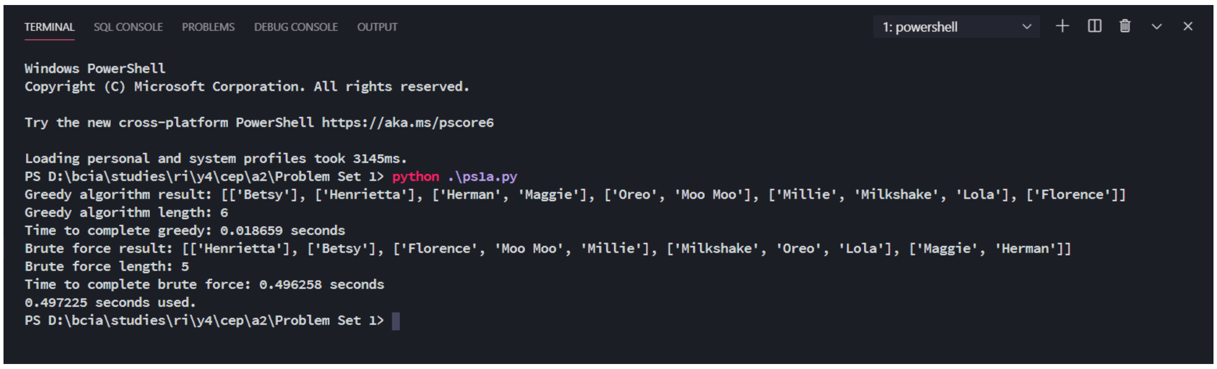
## Problem A

1. Both algorithms were very fast considering only 10 inputs were given. With that in mind, the greedy algorithm was significantly faster than the brute force algorithm as can be seen in the image below. The greedy algorithm took only 0.019s compared to the 0.496 seconds taken by the brute force algorithm, which is 26 times faster. This result is expected due to the different amount of time required to run each algorithm as the number of inputs increases. The time complexity for my greedy algorithm is only $O(n^3)$ where *n* is the number of animals inputted into the function, whereas for the greedy algorithm, it is $O(2^n)$. As $2^n$ increases faster than $n^3$, you can expect the greedy algorithm to be faster than the brute force algorithm.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

Loading personal and system profiles took 3145ms.
PS D:\bcia\studies\ri\y4\cep\a2\Problem Set 1> python .\ps1a.py
Greedy algorithm result: [['Betsy'], ['Henrietta'], ['Herman', 'Maggie'], ['Oreo', 'Moo Moo'], ['Millie', 'Milkshake', 'Lola'], ['Florence']]
Greedy algorithm length: 6
Time to complete greedy: 0.018659 seconds
Brute force result: [['Henrietta'], ['Betsy'], ['Florence', 'Moo Moo', 'Millie'], ['Milkshake', 'Oreo', 'Lola'], ['Maggie', 'Herman']]
Brute force length: 5
Time to complete brute force: 0.496258 seconds
0.497225 seconds used.
PS D:\bcia\studies\ri\y4\cep\a2\Problem Set 1>
```

2. The greedy algorithm does not return the optimal solution as it returned a list of length 6 instead of length 5 returned by the brute force algorithm. The greedy algorithm was too eager to put the largest sheep in first, which may not have optimally filled in each spaceship, which that on average there was a greater amount of free space left, resulting in a larger result.
3. The brute force algorithm gave the optimal solution. It could check all possible solutions, from which it can choose the smallest of the valid solutions.

## Problem B

1. A brute force algorithm has a time complexity of $O(2^n)$, and the same level space complexity if we are to consider the memory used to store the variables for each recursive call to a brute force function. Hence, an input with 30 eggs will take $2^{30-10} = 1{,}048{,}576$ times the amount of time to deal with an input with 10 eggs and certainly crash my laptop with limited RAM.
2. The function will try to shave as much of the value of the amount available capacity left using the heaviest egg that can fit in the remaining capacity until there is no capacity left. If there is some leftover capacity and the lightest egg is unable to fill it, the function can backtrack each shave and use another egg of a smaller weight to shave off the remaining capacity at that iteration.

3. It will always return the optimal solution as it will consider all possible solutions to the problem.