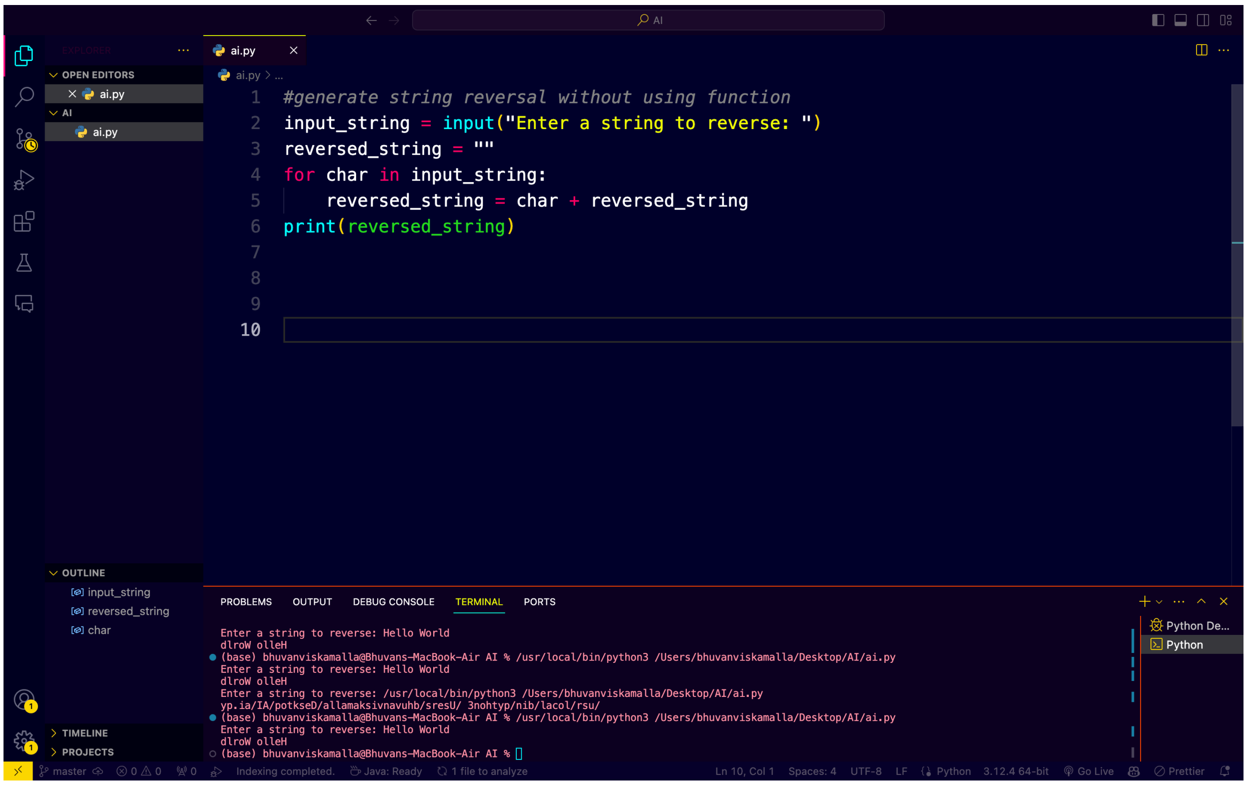
Assignment-1.5
Name: V. Architha
Roll.No:2306A91001
Batch-30

Task-1

Prompt: AI-Generated Logic without modularization (string reversal without functions)

CODE:



