Question 1

A (סעיף א)

1.1

The script prints to the console the following:

```
<class 'tuple'>
False
[True, True, True]
[99, 3, 4]
```

1.2

The assignment a[0] = 99 causes an error because the tuple type does not support item Assignment . meaning, you can assign a to an entirely new tuple (e.g: a = (99)) but not a specific item in the tuple.

2.1

The function returns a variable of type tuple and its' value is (1,10)

B (סעיף ב)

1.1

The set type does not support any direct way of altering the order of items. Unless redefining the set, the order will stay the same.

1.2

In a set, there are no duplicates (as proven by running s.add(3) twice one time after another).

2.1

- **clear():** removes all the elements from a list
- **discard(item):** removes a specific item from a set

C (סעיף ג)

1.1

When attempting to access a non-existent key, an error occurs.
There's no direct way to access a key based on a value without loops
A for loop on a dictionary iterates over the keys.

2.

Implemented in the python file under Q1.2

Question 2

a. *Implemented in python file*
b. *Implemented in python file*
c. For monty_hall(False, 10000) the output of the function is: 0.3292. For monty_hall(True, 10000) the output of the function is: 0.6653.
Based on these results, it's same to assume that switching is the preferable strategy. If a player decides to switch doors we're looking at a completely different statistical situation. If the player doesn't switch there's about a 33% change of winning as expected. If he chooses to switch however, if he does switch however. In the case that we can say

first_choice != car_door. Because he won't choose the rock door, and so now the probability is about 66%.

d. Implemented in python file

e. Number of times each anagram was returned:
{'bca': 1681, 'cab': 1625, 'cba': 1694, 'abc': 1620, 'bac': 1683, 'acb': 1697}
Based on the large scale results it looks like all the possible anagrams were returned and with a reasonably equal distribution.

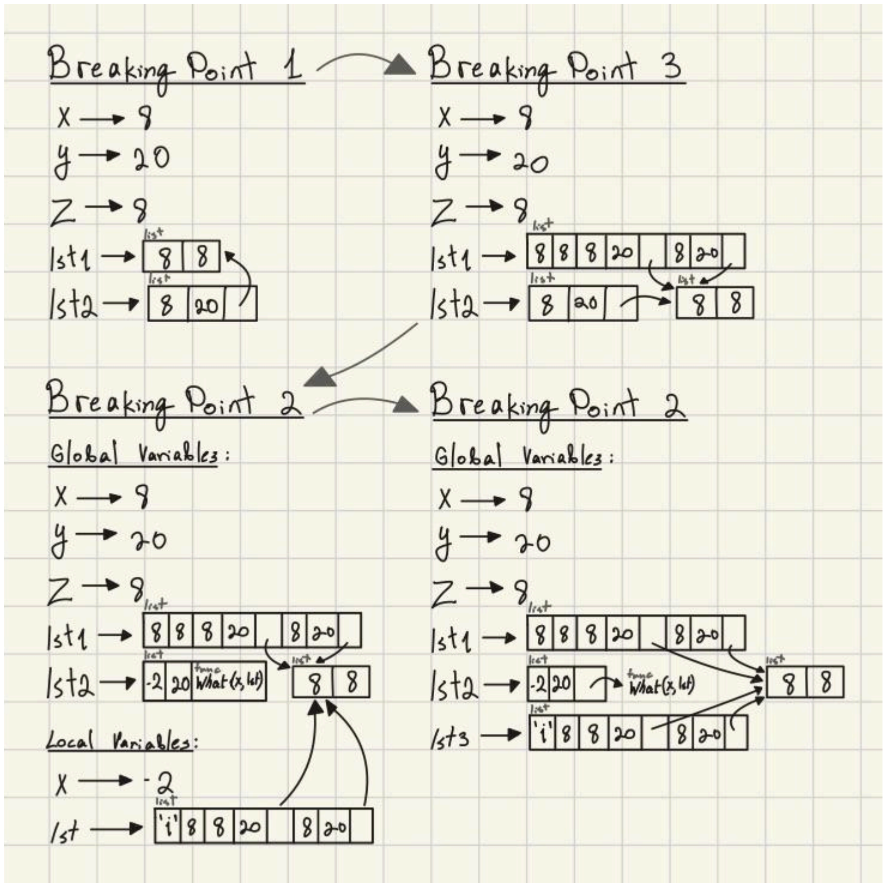
Question 3

a-d. Implemented in python file

e.

1. 2^42. The number 2^10 takes up 10 of the 52 bits meant to represent the fraction so 42 are left.
2. Since we established that 2^52 is the highest power possible to represent accurately with ieee 754 64-bit. Meaning, if we go any larger, like w^53 we start running into issues of rounding down. Like we see in the second example of (2^53)+1 that changes the last bit whereas when we add 2 it affects the second to last bit which still retains accuracy

Question 5



Breaking Point 1
x → 8
y → 20
z → 8
lst1 → 8 8
lst2 → 8 20
Breaking Point 3
x → 8
y → 20
z → 8
lst1 → 8 8 8 20 8 20
lst2 → 8 20 → 8 8
Breaking Point 2
Global Variables:
x → 8
y → 20
z → 8
lst1 → 8 8 8 20 8 20
lst2 → -2 20 What(x, lst) 8 8
Local Variables:
x → -2
lst → 'i' 8 8 20 8 20
Breaking Point 2
Global Variables:
x → 8
y → 20
z → 8
lst1 → 8 8 8 20 8 20
lst2 → -2 20 What(x, lst) 8 8
lst3 → 'i' 8 8 20 8 20