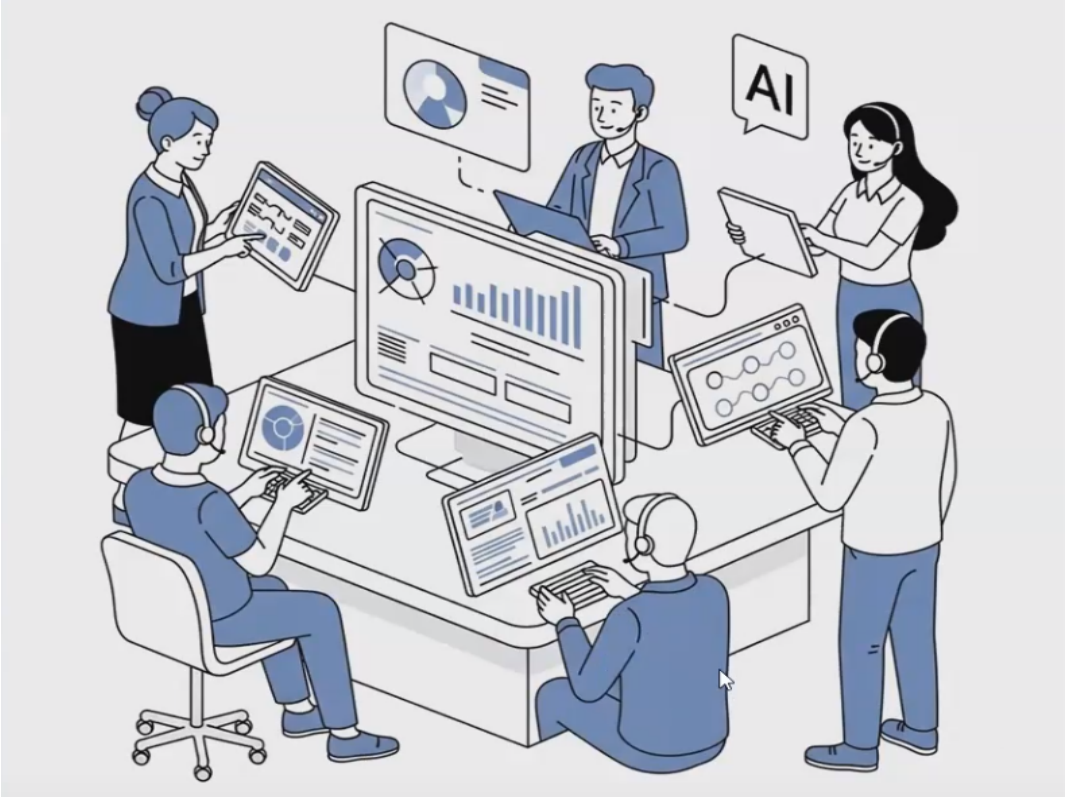
Revision on Loops, Functions

Minor in AI, IIT Ropar

10th June, 2025



1 AI Image Generation and Prompt Engineering

- **Scenario:**
  - Instructor asks students to guess the prompt used to generate an AI image.
  - Actual prompt: "show me how human life repeats in a loop."
  - AI generated image: IT professionals working in a loop, not general human routines.
- **Key Observations:**
  - **Prompt Bias:** The AI associated "loop" and "work" with IT/tech environments.
  - **Limitation Highlight:** AI models default to tech-related outputs due to their training data.
  - **Lesson:** AI needs better, more diverse training to represent broader human experiences.
- **Student Activity:**
  - Reverse-engineer the prompt from the AI-generated image.
  - Discuss why the image did not match the intended prompt.
  - Experiment with different prompts and observe the AI's output.

2 Loops and Repetition

2.1 Real-World Analogy

- Human routines (commute, work, sleep) are repetitive and can be modeled as loops.
- Example: Finding the optimal path to work becomes a subconscious routine over time.
- **Why use loops in programming?**
  - To automate repetitive tasks.
  - To process data efficiently (e.g., lists, arrays).
  - To implement algorithms that require iteration (e.g., searching, sorting).

2.2 Programming Loops Explained with Code

- **For Loop:** Used when the number of iterations is known.

```
# Print numbers from 1 to 5
for i in range(1, 6):
    print(i)
```

- `range(start, stop, step)` generates a sequence of numbers.
- The loop body executes once for each value in the sequence.