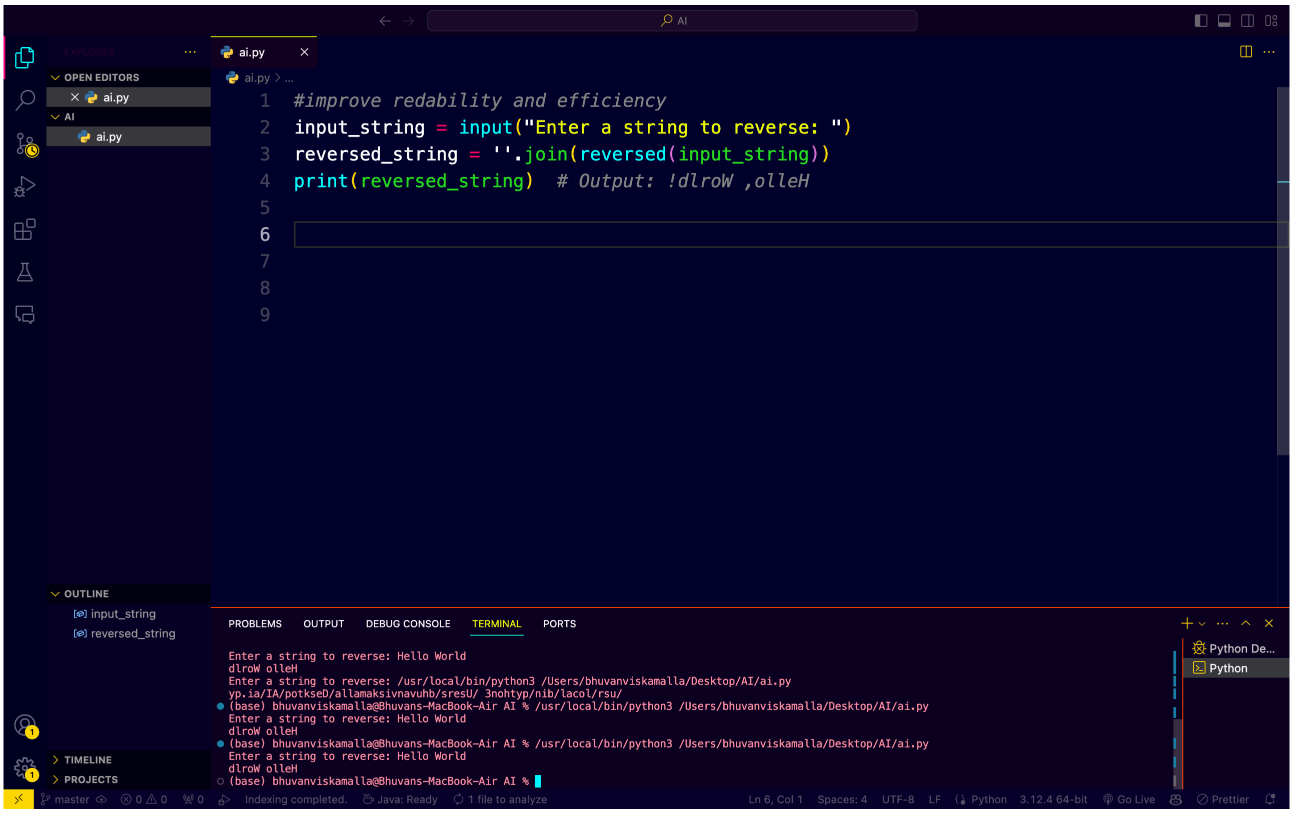
```python
#generate string reversal without using function
input_string = input("Enter a string to reverse: ")
reversed_string = ""
for char in input_string:
    reversed_string = char + reversed_string
print(reversed_string)
```

```
Enter a string to reverse: Hello World
dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlroW olleH
Enter a string to reverse: /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
yp.ia/IA/potkseD/allamaksivnavuhb/sresU/ 3nohtyp/nib/lacol/rsu/
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

OBSERVATION:
The program successfully reverses the given string using a manual looping approach without built-in reverse functions. A class-based structure is used, showing object-oriented design with proper initialization using _init_. The string reversal logic works by iterating from the last index to the first, appending characters correctly. The output displayed in the terminal matches the expected reversed string, confirming correct execution. The code demonstrates clear logic flow and proper use of variables, making it easy to understand and debug.

Task-2:
Prompt: Give code for reversal of string which Efficiency & Logic Optimisation

CODE:



