3. Student Activity

Student Activity: Debugging and Code Optimization with AI

Welcome to the hands-on session where you'll practice the concepts of debugging and code optimization with AI. This activity is designed to reinforce your understanding through practical examples. Let's get started!

Part 1: Debugging with AI

Example 1: Syntax Error Detection

1. **Task**: Write a simple Python function that adds two numbers.
2. **Error Introduction**: Intentionally introduce a syntax error by misspelling a keyword.
3. **AI Assistance**: Use an AI-powered code editor or tool to detect the syntax error and suggest corrections.

```
def add_numbers(a, b):
    return a + b

# Introduce an error
def add_numbers(a, b)
    return a + b
```

Example 2: Logical Error Detection

1. **Task**: Write a function to calculate the square of a number.
2. **Error Introduction**: Introduce a logical error by using the wrong operator.
3. **AI Assistance**: Use AI to identify the logical error and suggest the correct operator.

```
def square_number(x):
    return x * x

# Introduce a logical error
def square_number(x):
    return x + x
```

Example 3: Runtime Error Detection

1. **Task**: Write a function to divide two numbers.
2. **Error Introduction**: Introduce a runtime error by dividing by zero.
3. **AI Assistance**: Use AI to detect the potential runtime error and suggest handling it with a try-except block.

```
def divide_numbers(a, b):
    return a / b

# Introduce a runtime error
def divide_numbers(a, b):
    return a / 0
```

Part 2: Code Optimization with AI

Example 1: Refactoring Code

1. **Task**: Write a function with repetitive code.
2. **Optimization**: Use AI to refactor the code by removing redundancy.
3. **AI Assistance**: AI suggests using a loop to eliminate repetitive lines.

```
def print_numbers():
    print(1)
    print(2)
    print(3)

# AI refactoring suggestion
def print_numbers():
    for i in range(1, 4):
        print(i)
```

Example 2: Reducing Memory Usage

1. **Task**: Write a function that creates a large list.
2. **Optimization**: Use AI to suggest using a generator to reduce memory usage.
3. **AI Assistance**: AI suggests using a generator expression.

```
def create_large_list(n):
    return [i for i in range(n)]

# AI memory optimization suggestion
def create_large_list(n):
    return (i for i in range(n))
```

Example 3: Improving Execution Speed

1. **Task**: Write a function that performs a slow operation.
2. **Optimization**: Use AI to identify bottlenecks and suggest improvements.
3. **AI Assistance**: AI suggests using a more efficient algorithm or data structure.

```
def slow_operation(data):
    result = []
    for item in data:
        if item not in result:
            result.append(item)
    return result

# AI speed optimization suggestion
def fast_operation(data):
    return list(set(data))
```

Part 3: Incorporating Best Practices and Coding Standards

Example 1: Variable Naming

1. **Task**: Write a function with poorly named variables.
2. **Improvement**: Use AI to suggest better variable names.
3. **AI Assistance**: AI suggests descriptive names for clarity.

```
def calc(a, b):
    return a + b

# AI suggestion for better variable names
```

```
def calculate_sum(number1, number2):
    return number1 + number2
```

Example 2: Consistent Indentation

1. **Task**: Write a function with inconsistent indentation.
2. **Improvement**: Use AI to enforce consistent indentation.
3. **AI Assistance**: AI suggests correcting indentation for readability.

```
def check_even_odd(number):
if number % 2 == 0:
    print("Even")
else:
    print("Odd")

# AI suggestion for consistent indentation
def check_even_odd(number):
    if number % 2 == 0:
        print("Even")
    else:
        print("Odd")
```

Example 3: Following Industry Standards

1. **Task**: Write a function that doesn't follow PEP 8 standards.
2. **Improvement**: Use AI to ensure the code follows PEP 8 standards.
3. **AI Assistance**: AI suggests formatting changes to adhere to standards.

```
def multiply(a,b):return a*b

# AI suggestion for PEP 8 compliance
def multiply(a, b):
    return a * b
```

Conclusion

Through these examples, you've practiced using AI to assist with debugging and code optimization. Remember, AI is a tool to enhance your coding skills, but it's important to review

and understand the suggestions it provides. Keep practicing, and you'll become more proficient in writing efficient and error-free code!

Feel free to ask any questions or seek clarification on any of the examples. Happy coding!