Name:_____ **Partner:** _____

Python Activity 15: Lists of Lists

Lots of data requires a table or a matrix…

Learning Objectives

Students will be able to:

Content:

- Define a **nested list** or **list of lists**
- Identify **empty lists** and **empty strings**

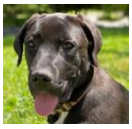
Process:

- Write code to construct and add elements to lists of lists
- Write code to access elements at a given index
- Write code to iterate over lists of lists

Prior Knowledge

- Python concepts: lists, types, len(), string literals, nested loops

Critical Thinking Questions:


1. Examine the sample code defining a list below.

Sample Code

```
dog2owner =
[["pixel","iris"],["chels","lida"],["artie","bill"]]
```

a. What element is at `dog2owner[0]`? _____

What is this element's data type (circle one): string list int bool

b. Within `dog2owner[0]`, what is stored at index **1**? _____

What is this element's data type (circle one): string list int bool

c. We can access this same string value using only list indexing with `dog2owner[0][1]`.

How might we access the name of Iris' dog using list indexing? _____

d. What element is in `dog2owner[2]`? _____

Within `dog2owner[2]`, what is stored at index 0? _____

How would we write this with list indexing? _____

e. Write a line of code to access and print the name of Lida's dog via list indexing:

_____

f. When working with nested lists (such as in the Sample Code above), what does the *first* list index refer to? _____

What does the *second* list index refer to? _____

2. Examine the two lists below:

```
alst = [ ["cat", "frog"],
         ["puma", "toad"],
         ["lion", "newt"] ]
```

```
blst = ["cat", "frog",
        "puma", "toad",
        "lion", "newt"]
```

a. What element is stored at **a**`lst[1][0]`? _____

What is this element's data type (circle one): string list int bool

b. What element is stored at **b**`lst[1][0]`? _____

What is this element's data type (circle one): string list int bool

c. What kind of list is **a**`lst`? A list of _____

What kind of list is **b**`lst`? A list of _____

How do you know?

_____

FYI: ***Lists*** are a sequence of elements and these elements can be of *any* data type, including more lists! While python does not require us to specify a variable's data type, we often assume the list has elements of a particular type. ***Lists of lists*** and ***lists of strings*** are easy to mix up as both lists and strings are sequences!

3. Examine the sample code below, it has a ***logic error***:

```
pet2age = [ ["pixel", "dog", 4], ["dizzy", "cat", 10] ]
pet2age = pet2age + ["moone", "demon", 2]
print(pet2age)
```

And its output:

```
[['pixel', 'dog', 4], ['dizzy', 'cat', 10], 'moone', 'demon', 2]
```

a. What kind of object is `pet2age` (circle one): string list int bool

b. What kind of objects are stored in `pet2age`: string list int bool

c. What kind of object did the programmer *try* to add in the second line of code? _____

What kind of object did the programmer *actually* add? _____

d. What line of code should the programmer have written to ensure the new element added was of the same type as the rest of the elements in `pet2age`?

_____

4. Examine the sample code below:

```
pet2age = [ ["pixel", 4], ["dizzy", 10], ["moone", 1] ]

cats_first = [ pet2age[-1], pet2age[1], pet2age[0]]
```

a. In the first line of code, what is stored in `pet2age[-1]`:

_____

In the first line of code, what is stored in `pet2age[1]`:

_____

In the first line of code, what is stored in `pet2age[0]`:

_____

b. What might this sample code do?

_____

c. We can achieve a similar output in `cats_first` by *reversing* our list of lists.

How might we do that? _____

5. Examine the three interactive python sessions below:

```
>>> new_str = ""
>>> len(new_str)
0
```

```
>>> new_lst = []
>>> len(new_lst)
0
```

```
>>> lst_lst = [[]]
>>> len(lst_lst)
1
```

a. Why might `len(new_str)` return 0? _____

Why might `len(new_lst)` return 0? _____

Why might `len(lst_lst)` return 1? _____

b. Which of the above variables would we describe as an ***empty list***? _____

Which of the above variables would we describe as an ***empty string***? _____

Which of the above variables is not empty? _____

c. What might the code `len(" ")` return ? _____

Why?

_____

d. What might the code `len([""])` return ? _____

Why?

_____

6. Observe the following code:

Python Program

```
def question2():
    lst_lsts = [[1,2,3],[4,5,6],[7,8,9]]
    new_lstlsts = []
    for row in lst_lsts:
        new_row = []
        for item in row:
            new_row = new_row + [item*item]
        new_lstlsts = new_lstlsts + [new_row]
    return new_lstlsts
```

Can be helpful to think of it like this:

```
lst_lsts = [[1,2,3],
            [4,5,6],
```

a. Examine the code above. What is the output of this program? Trace through the values as they change:

	new_lstlsts	row	new_row	item
Before the outerloop:	_____			
Outer Iteration 1:	_____	_____	_____	
Inner Iteration 1:	_____	_____	_____	_____
Inner Iteration 2:	_____	_____	_____	_____
Inner Iteration 3:	_____	_____	_____	_____
Outer Iteration 2:	_____	_____	_____	
Inner Iteration 1:	_____	_____	_____	_____
Inner Iteration 2:	_____	_____	_____	_____
Inner Iteration 3:	_____	_____	_____	_____
Outer Iteration 3:	_____	_____	_____	
Inner Iteration 1:	_____	_____	_____	_____
Inner Iteration 2:	_____	_____	_____	_____
Inner Iteration 3:	_____	_____	_____	_____
Final:	_____			

b. What does this code do?

_____

_____

c. There are two accumulator variables in this code, what are there names?

_____ and _____

Why do we need two accumulator variables?

_____

_____

d. What might happen if `[item * item]` did not have square brackets around it in the code above? _____

_____

What might happen if `[new_row]` did not have square brackets around it in the code above?_____

_____

Application Questions: Use the Python Interpreter to check your work

1. Write a function, `switcheroo`, that take a list of lists, `lol`, as a parameter, and returns a new list of lists that has swapped the first and last items of each element of the list of lists.

_____

_____

_____

_____

_____

_____

_____

_____