random
import time

# 1. Define acceptable ranges for vital signs
# Heart Rate (BPM)
HR_MIN_SAFE = 60
HR_MAX_SAFE = 100

# Body Temperature (Celsius)
TEMP_MIN_SAFE = 36.0
TEMP_MAX_SAFE = 37.5

print("Patient Monitoring System Initialized")
print(f"Safe Heart Rate Range: [{HR_MIN_SAFE}-{HR_MAX_SAFE}] bpm")
print(f"Safe Temperature Range: [{TEMP_MIN_SAFE}-{TEMP_MAX_SAFE}]°C\n")

# 2. Function to simulate generating random vital sign readings
def get_vital_signs():
    # Simulate heart rate (some within, some outside safe range)
    heart_rate = random.randint(50, 120)
    # Simulate temperature (some within, some outside safe range)
    temperature = round(random.uniform(35.0, 39.0), 1)
    return heart_rate, temperature

# 3. Function to check vital signs against predefined safe limits
def check_vitals(heart_rate, temperature):
    alert_triggered = False

```

```

[1]: # 6. Simulate the monitoring process
monitoring_cycles = 10
print("Starting monitoring cycles...")
for i in range(monitoring_cycles):
    print(f"\nCycle {i+1}/{monitoring_cycles}:")
    hr, temp = get_vital_signs()
    check_vitals(hr, temp)
    time.sleep(1) # Small delay for realism

print("Monitoring complete.")

*** Patient Monitoring System Initialized
Safe Heart Rate Range: 60-100 bpm
Safe Temperature Range: 36.0-37.5°C

Starting 10 monitoring cycles...

Cycle 1/10:
Checking Vital Signs: HR=72 bpm, Temp=38.5°C
ALERT: High Temperature - 38.5°C!

Cycle 2/10:
Checking Vital Signs: HR=92 bpm, Temp=37.1°C
Vitals are within normal limits.

Cycle 3/10:
Checking Vital Signs: HR=82 bpm, Temp=36.5°C
Vitals are within normal limits.

```

The image shows two side-by-side screenshots of Google Colab notebooks. Both notebooks are titled 'Untitled8.ipynb' and are running on Python 3 at 1:56 PM.

**Notebook 1 (Left):**

```

ALERT: High Temperature - 38.9°C!
-----
Cycle 5/10:
Checking Vital signs: HR=63 bpm, Temp=36.4°C
Vitals are within normal limits.

-----
Cycle 6/10:
Checking Vital signs: HR=116 bpm, Temp=37.5°C
ALERT: High Heart Rate - 116 bpm!

-----
Cycle 7/10:
Checking Vital signs: HR=88 bpm, Temp=36.6°C
Vitals are within normal limits.

-----
Cycle 8/10:
Checking Vital signs: HR=70 bpm, Temp=35.8°C
ALERT: Low Temperature - 35.8°C!

-----
Cycle 9/10:
Checking Vital signs: HR=50 bpm, Temp=39.0°C
ALERT: Low Heart Rate - 50 bpm!
ALERT: High Temperature - 39.0°C!

-----
Cycle 10/10:
Checking Vital signs: HR=117 bpm, Temp=38.9°C
ALERT: High Heart Rate - 117 bpm!
ALERT: High Temperature - 38.9°C!

```

**Notebook 2 (Right):**

```

# 3. Function to check vital signs against predefined safe limits
def check_vitals(heart_rate, temperature):
    alert_triggered = False

    print("Checking Vital signs: HR=(heart_rate) bpm, Temp=(temperature)°C")

    if heart_rate < HR_MIN_SAFE:
        print("ALERT: Low Heart Rate - (heart_rate) bpm!")
        alert_triggered = True
    elif heart_rate > HR_MAX_SAFE:
        print("ALERT: High Heart Rate - (heart_rate) bpm!")
        alert_triggered = True

    if temperature < TEMP_MIN_SAFE:
        print("ALERT: Low Temperature - (temperature)°C!")
        alert_triggered = True
    elif temperature > TEMP_MAX_SAFE:
        print("ALERT: High Temperature - (temperature)°C!")
        alert_triggered = True

    if not alert_triggered:
        print("Vitals are within normal limits.")