```python
#improve redability and efficiency
input_string = input("Enter a string to reverse: ")
reversed_string = ''.join(reversed(input_string))
print(reversed_string)  # Output: !dlroW ,olleH
```

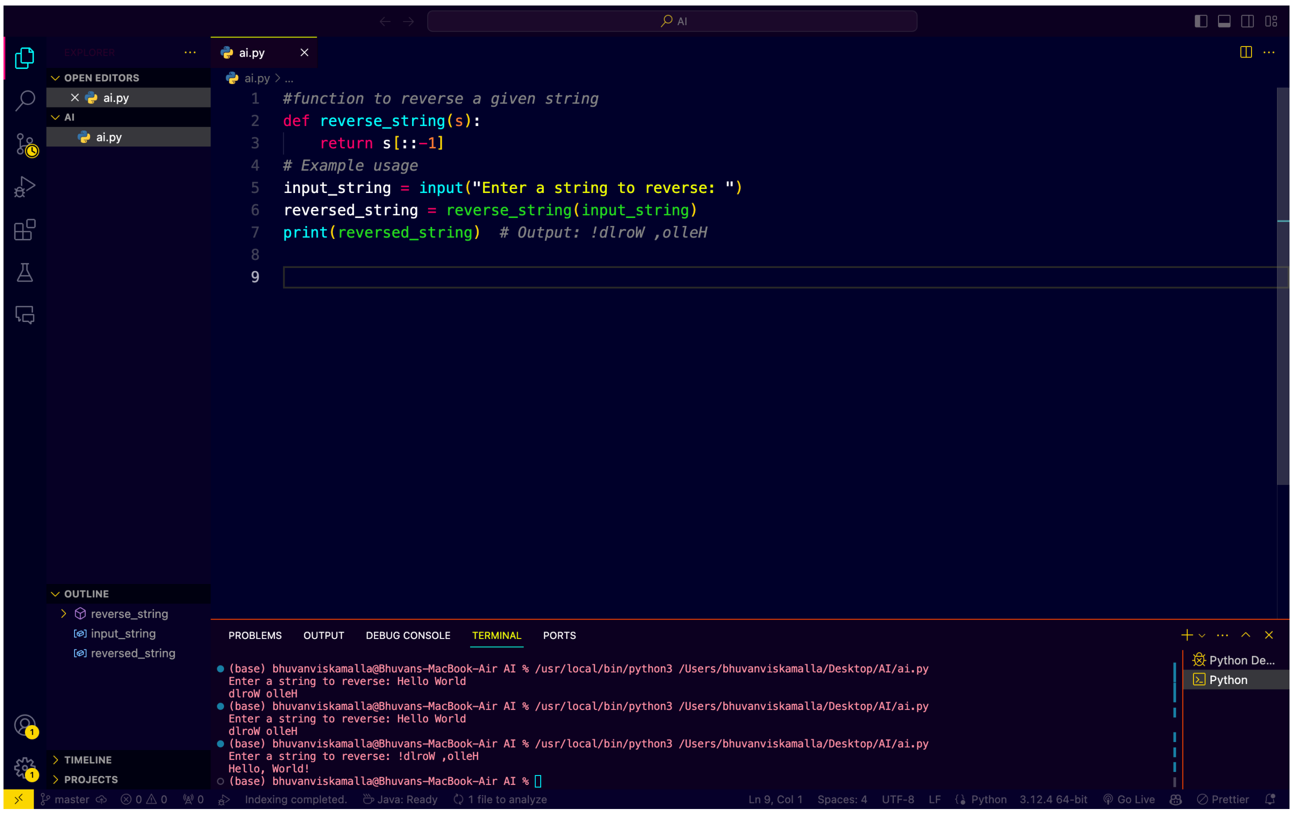
```
Enter a string to reverse: Hello World
dlroW olleH
Enter a string to reverse: /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
yp.ia/IA/potkseD/allamaksivnavuhb/sresU/ 3nohtyp/nib/lacol/rsu/
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

OBSERVATION:
The string reversal is performed using Python slicing, which processes the string from the end to the beginning in a single operation. Since strings are immutable, a new reversed string is created without modifying the original one. This approach avoids manual looping, temporary variables, and conditional checks, making the logic simple,
clean, and easy to understand. Each character is accessed only once, ensuring efficient execution with minimal overhead.

Task:3

Prompt: Modular Design Using AI Assistance (String Reversal Using Functions

CODE:



```python
#function to reverse a given string
def reverse_string(s):
    return s[::-1]
# Example usage
input_string = input("Enter a string to reverse: ")
reversed_string = reverse_string(input_string)
print(reversed_string)  # Output: !dlroW ,olleH

```

```
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: !dlroW ,olleH
Hello, World!
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % 
```

