Part of a rearrangement for Fall 2023 semester.

Name:______________________________ **Partner:** __________________________

Python Activity 21b: List Aliasing

Learning Objectives
Students will be able to:
Content:
- Define what ***aliasing*** is
- Predict how modifying a list will change the values of its ***aliases***
- Explain why creating ***aliases*** is not the same as creating ***copies*** of objects

Process:
- Write code that creates *aliases* of mutable objects
- Write code that creates *copies* of mutable objects

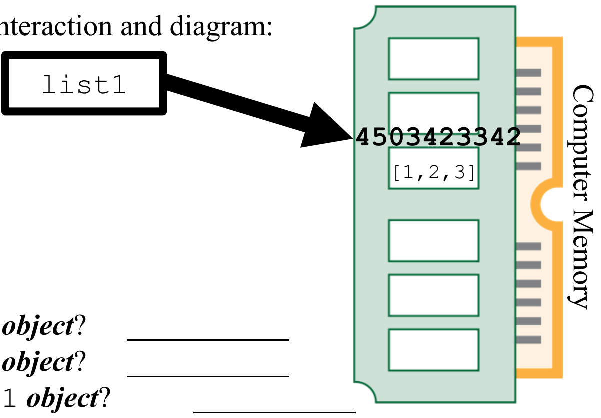
Prior Knowledge
- Python concepts: identity vs. value, mutability, lists, strings, boolean operators

Critical Thinking Questions:

1. Examine the sample interactive python interaction and diagram:

Interactive Python

```
>>> list1 = [1, 2, 3]
>>> id(list1)
4503423342
>>> list2 = list1
>>> list1 is list2
True
```



a. What is the ***value*** of the `list1` ***object***? ____________
   What is the ***value*** of the `list2` ***object***? ____________

b. What is the ***identity*** of the `list1` ***object***? ____________
   What is the ***identity*** of the `list2` ***object***? ____________

c. Draw `list2` in the diagram above with the arrow pointing to memory and its value.

2. Examine the following interaction, which continues from the previous example:

Continued

```
>>> list1 += [4]
```

a. Modify the diagram in Question 1 to reflect the change in this new code.

b. According to the diagram:
   Did `list1`'s ***identity*** change? ____ Did `list1`'s ***value*** change? ____
   Did `list2`'s ***identity*** change? ____ Did `list2`'s ***value*** change? ____

c. What is now stored at the `4503423342` ***memory address***? ____________

d. If we executed the following line, what would be stored at `list1`?
   `list2 += [["hi", "bye"]]`

   ____

FYI: If multiple variable names are pointing to the same place in memory, this is known as ***aliasing***. If that object is ***mutable*** then any changes to the value through one variable name, will impact all variables pointing to that same address! This **does not** make a copy of the mutable object!

3. Observe the following interaction in interactive python:

```
>>> list1 = [1, 2, 3]
>>> list2 = list1
>>> my_lst = [1, 2, 3]
>>> my_lst == list1 == list2
True
>>> my_lst is list1
```

a. Why does the `my_lst == list1 == list2` line return its boolean value?

____

b. What might the `list2 = list1` line do?

____

How might this affect the ***memory address*** of `list2`?

____

c. What does the `my_lst = [1, 2, 3]` line do?

____

How might that line affect the ***memory address*** of `my_lst`?

____

d. What might be the output of `my_lst is list1`? __________

4. Observe the following interactions in interactive python:

```
>>> list1 = [1, 2, 3]
>>> list3 = list1[:]
>>> list3
[1, 2, 3]
>>> list3 is list1
False
```

```
>>> list4 = [ele for ele in list1]
>>> list4
[1, 2, 3]
>>> list4 is list1
False
```

a. Does `list1 == list3 == list4`? ________________

b. Do `list1` and `list3` point to the same memory address? What about `list4`?

____

c. After all this code is executed, if we entered `list1+=[4]`, what would be the value of `list3`? `list4`? ________________________

____

d. What does the `list3 = list1[:]` line do?

____

What does the `list4 = [ele for ele in list1]` line do?

____

Application Questions: Use the Python Interpreter to check your work

1. Observe the following interaction in interactive python:

```
>>> nums = [23, 19]
>>> words = ["hello", "world"]
>>> mixed = [12, nums, "nice", words]
```

a. Draw the three lists in a diagram, pointing to their places in memory:

b. If we executed the line `print(mixed)`, what would be displayed?

____

c. If we executed the line `words.append("sky")`, what is stored at `mixed`?

____

d. If we executed the line `mixed[1].append(27)`: What is stored at `nums`?

____

What is stored at `mixed`? ______________________________