# 6. Simulate the monitoring process
monitoring_cycles = 10
print("\nStarting (%d monitoring cycles...)" % monitoring_cycles)
for i in range(monitoring_cycles):
    print("%d Cycle (%d)/%d" % (i+1, i+1, monitoring_cycles))

```

**Gemini Sidebar (Right):**

System that checks vital signs (e.g., heart rate and temperature) and prints alerts if values exceed safe limits.

- Explain Accountability for AI-Assisted Code Failure: Explain who would be responsible if such AI-assisted code fails in a real hospital environment and causes harm — the AI tool, the human developer, or the organization using it. Provide a detailed reflection discussing accountability, shared responsibility, and ethical considerations in human-AI collaboration.
- Final Task: Summarize the generated patient monitoring code and the detailed discussion on accountability, shared responsibility, and ethical considerations in human-AI collaboration for critical systems.

Generating... Preparing

Generate Python code for the patient monitoring system Explain the accountability discussion for AI-assisted code failure Reflect on the security flaws of the authentication script

What can I help you build?

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

**OBSERVATION:** A key observation is that this script provides a simplified, simulated patient monitoring system. While it effectively demonstrates basic vital sign checking and alerting logic, a real-world medical application would require significantly more robust data handling, error checking, and advanced features for patient safety.

## Task 5 – Limitations of AI in Critical Systems

- Ask AI to generate code for a life-critical application (e.g., railway signaling, medical diagnosis).
- Discuss:
  - Should AI code be used directly in such systems? o What level of human oversight is mandatory?
  - How do we balance efficiency vs safety?

**PROMT :** Generate a simple Python script for a life-critical system, such as a basic railway-signal safety checker that monitors track status and prints warnings if a collision risk is detected.

After the code, discuss:

- whether AI-generated code should ever be used directly in life-critical systems,
- what level of human oversight is mandatory,
- and how to balance efficiency, automation, and safety when using AI in high-risk environments.”

### • CODE :

The screenshot shows a Google Colab notebook titled "Untitled8.ipynb". The code cell contains the following Python script:

```
import time
import random

# 1. Define a list of simulated 'tracks' or 'track segments'
tracks = [str(i) for i in range(10)] # Tracks named "0" to "9"
print(f"Defined Tracks: {tracks}\n")

# 2. Create a class to represent 'trains'
class Train:
    def __init__(self, train_id, current_track_position, direction=1): # direction: 1 for forward, -1 for backward
        self.train_id = train_id
        self.current_track_position = current_track_position
        self.direction = direction

    def __str__(self):
        return f"Train ID: {self.train_id}, Position: {self.current_track_position}, Direction: {'Forward' if self.direction == 1 else 'Backward'}
```

The right side of the screen displays a sidebar titled "Gemini" with the following sections:

- Discuss Balancing Efficiency, Automation, and Safety:  
Detail how to balance efficiency, automation, and safety when deploying AI in high-risk environments, such as railway signaling, emphasising strategies to mitigate risks without sacrificing necessary advancements.
- Final Task: Summarize the generated railway-signal safety checker code and the key points from the discussions on using AI-generated code in life-critical systems, human oversight, and the balance between efficiency, automation, and safety.

Below the sidebar, there is a "What can I help you build?" input field and a note: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

Bias detection in AI

Untitled8.ipynb

```

# 4. Define a function to simulate train movement
def simulate_train_movement(trains, tracks):
    print("---- Simulating Train Movement ----")
    track_length = len(tracks)
    for train in trains:
        current_index = tracks.index(train.current_track_position)
        next_index = current_index + train.direction

        # Ensure trains stay within track segments (wrap around for simplicity)
        if next_index >= track_length:
            next_index = 0 # Wrap to the beginning
        elif next_index < 0:
            next_index = track_length - 1 # Wrap to the end

        train.current_track_position = tracks[next_index]
        print(f"Train {train.train_id} moved to track {train.current_track_position}")

# 5. Create a function to check for collision risk
def check_collision_risk(trains):
    print("---- Checking for collision Risk ----")
    positions = {}
    for train in trains:
        if train.current_track_position not in positions:
            positions[train.current_track_position] = []
        positions[train.current_track_position].append(train.train_id)

    collision_detected = False
    for track_segment, trains_on_segment in positions.items():
        if len(trains_on_segment) > 1:
            print(f"!!! COLLISION RISK DETECTED on Track {track_segment} involving trains: {', '.join(trains_on_segment)}")
            collision_detected = True

    if not collision_detected:
        print("No collision risk detected.")
    print("-----")

# 7. simulate the system over several time steps
simulation_steps = 5
print(f"\nStarting {simulation_steps} simulation steps...")
for step in range(simulation_steps):
    print(f"\n==== Simulation Step {step + 1} ====")
    simulate_train_movement(trains, tracks)
    check_collision_risk(trains)

