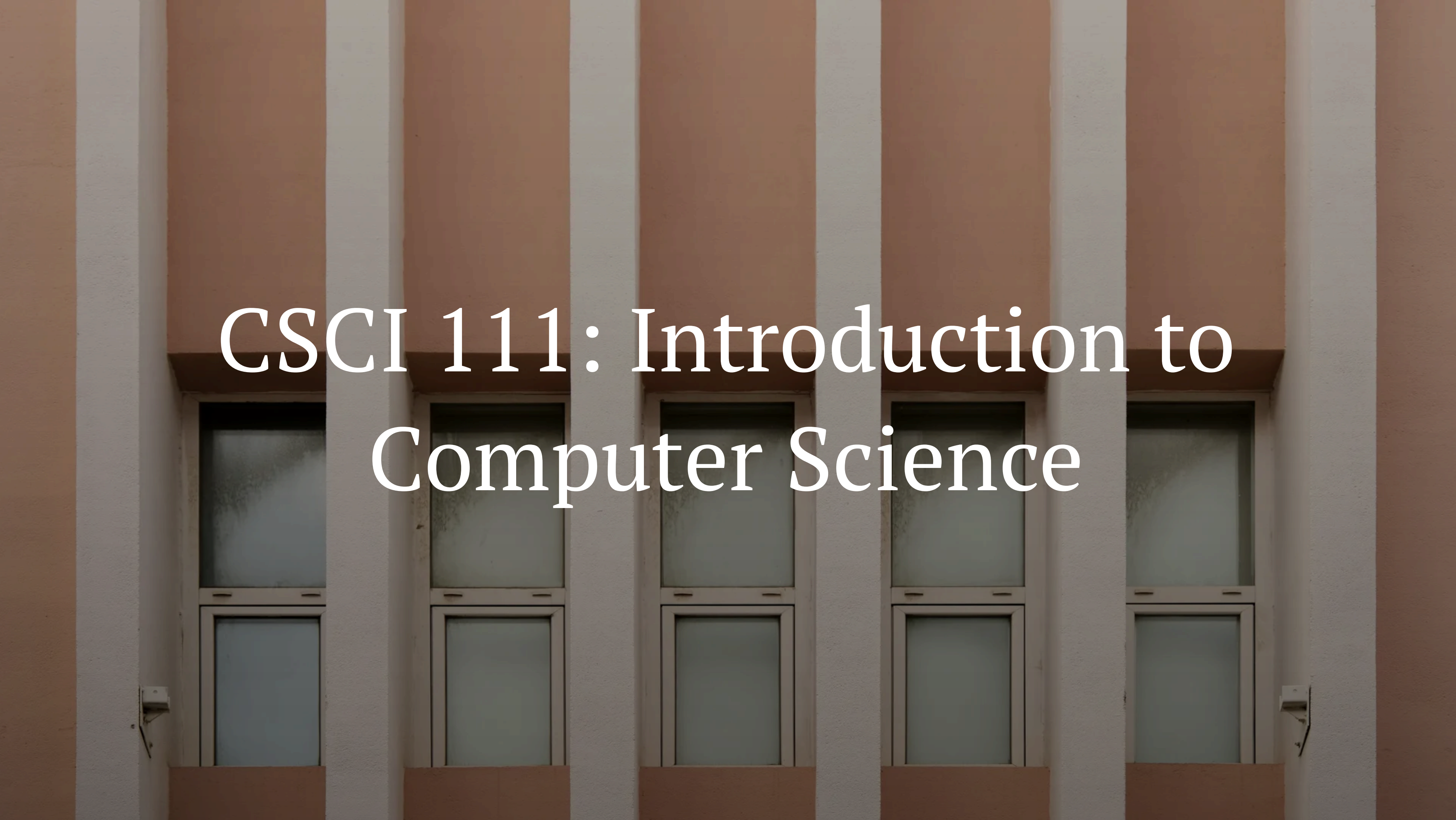

CSCI 111: Introduction to Computer Science

Loops

- Loops let you repeat a block of code multiple times.
- The most common types of loops are for loops and while loops.

Example:

```
# For-loop
for i in range(5):
    print(i)
```

The `while` Loop

- A while loop continues to run as long as its condition is true.