- **While Loop:** Used when the number of iterations is unknown, and you want to repeat as long as a condition is true.

```
# Simulate a user input loop until "exit" is entered
user_input = ""
while user_input.lower() != "exit":
    user_input = input("Type 'exit' to stop: ")
    print("You typed:", user_input)
```

- The loop continues until the condition becomes false.
- Be cautious to avoid infinite loops by ensuring the condition changes.

- **Key Differences:**
  - `for` loops are ideal for iterating over a known sequence.
  - `while` loops are best when the stopping condition is dynamic or unknown in advance.
  - `while` loops require careful management to prevent infinite execution.

- **Counter Concept:**
  - Use a counter variable to track the number of iterations.
  - Example:

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

2.3 Hands-On Exercise: Printing Pairs of Input and Output

- **Task:** Given two lists, print each pair of input (features) and output (targets).
- **Solution:**

```
features = [1, 2, 3, 4, 5]
targets = [10, 20, 30, 40, 50]

for i in range(len(features)):
    print("Feature:", features[i], "Target:", targets[i])
```

- **Key Points:**
  - Only one counter is needed if both lists have the same length.
  - Print statements inside the loop display results for each iteration.
  - This approach is common in data preprocessing and model training.

3 Gradient Descent and Loss Minimization

- **Scenario:**
  - Simulate gradient descent by iteratively minimizing a loss value.
  - Start with a high loss (e.g., 10) and reduce it by multiplying with a factor (e.g., 0.8) each iteration.
- **Implementation with While Loop:**

```
loss = 10.0
threshold = 0.1
iteration = 0

while loss > threshold:
    loss = loss * 0.8
    iteration += 1
    print("Iteration:", iteration, "Loss:", loss)
```

- **Key Insights:**
  - `while` loops are suitable for optimization tasks where the stopping condition is based on a threshold.
  - The loop continues until the loss is sufficiently small.
  - This pattern is widely used in machine learning for model training.

4 Functions: Modularizing Code for Reusability

4.1 Why Use Functions?

- **Reusability:** Write code once and use it multiple times.
- **Modularity:** Break down complex problems into manageable parts.
- **Maintainability:** Easier to debug and update code.

4.2 Function Example: Mean Squared Error (MSE)

- **Scenario:** Calculate the mean squared error between true and predicted values.
- **Function Definition:**

```
def mean_squared_error(y_true, y_pred):
    error = 0
    for i in range(len(y_true)):
        error += (y_true[i] - y_pred[i]) ** 2
    return error / len(y_true)
```

- **How to Use:**

```
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
print("MSE:", mean_squared_error(y_true, y_pred))
```

- **Explanation:**
  - The function iterates over each pair of true and predicted values.
  - It accumulates the squared differences.
  - Finally, it returns the average squared error.

4.3 Function Example: Prompt Analysis

- **Scenario:** Analyze an AI prompt and classify its content.
- **Function Definition:**

```
def analyze_prompt(prompt):
    if "loop" in prompt and "work" in prompt:
        return "Tech-related prompt"
    elif "routine" in prompt:
        return "General human routine prompt"
    else:
        return "Generic prompt"
```

- **How to Use:**

```
prompt = "show me how human life repeats in a loop"
print("Prompt analysis:", analyze_prompt(prompt))
```

5 Key Takeaways

- **AI and Programming:**
  - AI models can reflect biases from their training data.
  - Understanding these limitations is crucial for building better systems.
- **Programming Fundamentals:**
  - Loops and functions are essential for modeling repetitive tasks and modularizing code.
  - Choose the right loop based on whether the number of iterations is known.
  - Functions help break down complex problems and promote code reuse.
- **Real-World Application:**
  - Human routines and optimization problems can be modeled using loops and functions.
  - Always test and debug code to ensure it matches the intended logic.
- **Exam Questions:**
  - May ask to explain the difference between `for` and `while` loops.
  - Could involve writing a function to calculate a metric (e.g., MSE).
  - May require understanding how AI models can be biased based on training data.

6 Summary Table

Concept	**Description**
For Loop	Iterates a set number of times, ideal for known sequences
While Loop	Repeats while a condition is true, suitable for dynamic stopping
Function	Modularizes code for reusability and maintainability
AI Image Generation	Can reflect biases from training data
Prompt Engineering	Crafting prompts to guide AI output
Gradient Descent	Iterative optimization algorithm
Mean Squared Error	Metric for regression model evaluation