    # Introduce a small delay for better readability
    time.sleep(1)

print("\nSimulation Complete.")
Starting 5 simulation steps...
==== Simulation Step 1 ====
--- Simulating Train Movement ---
Train T001 moved to track 1
Train T002 moved to track 3
Train T003 moved to track 7

```

Variables Terminal

Gemini

- Discuss Balancing Efficiency, Automation, and Safety:** Detail how to balance efficiency, automation, and safety when deploying AI in high-risk environments, such as railway signaling, emphasizing strategies to mitigate risks without sacrificing necessary advancements.
- Final Task:** Summarize the generated railway-signal safety checker code and the key points from the discussions on using AI-generated code in life-critical systems, human oversight, and the balance between efficiency, automation, and safety.

Generalizing... Preparing Generate Python Railway-Signal Safety Checker Code Discuss Use of AI-Generated Code in Life-Critical Systems Discuss Mandatory Human Oversight Discuss Balancing Efficiency, Automation, and Safety Final Task Snipping Tool

What can I help you? Gemini can make mistakes

Screenshot copied to clipboard Automatically saved to screenshots folder.

Mark up and share

Bias detection in AI

Untitled8.ipynb

```

collision_detected = False
for track_segment, trains_on_segment in positions.items():
    if len(trains_on_segment) > 1:
        print(f"!!! COLLISION RISK DETECTED on Track {track_segment} involving trains: {', '.join(trains_on_segment)}")
        collision_detected = True

if not collision_detected:
    print("No collision risk detected.")
print("-----")

# 7. simulate the system over several time steps
simulation_steps = 5
print(f"\nStarting {simulation_steps} simulation steps...")
for step in range(simulation_steps):
    print(f"\n==== Simulation Step {step + 1} ====")
    simulate_train_movement(trains, tracks)
    check_collision_risk(trains)

    # Introduce a small delay for better readability
    time.sleep(1)

print("\nSimulation Complete.")
Starting 5 simulation steps...
==== Simulation Step 1 ====
--- Simulating Train Movement ---
Train T001 moved to track 1
Train T002 moved to track 3
Train T003 moved to track 7

```

Variables Terminal

Gemini

- Discuss Balancing Efficiency, Automation, and Safety:** Detail how to balance efficiency, automation, and safety when deploying AI in high-risk environments, such as railway signaling, emphasizing strategies to mitigate risks without sacrificing necessary advancements.
- Final Task:** Summarize the generated railway-signal safety checker code and the key points from the discussions on using AI-generated code in life-critical systems, human oversight, and the balance between efficiency, automation, and safety.

Generalizing... Preparing Generate Python Railway-Signal Safety Checker Code Discuss Use of AI-Generated Code in Life-Critical Systems Discuss Mandatory Human Oversight Discuss Balancing Efficiency, Automation, and Safety Final Task

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

2:03PM Python 3

The screenshot shows a Google Colab notebook titled "Untitled8.ipynb". The code in the notebook simulates train movements and checks for collisions. The output shows three steps of simulation:

```

==== simulation Step 3 ====
...
--- Simulating Train Movement ---
Train T001 moved to track 3
Train T002 moved to track 5
Train T003 moved to track 5

--- Checking for Collision Risk ---
!!! COLLISION RISK DETECTED on Track 5 involving trains: T002, T003 !!!
-----

==== Simulation Step 4 ====
...
--- Simulating Train Movement ---
Train T001 moved to track 4
Train T002 moved to track 6
Train T003 moved to track 4

--- Checking for Collision Risk ---
!!! COLLISION RISK DETECTED on Track 4 involving trains: T001, T003 !!!
-----

==== Simulation Step 5 ====
...
--- Simulating Train Movement ---
Train T001 moved to track 5
Train T002 moved to track 7
Train T003 moved to track 3