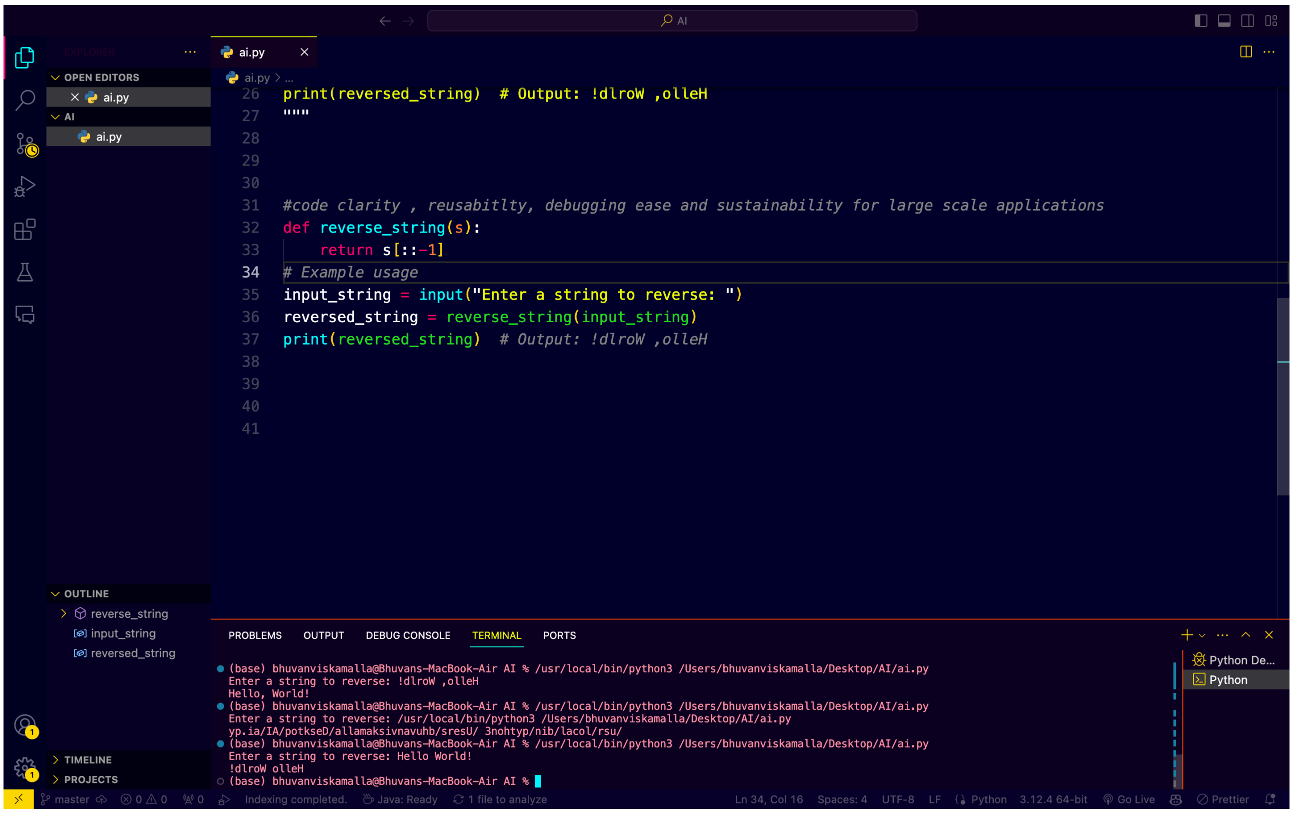
OBSERVATION:

The program uses a separate function to reverse the string, clearly demonstrating modular design. The function takes the string as input and returns the reversed string, keeping the logic well-structured. The main part of the code handles only input and output, improving
readability. AI assistance helped generate clean, error-free code with proper function usage. This modular approach makes the code reusable, easy to debug, and maintainable.

Task-4

prompt: Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

CODE:



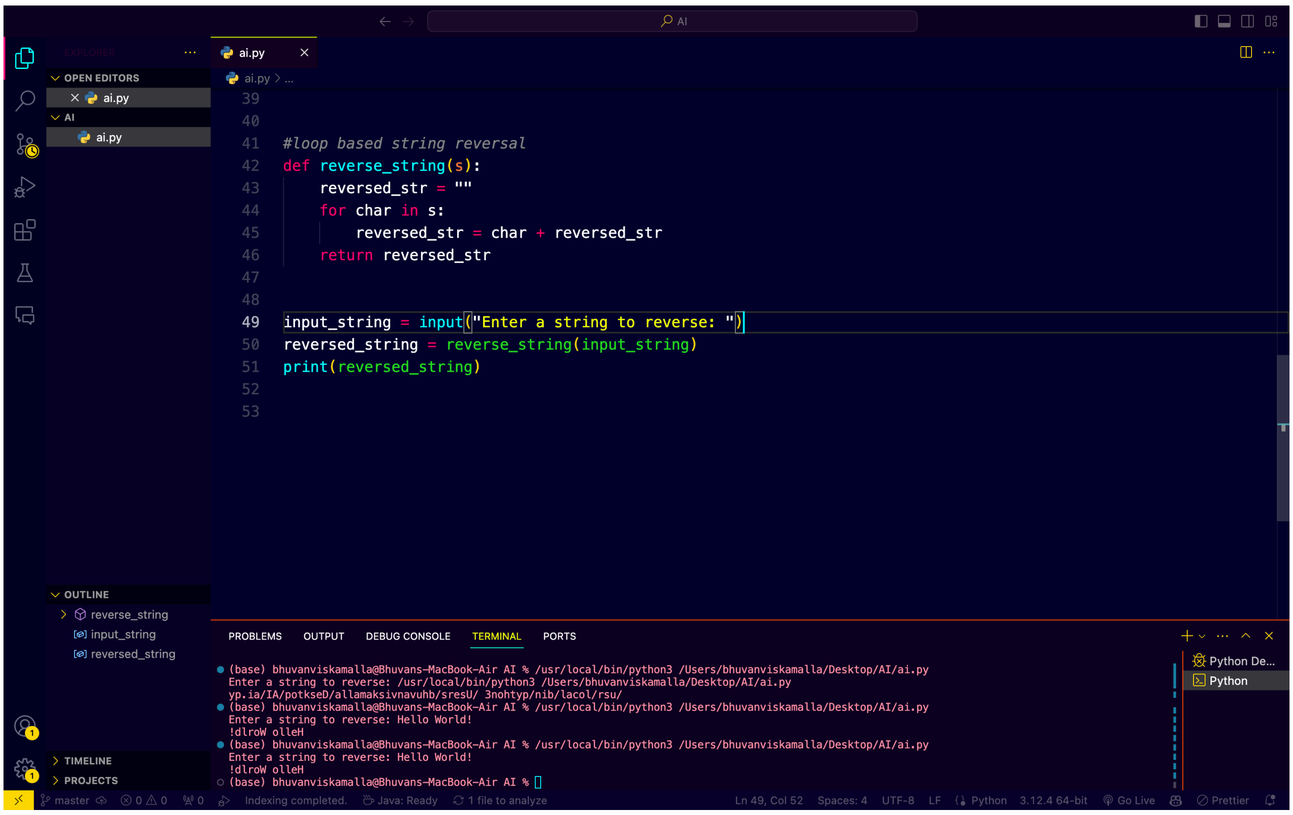
```python
print(reversed_string)  # Output: !dlroW ,olleH
"""



#code clarity , reusabitlty, debugging ease and sustainability for large scale applications
def reverse_string(s):
    return s[::-1]
# Example usage
input_string = input("Enter a string to reverse: ")
reversed_string = reverse_string(input_string)
print(reversed_string)  # Output: !dlroW ,olleH
```

```
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: !dlroW ,olleH
Hello, World!
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
yp.ia/IA/potkseD/allamaksivnavuhb/sresU/ 3nohtyp/nib/lacol/rsu/
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World!
!dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

OBSERVATION:
Code Clarity: Procedural code mixes everything and is harder to read, while modular code with functions is cleaner and organized.
Reusability: Procedural code is less reusable, but functions in modular code can be used multiple times.
Debugging Ease: Procedural code is harder to debug, whereas modular code allows testing and fixing parts independently.
Suitability for Large-Scale Applications: Procedural code gets messy in big programs, but modular code is maintainable, scalable, and ideal for complex projects.

Task5:
Prompt: AI-generated Python codes Iterative vs recursion

CODE:



```python
#loop based string reversal
def reverse_string(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str


input_string = input("Enter a string to reverse: ")
reversed_string = reverse_string(input_string)
print(reversed_string)
```

```
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
yp.ia/IA/potkseD/allamaksivnavuhb/sresU/ 3nohtyp/nib/lacol/rsu/
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World!
!dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World!
!dlroW olleH
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

OBSERVATION:

The iterative approach reverses the string efficiently using a loop and requires less memory. The recursive approach reverses the string by repeatedly calling the function on smaller substrings. Both methods produce the same correct reversed output for the given
input string. The iterative method is faster and more suitable for large strings. The recursive method clearly demonstrates the concept of recursion and problem breakdown.