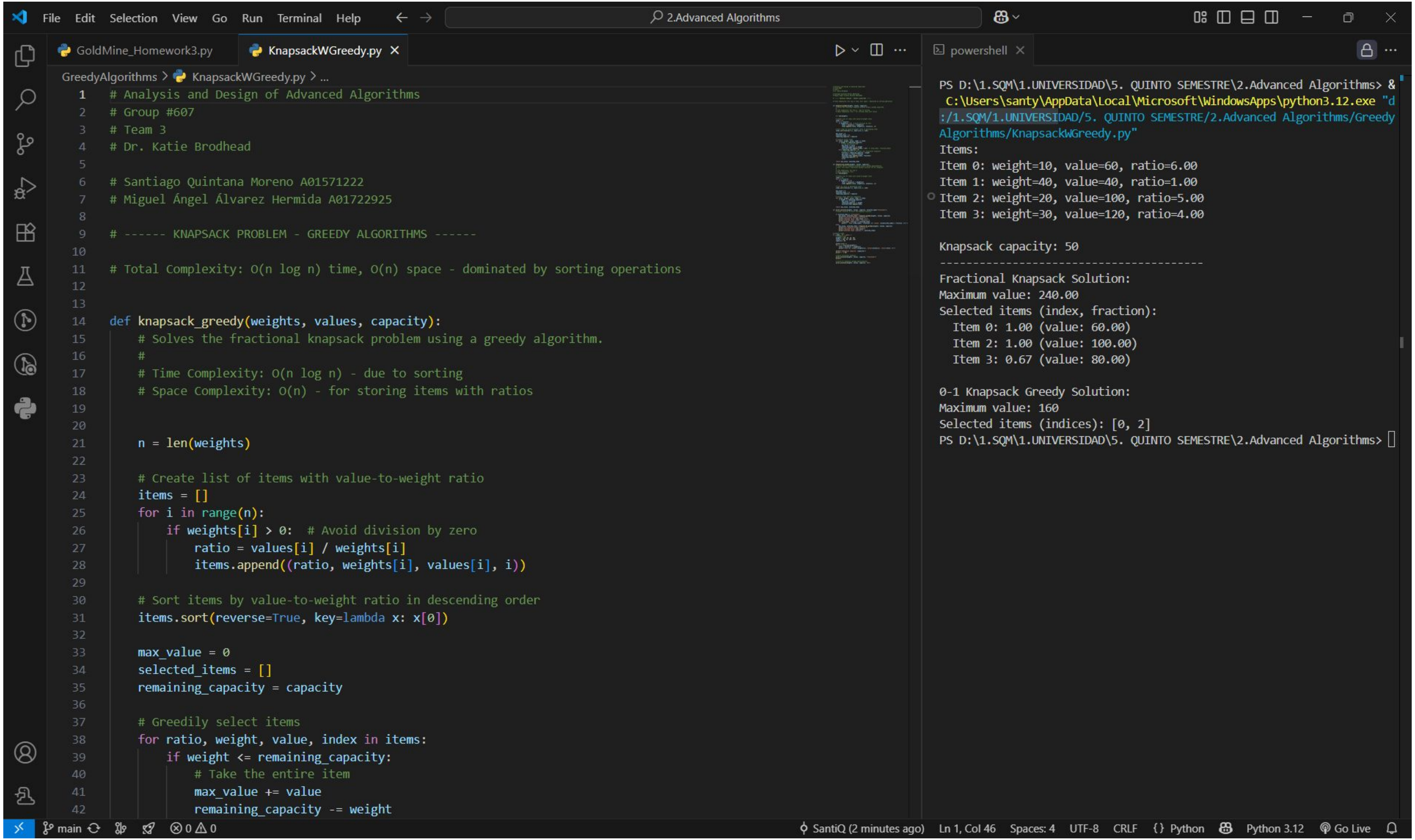
Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

