Activity 1: Create and Update Pages Based on Conditions

Objective:

Learners will create a script that uses **conditionals** to check the current date and update an XWiki page with a message depending on the day of the week.

Steps:

1. **Create a Groovy script** that gets the current day of the week.
2. **Use conditionals** (`if`, `else`) to check:
    - If it's Monday, update the page with the message: "Start of the Week!"
    - If it's Friday, update the page with: "Almost Weekend!"
    - For any other day, update the page with: "Have a productive day!"
3. **Save the page** and show the corresponding message based on the current day.

Expected Outcome:

- Learners will practice **using conditionals** to handle logic and **update an XWiki page** based on dynamic conditions.

Activity 2: List All Users and Their Roles in XWiki

Objective:

Learners will write a Groovy script that retrieves all the users and their roles in the wiki, using **loops** and **arrays**.

Steps:

1. Use the **XWiki API** to get a list of all users in the wiki.
2. For each user, print out their username and the roles they belong to.
3. The script should use a **for loop** or **each** to iterate through the list of users.
4. Display the results in the console or return them in the page output.

Expected Outcome:

- Learners will practice **working with collections (arrays and lists)** and **iterating over them** using loops.

Activity 3: Function to Calculate Total Word Count

Objective:

Learners will define a function that calculates the **total word count** of a given text in the wiki.

Steps:

1. **Define a function** `countWords(text)` that accepts a string and counts the number of words in it.
2. Use **string manipulation** (`split()`, `size()`) to break the text into words and return the word count.
3. **Get the content** of a page and use the function to calculate the word count of the page's content.
4. Display the word count on the page.

Expected Outcome:

- Learners will practice **defining functions** and working with **strings** to manipulate text.

Activity 4: Creating a To-Do List with Add and Remove Features

Objective:

Learners will build a simple **To-Do list** using **arrays (lists)** and **functions** to allow adding and removing items dynamically.

Steps:

1. **Create a list** of tasks (e.g., `tasks = ["Buy groceries", "Finish homework"]`).
2. Define functions:
   - `addTask(task)` to add a task to the list.
   - `removeTask(task)` to remove a task from the list.
3. Use a **loop** (`each`) to print all tasks to the page.
4. Implement functionality to **add** and **remove tasks** dynamically and show the updated list.

Expected Outcome:

- Learners will practice working with **lists**, **functions**, and **dynamic content**.

Activity 5: Sum of Numbers in a List

Objective:

Learners will write a function that calculates the **sum of numbers** in a list using **loops**.

Steps:

1. Define a list of numbers: `numbers = [1, 2, 3, 4, 5].`
2. Create a function `sumNumbers(list)` that iterates over the list and returns the sum of the numbers.
3. Call the function and display the result.

Expected Outcome:

- Learners will practice working with **loops**, **arrays**, and **functions** to process data.

Activity 6: Create a Simple Greeting Based on User Input

Objective:

Learners will create a Groovy script that accepts a user's **name** and outputs a personalized greeting.

Steps:

1. Define a function `greetUser(name)` that accepts a name as a parameter and returns a greeting (e.g., "Hello, [name]!").
2. Accept **user input** for the name.
3. Display the greeting message on the page or in the console.

Expected Outcome:

- Learners will practice **using functions** and **string interpolation** to personalize content dynamically.

Activity 7: Reverse a String and Check Palindrome

Objective:

Learners will write a Groovy script to **reverse a string** and check if it's a **palindrome** (same forwards and backwards).

Steps:

1. Define a function `reverseString(str)` that takes a string and returns the reversed version of it.
2. Check if the original string is the same as the reversed string.
3. Print out whether the string is a **palindrome** or not.

Expected Outcome:

- Learners will practice **string manipulation**, **functions**, and **conditions**.

Activity 8: Display Fibonacci Sequence up to N Terms

Objective:

Learners will generate the **Fibonacci sequence** up to `n` terms, where `n` is specified by the user.

Steps:

1. Define a function `generateFibonacci(n)` that generates the Fibonacci sequence.
2. Use **loops** to generate and print the sequence up to `n` terms.
3. Allow the user to specify `n` and display the sequence on the page.

Expected Outcome:

- Learners will practice **functions**, **loops**, and **user input** to generate dynamic content.

Activity 9: Create a Custom Greeting Message Based on Time of Day

Objective:

Learners will create a Groovy script that provides a custom greeting based on the time of day (morning, afternoon, evening).

Steps:

1. Get the **current time** of day.
2. Use **conditionals** to check if it's morning, afternoon, or evening.
3. Display a corresponding message (e.g., "Good Morning!", "Good Afternoon!", "Good Evening!").