Example:

```
count = 0
while count < 5:
    print(count)
    count += 1
```

- The loop will stop when count reaches 5.

Loop Control: `break` and `continue`

- You can control loops with break and continue:
  - `break` : Exit the loop entirely.
  - `continue` : Skip the current iteration and move to the next one.

Example:

```python
for i in range(5):
    if i == 3:
        break
    print(i)  # Output: 0, 1, 2
```

Example:

```python
for i in range(5):
    if i == 3:
        continue
    print(i)  # Output: 0, 1, 2, 4
```

Combining Loops and Conditionals

- Loops and conditionals work together for more complex logic.

Example:

```
for i in range(10):
    if i % 2 == 0:
        print(f"{i} is even")
    else:
        print(f"{i} is odd")
```

Nested Loops

- Loops can be nested inside other loops to handle multi-dimensional data or complex tasks.

Example:

```
for i in range(3):
    for j in range(3):
        print(f"i={i}, j={j}")
```

Practice Problem: Number Guessing Game

Problem:

- Write a simple number guessing game:
  - 1. The program generates a random number between 1 and 20.
  - 2. The user has 5 tries to guess the number.
  - 3. After each guess, tell the user if they were too high, too low, or correct.

Introduction to Debugging

- **What is Debugging?**
  - Debugging is the process of finding and fixing errors in your code.
  - It ensures your program runs as intended and produces correct results.
  - Debugging is an essential skill for all developers.

Types of Errors

- **Syntax Errors**: Mistakes in the structure of the code.
- **Runtime Errors**: Errors that occur when the program is running.
- **Logical Errors**: The program runs but does not produce the expected output.

Types of Errors: Syntax Errors

- **Syntax Errors**:
  - Occur when the code structure is incorrect (e.g., missing parentheses).
  - Prevents code from being executed.
  - Example:

```
print("Hello"  # Missing closing parenthesis
```

Types of Errors: Runtime Errors

- **Runtime Errors:**
  - Errors that occur while the program is running.
  - Example: Division by zero.
  - Example:

```python
def divide(a, b):
    return a / b  # Fails if b is 0
divide(5, 0)  # Raises ZeroDivisionError
```

Types of Errors: Logical Errors

- **Logical Errors**:
  - The program runs but produces the wrong result.
  - These are often the hardest to find because the program doesn’t crash.
  - Example:

    ```
    def add(a, b):
        return a - b  # Logical error, should be a + b
    ```

Debugging Technique 1: Print Statements

- **Print Statements**:
  - The simplest and most common debugging tool.
  - Print values of variables to track their changes during execution.
  - Example:

```
def divide(a, b):
    print(f"a: {a}, b: {b}")
    return a / b
divide(10, 2)  # Output: a: 10, b: 2
```

Debugging Technique 2: Reading Error Messages

- **Reading Error Messages**:
  - Python gives you error messages that explain what went wrong.
  - Example of `ZeroDivisionError`:

    ```python
    def divide(a, b):
        return a / b
    divide(5, 0)
    ```

  - The error message:

    ```
    ZeroDivisionError: division by zero
    ```

Debugging Technique 3: Using a Debugger

- **Using a Debugger**:
  - Debuggers let you pause your code and step through it line by line.
  - Inspect variables at each step and see how the program executes.

Practice Debugging: Exercise

- **Fix the Bug:**
    - Example of a buggy function. Use print statements and error messages to debug it:

```
def divide_numbers(a, b):
    return a / b  # This will fail if b is 0

print(divide_numbers(10, 0))  # Causes ZeroDivisionError
```

Conclusion

- Debugging is an iterative process.
- Start with identifying the error type, use print statements, read error messages, and step through code with a debugger.
- The more you practice, the more effective your debugging will become.

Introduction to *Do Artifacts Have Politics?*

- **Langdon Winner's Central Thesis**:
  - Technologies are not neutral.
  - They often embed forms of power and control.
  - *Artifacts*—physical or technological—can have political consequences.

The Politics of Technology

- **What does it mean for a technology to have politics?**
  - Technologies can:
    - Reinforce existing power structures.
    - Exclude certain groups or benefit others.
    - Shape social and political relationships.