Homework 4:
Knapsack Problem using Greedy Algorithm

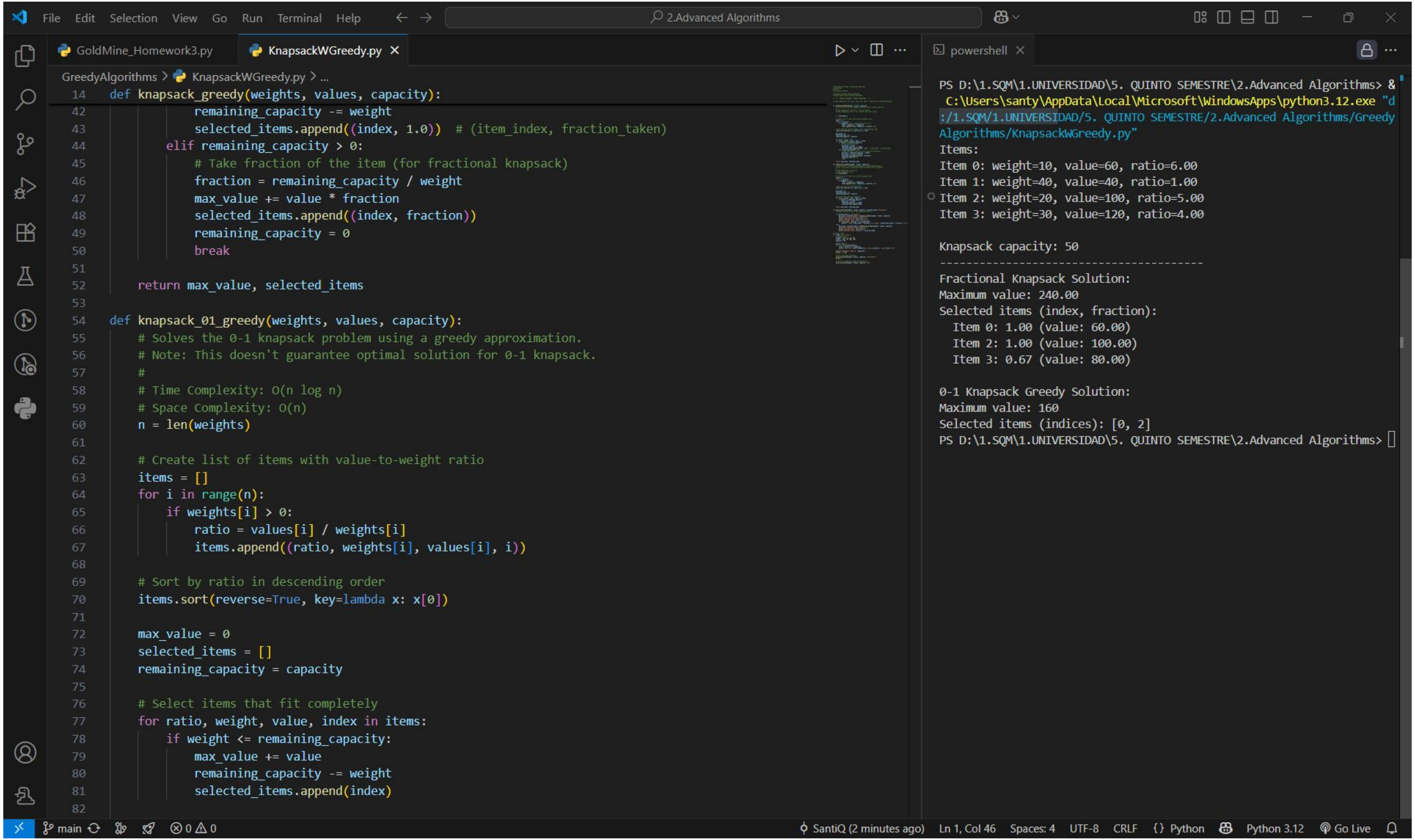
Group: 607
Team #3
Dr. Katie Brodhead

Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925



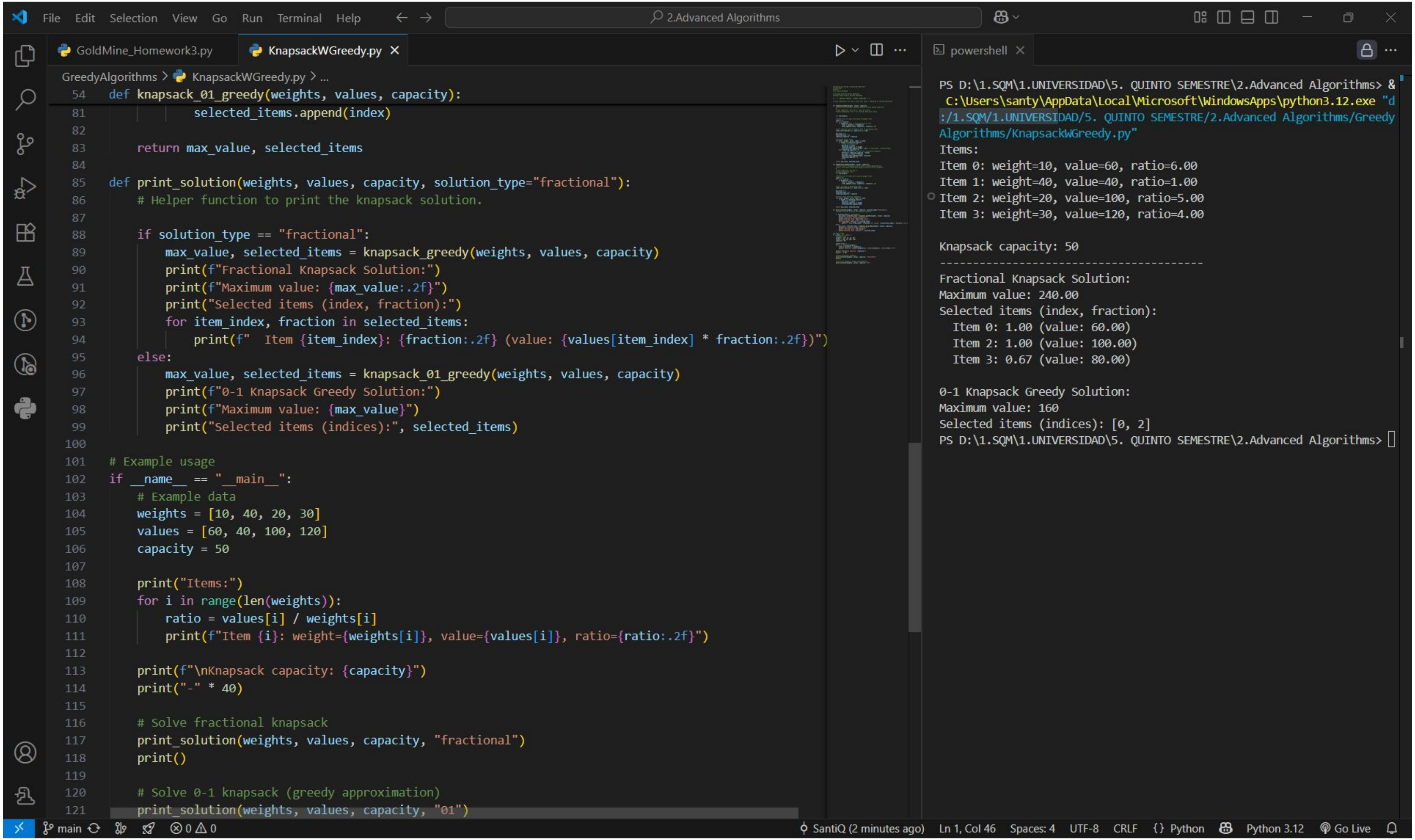
```python
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 3
# Dr. Katie Brodhead

# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925

# ------ KNAPSACK PROBLEM - GREEDY ALGORITHMS ------

# Total Complexity: O(n log n) time, O(n) space - dominated by sorting operations


def knapsack_greedy(weights, values, capacity):
    # Solves the fractional knapsack problem using a greedy algorithm.
    #
    # Time Complexity: O(n log n) - due to sorting