Expected Outcome:

- Learners will practice **working with date and time**, **conditionals**, and **string manipulation**.

Activity 10: Count the Occurrence of a Word in a Text

Objective:

Learners will write a Groovy script that counts how many times a specific word appears in the content of an XWiki page.

Steps:

1. Get the **content of a page**.
2. Define a function `countWordOccurrences(text, word)` that counts how many times `word` appears in `text`.
3. Display the result on the page (e.g., "The word 'Groovy' appears X times.").

Expected Outcome:

- Learners will practice **string manipulation**, **functions**, and **iterating through content**.

Activity 11: Build a Simple Calculator

Objective:

Learners will create a simple calculator that can add, subtract, multiply, and divide two numbers.

Steps:

1. Define a function for each operation: `add(a, b)`, `subtract(a, b)`, `multiply(a, b)`, and `divide(a, b)`.
2. Prompt the user for two numbers and the operation they want to perform.
3. Display the result of the calculation.

Expected Outcome:

- Learners will practice **functions**, **user input**, and basic **mathematical operations**.

Activity 12: Generate a Random Number and Guessing Game

Objective:

Learners will write a script that generates a **random number** and asks the user to guess it.

Steps:

1. Generate a random number between 1 and 100.
2. Allow the user to guess the number and give feedback if the guess is too high or too low.
3. Repeat until the user guesses correctly.

Expected Outcome:

- Learners will practice **working with random numbers**, **user input**, and **loops**.

🧪 XWiki Groovy Scripting Lab Activities

Activity 1: Hello World Refresher

📌 **Goal:** Run a basic Groovy script in a wiki page.

Steps:

1. Open or create a new wiki page.
2. Insert the script macro with:

```
groovy

{{groovy}}
println("Hello, world from Groovy in XWiki!")
{{/groovy}}
```

3. Save and view the output.

✅ **Expected Output:**

```
Hello, world from Groovy in XWiki!
```

Activity 2: Use a Variable in Groovy

📌 **Goal:** Define and use a variable in a script.

Steps:

1. Use the code:

```
groovy

{{groovy}}
def name = "Alice"
def greeting = "Hello, ${name}!"
return greeting
{{/groovy}}
```

2. Save and view the page.

✅ **Expected Output:**

```
Hello, Alice!
```

🔁 **Try:** Change `Alice` to another name and observe the output.

Activity 3: Use If-Else Logic

📌 **Goal:** Use conditional logic to display different messages.

Steps:

```
groovy

{{groovy}}
def hour = new Date().format("H").toInteger()
if (hour < 12) {
  return "Good morning!"
} else {
  return "Good afternoon!"
}
{{/groovy}}
```

✅ **Expected Output:**
A greeting based on the current hour.

Activity 4: Display a List of Pages

📌 **Goal:** Use XWiki API to list documents in a space.

Steps:

```
groovy

{{groovy}}
def docs = xwiki.getStore().searchDocuments("where doc.space = 'Sandbox'")
return docs.join("<br/>")
{{/groovy}}
```

✅ **Expected Output:**
List of all pages in the "Sandbox" space.

Activity 5: Use Form Input with Groovy

📌 **Goal:** Capture and use user input on a page.

Steps:

```
xwiki

{{html clean="false"}}
<form method="post">
  <input type="text" name="name" placeholder="Enter your name" />
  <input type="submit" value="Greet Me!" />
</form>
{{/html}}
```

```
{{groovy}}
def name = request.get("name")
if (name) {
  return "Hello, ${name}!"
}
{{/groovy}}
```

✅ **Expected Output:**
User sees a form, enters a name, and gets a personalized greeting.

Activity 6: Create a New Page via Script

📌 **Goal:** Learn how to programmatically create a new page.

Steps:

```
groovy

{{groovy}}
def doc = xwiki.getDocument("Sandbox.GroovyCreatedPage")
doc.setContent("This page was created with Groovy!")
xwiki.saveDocument(doc)
return "Page 'GroovyCreatedPage' has been created in the Sandbox space."
{{/groovy}}
```

✅ **Expected Output:**
New page appears under `Sandbox.GroovyCreatedPage`.

Bonus Activity: Combine Input and Page Creation

📌 **Goal:** Take user input and use it to create a personalized wiki page.

🏁 Wrap-Up Task: Mini Project

Build a simple "Guestbook" page:

- A form where users can enter their name and message.
- Stores the input on a new subpage or logs it on the same page.
- Uses XWiki and Groovy features together.