--- Checking for Collision Risk ---
No collision risk detected.

```

To the right, a Gemini AI sidebar provides a summary of the task and a list of generated points:

- Discuss Balancing Efficiency, Automation, and Safety: Detail how to balance efficiency, automation, and safety when deploying AI in high-risk environments, such as railway signaling, emphasizing strategies to mitigate risks without sacrificing necessary advancements.
- Final Task: Summarize the generated railway signal safety checker code and the key points from the discussions on using AI-generated code in life-critical systems, human oversight, and the balance between efficiency, automation, and safety.

Below the sidebar, a message says "Generating..." and "Preparing". A list of generated points includes:
 

- Generate Python Railway-Signal Safety Checker Code
- Discuss Use of AI-Generated Code in Life-Critical Systems
- Discuss Mandatory Human Oversight
- Discuss Balancing Efficiency, Automation, and Safety
- Final Task

## OBSERVATION :

*This script is a simplified simulation of a railway safety system. It effectively demonstrates the core idea of tracking multiple trains and detecting potential collision risks by checking if more than one train occupies the same track segment. While basic, it clearly illustrates the principle of collision detection.*

## Task 6 – Ethical Use of AI-Generated Code

- Students analyze real-world cases (e.g., plagiarism, copyright violation, misuse of open-source code).
- Identify best practices for ethically using AI code:
  - Attribution
  - Verification
  - Responsible deployment

**PROMT :** “Write a simple Python program (any small utility, such as a text formatter or calculator). After giving the code, analyze ethical issues related to AI-generated code using real-world concerns such as plagiarism, copyright violations, and misuse of opensource code. Then explain best practices for ethically using AI-generated code, including proper attribution, verification, and responsible deployment.”

## CODE :

Bias detection in AI

Untitled8.ipynb · Colab

Untitled8.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text | Run all

```
[17] ✓ 5s
import time
import random

# 1. Define a list of simulated 'tracks' or 'track segments'
tracks = [str(i) for i in range(10)] # Tracks named "0" to "9"
print(f"Defined Tracks: {tracks}\n")

# 2. Create a class to represent 'trains'
class Train:
    def __init__(self, train_id, current_track_position, direction=1):
        self.train_id = train_id
        self.current_track_position = current_track_position
        self.direction = direction

    def __str__(self):
        return f"Train ID: {self.train_id}, Position: {self.current_track_position}, Direction: {'Forward' if self.direction == 1 else 'Backward'}
```

# 3. Initialize a few sample trains

```
trains = [
    Train("T001", "0", 1), # Train 1 starts at track 0, moving forward
    Train("T002", "2", 1), # Train 2 starts at track 2, moving forward
    Train("T003", "8", -1) # Train 3 starts at track 8, moving backward
]

print("Initial train Positions:")
for train in trains:
    print(train)
    print("\n")
```

Gemini

such as plagiarism, copyright violations, and misuse of open-source code. Then explain best practices for ethically using AI-generated code, including proper attribution, verification, and responsible deployment.

Certainly! I'll generate a simple Python utility program for you, and then we'll dive into the critical ethical issues surrounding AI-generated code, including plagiarism, copyright, and misuse of open-source resources, along with best practices for its ethical use. Here is the plan:

- Generate Simple Python Utility Program: Create a Python code snippet for a small utility, such as a basic calculator or a text formatter.
- Analyze Ethical Issues of AI-Generated Code: Discuss ethical issues related to AI-generated code, including plagiarism, copyright violations, and misuse of open-source

Generate a simple Python calculator program

Analyze ethical issues of AI-generated code

Explain best practices for ethically using AI-generated code

What can I help you build?

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

✓ 2:10PM Python 3

Bias detection in AI

Untitled8.ipynb · Colab

Untitled8.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text | Run all

```
[17] ✓ 5s
# 4. Define a function to simulate train movement
def simulate_train_movement(trains, tracks):
    print("\n--- Simulating Train Movement ---")
    track_length = len(tracks)
    for train in trains:
        current_index = tracks.index(train.current_track_position)
        next_index = current_index + train.direction

        # Ensure trains stay within track segments (wrap around for simplicity)
        if next_index > track_length:
            next_index = 0 # Wrap to the beginning
        elif next_index < 0:
            next_index = track_length - 1 # Wrap to the end

        train.current_track_position = tracks[next_index]
        print(f"Train {train.train_id} moved to track {train.current_track_position}")

# 5. Create a function to check for collision risk
def check_collision_risk(trains):
    print("\n--- Checking for Collision Risk ---")
    positions = {}
    for train in trains:
        if train.current_track_position not in positions:
            positions[train.current_track_position] = []
        positions[train.current_track_position].append(train.train_id)

    collision_detected = False
    for track_segment, trains_on_segment in positions.items():
        if len(trains_on_segment) > 1:
            print(f"!!! COLLISION RISK DETECTED on Track {track_segment} involving trains: {trains_on_segment}")
            collision_detected = True

    return collision_detected
```

Gemini

such as plagiarism, copyright violations, and misuse of open-source code. Then explain best practices for ethically using AI-generated code, including proper attribution, verification, and responsible deployment.

Certainly! I'll generate a simple Python utility program for you, and then we'll dive into the critical ethical issues surrounding AI-generated code, including plagiarism, copyright, and misuse of open-source resources, along with best practices for its ethical use. Here is the plan:

- Generate Simple Python Utility Program: Create a Python code snippet for a small utility, such as a basic calculator or a text formatter.
- Analyze Ethical Issues of AI-Generated Code: Discuss ethical issues related to AI-generated code, including plagiarism, copyright violations, and misuse of open-source

Generate a simple Python calculator program

Analyze ethical issues of AI-generated code

Explain best practices for ethically using AI-generated code

What can I help you build?

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

✓ 2:10PM Python 3

Bias detection in AI

Untitled8.ipynb · Colab

File Edit View Insert Runtime Tools Help

Commands + Code + Text | Run all