    # Space Complexity: O(n) - for storing items with ratios


    n = len(weights)

    # Create list of items with value-to-weight ratio
    items = []
    for i in range(n):
        if weights[i] > 0:  # Avoid division by zero
            ratio = values[i] / weights[i]
            items.append((ratio, weights[i], values[i], i))

    # Sort items by value-to-weight ratio in descending order
    items.sort(reverse=True, key=lambda x: x[0])

    max_value = 0
    selected_items = []
    remaining_capacity = capacity

    # Greedily select items
    for ratio, weight, value, index in items:
        if weight <= remaining_capacity:
            # Take the entire item
            max_value += value
            remaining_capacity -= weight
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Greedy Algorithms/KnapsackWGreedy.py"
Items:
Item 0: weight=10, value=60, ratio=6.00
Item 1: weight=40, value=40, ratio=1.00
Item 2: weight=20, value=100, ratio=5.00
Item 3: weight=30, value=120, ratio=4.00

Knapsack capacity: 50
----------------------------------------
Fractional Knapsack Solution:
Maximum value: 240.00
Selected items (index, fraction):
  Item 0: 1.00 (value: 60.00)
  Item 2: 1.00 (value: 100.00)
  Item 3: 0.67 (value: 80.00)

0-1 Knapsack Greedy Solution:
Maximum value: 160
Selected items (indices): [0, 2]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> 
```



