3. Student Activity

Student Activity: Practicing AI-Powered Code Generation

Welcome to the hands-on activity session on AI-Powered Code Generation! In this activity, you'll practice creating effective prompts and using AI to generate code. Follow the steps and examples provided, and try them out on your own.

Activity 1: Understanding AI-Powered Code Generation

Objective: Learn how AI generates code from prompts.

Steps:

1. **Simple Function Generation:**
   - **Prompt:** "Write a Python function that multiplies two numbers."
   - **Expected Output:**

```python
def multiply_numbers(a, b):
    return a * b
```

2. **Conditional Logic:**
   - **Prompt:** "Write a Python function that checks if a number is even."
   - **Expected Output:**

```python
def is_even(number):
    return number % 2 == 0
```

3. **String Manipulation:**
   - **Prompt:** "Write a Python function that reverses a string."
   - **Expected Output:**

```python
def reverse_string(s):
    return s[::-1]
```

Practice: Try creating your own prompts for simple functions and see what the AI generates.

Activity 2: Creating Effective Prompts

Objective: Learn how to create clear and specific prompts for better code generation.

Steps:

1. **Specific Task:**
   - **Prompt:** "Write a Python function that calculates the factorial of a number."
   - **Expected Output:**

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

2. **Providing Context:**
   - **Prompt:** "Write a Python script that reads a file and prints its contents line by line."
   - **Expected Output:**

```python
def read_file(file_path):
    with open(file_path, 'r') as file:
        for line in file:
            print(line.strip())
```

3. **Iterative Improvement:**
   - **Initial Prompt:** "Write a Python program that sorts a list."
   - **Refined Prompt:** "Write a Python program that sorts a list of integers in ascending order using the bubble sort algorithm."
   - **Expected Output:**

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

Practice: Create prompts for tasks you want to automate and refine them for clarity and specificity.

Activity 3: Real-World Applications

Objective: Use AI to generate code for practical applications.

Steps:

1. **Building a Simple App:**
   - **Prompt:** "Write a Python program that simulates a basic to-do list app with add and remove functionality."
   - **Expected Output:**

```
def todo_list():
    tasks = []
    while True:
        action = input("Enter 'add' to add a task, 'remove' to remove
a task, or 'quit' to exit: ")
        if action == 'add':
            task = input("Enter task: ")
            tasks.append(task)
        elif action == 'remove':
            task = input("Enter task to remove: ")
            if task in tasks:
                tasks.remove(task)
        elif action == 'quit':
            break
        print("Current tasks:", tasks)

todo_list()
```

2. **Creating a Web Page:**
   - **Prompt:** "Generate HTML code for a simple webpage with a header, paragraph, and footer."
   - **Expected Output:**

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple Webpage</title>
```

```
</head>
<body>
    <header>
        <h1>Welcome to My Webpage</h1>
    </header>
    <p>This is a simple paragraph on my webpage.</p>
    <footer>
        <p>Footer content goes here.</p>
    </footer>
</body>
</html>
```

3. **Automating a Task:**
    - **Prompt:** "Write a Python script that renames all .txt files in a directory to have a '_backup' suffix."
    - **Expected Output:**

```
import os

def rename_files(directory):
    for filename in os.listdir(directory):
        if filename.endswith('.txt'):
            new_name = filename.replace('.txt', '_backup.txt')
            os.rename(os.path.join(directory, filename),
os.path.join(directory, new_name))

rename_files('/path/to/directory')
```

Practice: Think of a small project or task you want to automate and create prompts to generate the necessary code.

Conclusion

By completing these activities, you should have a better understanding of how to use AI for code generation. Remember, the key to success is creating clear and specific prompts and iterating on them to achieve the desired results. Keep practicing with different prompts and explore the possibilities of AI-powered code generation!