```
if not collision_detected:  
    print("No collision risk detected.")  
    print("-----")  
  
# 7. Simulate the system over several time steps  
simulation_steps = 5  
print(f"\nStarting {simulation_steps} simulation steps...")  
for step in range(simulation_steps):  
    print(f"\n--- Simulation Step {step + 1} ---")  
    simulate_train_movement(trains, tracks)  
    check_collision_risk(trains)  
  
    # Introduce a small delay for better readability  
    time.sleep(1)  
  
print("\nSimulation Complete.")  
  
Starting 5 simulation steps...  
--- Simulation Step 1 ---  
--- SIMULATING TRAIN MOVEMENT ---  
Train T001 moved to track 1  
Train T002 moved to track 3  
Train T003 moved to track 7  
--- Checking for collision Risk ---  
No collision risk detected.  
-----
```

Gemini

such as plagiarism, copyright violations, and misuse of open-source code. Then explain best practices for ethically using AI-generated code, including proper attribution, verification, and responsible deployment.

Certainly! I'll generate a simple Python utility program for you, and then we'll dive into the critical ethical issues surrounding AI-generated code, including plagiarism, copyright, and misuse of open-source resources, along with best practices for its ethical use. Here is the plan:

- Generate Simple Python Utility Program: Create a Python code snippet for a small utility, such as a basic calculator or a text formatter.
- Analyze Ethical Issues of AI-Generated Code: Discuss ethical issues related to AI-generated code, including plagiarism, copyright violations, and misuse of open-source

Generate a simple Python calculator program  
Analyze ethical issues of AI-generated code  
Explain best practices for ethically using AI-generated code

What can I help you build? + >

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

Variables Terminal

2:10 PM Python 3

Bias detection in AI

Untitled8.ipynb · Colab

File Edit View Insert Runtime Tools Help

Commands + Code + Text | Run all

```
--- Simulation Step 3 ---  
... --- simulating Train Movement ---  
Train T001 moved to track 3  
Train T002 moved to track 5  
Train T003 moved to track 5  
  
--- Checking for Collision Risk ---  
!!! COLLISION RISK DETECTED on Track 5 involving trains: T002, T003 !!!  
-----  
  
--- Simulation Step 4 ---  
--- SIMULATING TRAIN MOVEMENT ---  
Train T001 moved to track 4  
Train T002 moved to track 6  
Train T003 moved to track 4  
  
--- Checking for Collision Risk ---  
!!! COLLISION RISK DETECTED on Track 4 involving trains: T001, T003 !!!  
-----  
  
--- Simulation Step 5 ---  
--- SIMULATING TRAIN MOVEMENT ---  
Train T001 moved to track 5  
Train T002 moved to track 7  
Train T003 moved to track 3  
  
--- Checking for Collision Risk ---  
No collision risk detected.
```

Gemini

such as plagiarism, copyright violations, and misuse of open-source code. Then explain best practices for ethically using AI-generated code, including proper attribution, verification, and responsible deployment.

Certainly! I'll generate a simple Python utility program for you, and then we'll dive into the critical ethical issues surrounding AI-generated code, including plagiarism, copyright, and misuse of open-source resources, along with best practices for its ethical use. Here is the plan:

- Generate Simple Python Utility Program: Create a Python code snippet for a small utility, such as a basic calculator or a text formatter.
- Analyze Ethical Issues of AI-Generated Code: Discuss ethical issues related to AI-generated code, including plagiarism, copyright violations, and misuse of open-source

Generate a simple Python calculator program  
Analyze ethical issues of AI-generated code  
Explain best practices for ethically using AI-generated code

What can I help you build? + >

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

Variables Terminal

2:10 PM Python 3

**OBSERVATION :** A key observation is that this script delivers a functional basic calculator with an interactive menu and includes essential error handling for invalid input and division by zero, making it user-friendly for simple arithmetic operations.