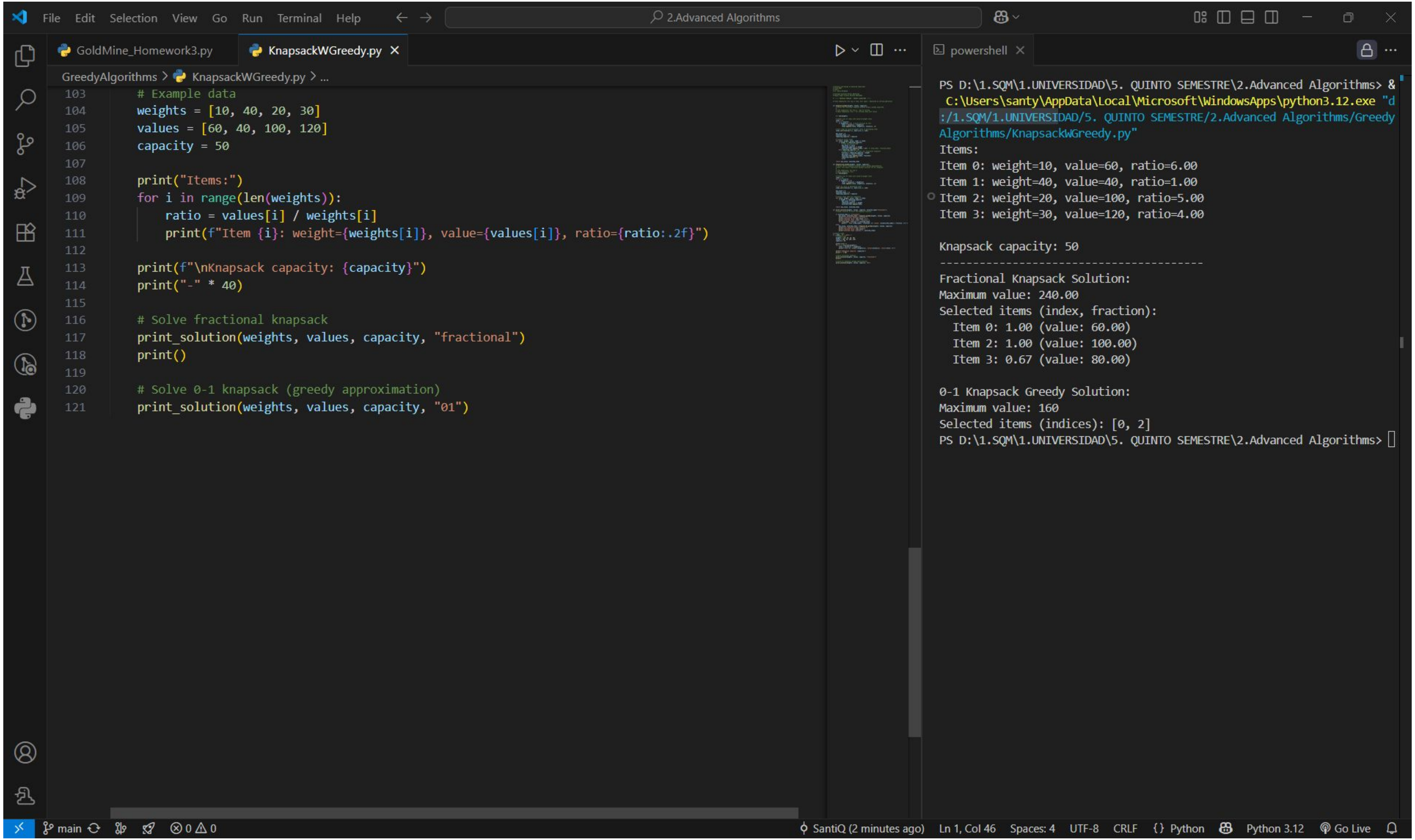
```python
def knapsack_greedy(weights, values, capacity):
            remaining_capacity -= weight
            selected_items.append((index, 1.0))  # (item_index, fraction_taken)
        elif remaining_capacity > 0:
            # Take fraction of the item (for fractional knapsack)
            fraction = remaining_capacity / weight
            max_value += value * fraction
            selected_items.append((index, fraction))
            remaining_capacity = 0
            break

    return max_value, selected_items

def knapsack_01_greedy(weights, values, capacity):
    # Solves the 0-1 knapsack problem using a greedy approximation.
    # Note: This doesn't guarantee optimal solution for 0-1 knapsack.
    #
    # Time Complexity: O(n log n)
    # Space Complexity: O(n)
    n = len(weights)

    # Create list of items with value-to-weight ratio
    items = []
    for i in range(n):
        if weights[i] > 0:
            ratio = values[i] / weights[i]
            items.append((ratio, weights[i], values[i], i))

    # Sort by ratio in descending order
    items.sort(reverse=True, key=lambda x: x[0])

    max_value = 0
    selected_items = []
    remaining_capacity = capacity

    # Select items that fit completely
    for ratio, weight, value, index in items:
        if weight <= remaining_capacity:
            max_value += value
            remaining_capacity -= weight
            selected_items.append(index)