Robert Moses’ Bridges

- **Winner’s Case Study**:
  - Robert Moses, a city planner in New York, built bridges too low for buses.
  - This decision restricted access to areas like Long Island beaches for low-income and minority groups.
  - The bridge design served as a political tool for exclusion.

Technologies as Forms of Power

- **Control through Design**:
  - Technologies can centralize or decentralize power.
  - Example: Nuclear energy is highly centralized, requiring strict control.
  - Example: Solar power is decentralized, giving individuals more control over energy production.

Inherently Political Technologies

- **Inherently Political vs. Flexible Technologies**:
  - Some technologies require specific political structures.
  - Example: Nuclear power requires centralized authority.
  - Example: Solar power can function with decentralized control, giving individuals more autonomy.

Modern Technological Examples

- **Algorithms and AI**:
  - AI algorithms used in policing, hiring, or social media can reinforce social biases.
  - Example: Facial recognition technology's bias against people of color.
  - These technologies may appear neutral but have significant political implications.

Surveillance and Control

- **The Power of Surveillance Technologies**:
  - CCTV, online tracking, and facial recognition allow governments and corporations to control behavior.
  - The use of surveillance technology raises questions about privacy and autonomy.
  - Example: How governments use these tools for control, as in mass surveillance programs.

Encryption: Empowerment or Control?

- **The Role of Encryption**:
  - Encryption empowers individuals to protect privacy and secure communication.
  - However, governments and corporations often attempt to weaken encryption under the guise of “national security.”
  - Example: Debates over backdoor access to encrypted communications.

Student Discussion Prompts

- **Questions to Consider**:
  - Can technologies ever be truly neutral, or do they always serve some political purpose?
  - How do modern technologies—especially in AI and surveillance—embody political power structures?
  - How can we, as technologists, design more equitable technologies?

Common Themes in Student Responses

- **Surveillance, Algorithm Bias, and Social Media** were the most frequently discussed topics.
  - Approximately 70% of students reflected on **surveillance** and its role in modern political control.
  - **Algorithm Bias** was mentioned by around 60% of the class, particularly how social media platforms use algorithms to manipulate behavior.

Surveillance and Authoritarianism

- **Political Control Through Surveillance**:
  - Many students linked surveillance technology to the rise of authoritarianism.
  - Several mentioned the role of governments using surveillance to monitor and control citizens.
  - Example: Technologies like CCTV, internet tracking, and social media monitoring were highlighted for their potential to reinforce control.

Algorithm Bias and Political Influence

- **Algorithms as Political Tools**:
  - Over half the students discussed how algorithms, particularly in social media, reinforce existing political biases.
  - Algorithms trap users in filter bubbles, exposing them only to views they agree with, which can polarize society and limit diverse perspectives.

Social Media and Political Manipulation

- **Social Media as a Political Artifact**:
  - About 40% of students noted how social media platforms are used to manipulate public opinion.
  - They discussed how recommendation algorithms can be used to either educate or entertain depending on political goals, thus shaping political and social dynamics.

Modern Day Artifacts and Social Control

- **Technologies as Forms of Social Control**:
  - Several students drew parallels between modern algorithms and historical artifacts like bridges.
  - Algorithms today, much like the bridges of the past, can become socially legitimized tools of control, influencing public behavior and decision-making.

Unique Perspectives: Job Automation and Algorithm Legitimization

- **Job Automation**:
  - A few students discussed the political implications of job automation, particularly how algorithms are replacing jobs in fields like healthcare and education, reshaping the economy.
- **Legitimization of Algorithms**:
  - Some students raised the question of when and how algorithms themselves might become legitimized in society, potentially wielding the same influence over social behavior as historical artifacts.

Discussion Questions

- **Key Questions**:
  - How do algorithms influence politics today?
  - Can technologies like social media and algorithms ever be neutral?
  - How do we, as technologists, reduce the political biases of the tools we create?

The Role of Computer Scientists

- **Our Responsibility**:
  - As computer scientists, we must be aware of the power structures embedded in the technologies we design.
  - How can we ensure that the technologies we create are empowering rather than exclusionary?
  - Reflection: What can you do as a developer or engineer to create more equitable technologies?