```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> &
 C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d
:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Greedy
Algorithms/KnapsackWGreedy.py"
Items:
Item 0: weight=10, value=60, ratio=6.00
Item 1: weight=40, value=40, ratio=1.00
Item 2: weight=20, value=100, ratio=5.00
Item 3: weight=30, value=120, ratio=4.00

Knapsack capacity: 50
----------------------------------------
Fractional Knapsack Solution:
Maximum value: 240.00
Selected items (index, fraction):
  Item 0: 1.00 (value: 60.00)
  Item 2: 1.00 (value: 100.00)
  Item 3: 0.67 (value: 80.00)

0-1 Knapsack Greedy Solution:
Maximum value: 160
Selected items (indices): [0, 2]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```



```python
def knapsack_01_greedy(weights, values, capacity):
            selected_items.append(index)

    return max_value, selected_items

def print_solution(weights, values, capacity, solution_type="fractional"):
    # Helper function to print the knapsack solution.

    if solution_type == "fractional":
        max_value, selected_items = knapsack_greedy(weights, values, capacity)
        print(f"Fractional Knapsack Solution:")
        print(f"Maximum value: {max_value:.2f}")
        print("Selected items (index, fraction):")
        for item_index, fraction in selected_items:
            print(f"  Item {item_index}: {fraction:.2f} (value: {values[item_index] * fraction:.2f})")
    else:
        max_value, selected_items = knapsack_01_greedy(weights, values, capacity)
        print(f"0-1 Knapsack Greedy Solution:")
        print(f"Maximum value: {max_value}")
        print("Selected items (indices):", selected_items)

# Example usage
if __name__ == "__main__":
    # Example data
    weights = [10, 40, 20, 30]
    values = [60, 40, 100, 120]
    capacity = 50

    print("Items:")
    for i in range(len(weights)):
        ratio = values[i] / weights[i]
        print(f"Item {i}: weight={weights[i]}, value={values[i]}, ratio={ratio:.2f}")

    print(f"\nKnapsack capacity: {capacity}")
    print("-" * 40)

    # Solve fractional knapsack
    print_solution(weights, values, capacity, "fractional")
    print()

    # Solve 0-1 knapsack (greedy approximation)
    print_solution(weights, values, capacity, "01")
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> &
 C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d
:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Greedy
Algorithms/KnapsackWGreedy.py"
Items:
Item 0: weight=10, value=60, ratio=6.00
Item 1: weight=40, value=40, ratio=1.00
Item 2: weight=20, value=100, ratio=5.00
Item 3: weight=30, value=120, ratio=4.00

Knapsack capacity: 50
----------------------------------------
Fractional Knapsack Solution:
Maximum value: 240.00
Selected items (index, fraction):
  Item 0: 1.00 (value: 60.00)
  Item 2: 1.00 (value: 100.00)
  Item 3: 0.67 (value: 80.00)

0-1 Knapsack Greedy Solution:
Maximum value: 160
Selected items (indices): [0, 2]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```



```python
    # Example data
    weights = [10, 40, 20, 30]
    values = [60, 40, 100, 120]
    capacity = 50

    print("Items:")
    for i in range(len(weights)):
        ratio = values[i] / weights[i]
        print(f"Item {i}: weight={weights[i]}, value={values[i]}, ratio={ratio:.2f}")

    print(f"\nKnapsack capacity: {capacity}")
    print("-" * 40)

    # Solve fractional knapsack
    print_solution(weights, values, capacity, "fractional")
    print()

    # Solve 0-1 knapsack (greedy approximation)
    print_solution(weights, values, capacity, "01")
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> &
 C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d
:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Greedy
Algorithms/KnapsackWGreedy.py"
Items:
Item 0: weight=10, value=60, ratio=6.00
Item 1: weight=40, value=40, ratio=1.00
Item 2: weight=20, value=100, ratio=5.00
Item 3: weight=30, value=120, ratio=4.00

Knapsack capacity: 50
----------------------------------------
Fractional Knapsack Solution:
Maximum value: 240.00
Selected items (index, fraction):
  Item 0: 1.00 (value: 60.00)
  Item 2: 1.00 (value: 100.00)
  Item 3: 0.67 (value: 80.00)

0-1 Knapsack Greedy Solution:
Maximum value: 160
Selected items (indices): [0, 2]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

https://colab.research.google.com/drive/1rWqSihblBB6cZ3ZsHH01WmYgTsd_mhJV?usp=sharing

REFERENCES

GeeksforGeeks. (2025, July 23). *0/1 Knapsack problem*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/0-1-knapsack-problem-dp-10/

W3Schools.com. (n.d.). https://www.w3schools.com/dsa/dsa_ref_greedy.php

GeeksforGeeks. (2025, July 25). *Greedy algorithms*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/greedy-algorithms/

Greedy Algorithms | Brilliant Math & Science Wiki. (n.d.). https://brilliant.org/wiki/greedy-algorithm/