CS PROFESSIONAL ELECTIVE | FINAL ASSIGNMENT #2

MATILLA, CARL ANDRIE D. | BSCS 3C

DEFINING A LIST

A list in Python is a collection of things contained by square brackets **"[ ]"**. It may hold items of many data types, such as integers, floats, texts, and even other lists.

Example:

```
my_list = [1, 2, 3, 4, 5]
```

LIST SYNTAX

In Python, the syntax for defining a list is to wrap the list's items with square brackets **"[ ]"**. Elements in the list are separated by commas. Here's some basic syntax:

Example:

```
my_list = [element 1, element 2, element 3, …]
```

Each entry can be any data type, including integers, floats, strings, booleans, and even other lists.

Example:

```
Int: numbers = [1, 2, 3, 4, 5]
strings: fruits = ['apple', 'banana', 'orange', 'kiwi']
mixed: mixed_list = [1, 'hello', 3.14, True]
```

Lists are changeable, which means you may modify their elements after they have been formed. Lists in Python also offer indexing and slicing operations, which allow you to access specific elements or sublists within the list.

ACCESSING LIST ELEMENTS

Indexing allows you to access the elements of a list. Indexing begins with 0 for the first element, -1 for the last element, -2 for the second last, and so on.

Example:

```
print(my_list[0])  # Output: 1
print(my_list[-1]) # Output: 5
```

LOOP THROUGH A LIST
In Python, the **'for'** loop is widely used to iterate through a list. Here's how you can iterate over each item in a list:

Example:

```
my_list = [1, 2, 3, 4, 5]

for item in my_list:
   print(item)
```

In this example, the 'for' loop outputs each element ('item') from the'my_list'. The loop repeats until all entries in the list have been iterated over.

LIST LENGTH
The **'len()'** method in Python returns the length of a list, or the number of entries it contains. Here's how you can accomplish it:

Example:

```
my_list = [1, 2, 3, 4, 5]
length = len(my_list)

print("Length of the list:", length) Output: Length of the list: 5
```

The len() function returns the number of entries in the list, in this example '5'. It's a useful function to have when you need to know how many elements are in a list, especially when dealing with loops or conditions that depend on the length of the list.

ADD ITEMS TO THE LIST
You may add items to a list using methods like **'append()', 'insert()', and 'extend()'.**

Example:

```
my_list.append(6)    # Adds 6 at the end
my_list.insert(0, 0) # Adds 0 at the beginning
```

REMOVE ITEM TO THE LIST

Items in a list can be deleted using methods such as **'remove()', 'pop()', or 'del'.**

Example:

```
my_list.remove(3)    # Removes the first occurrence of 3
my_list.pop(0)       # Removes the item at index 0
del my_list[1:3]     # Removes items from index 1 to 2
```

THE LIST () CONSTRUCTOR

The list() constructor allows you to create a list.

Example:

```
new_list = list((1, 2, 3))  # Creates a new list from a tuple
```

LIST METHODS

In Python, lists include a number of built-in functions that allow you to modify and interact with them effectively. Here are some typical list methods:

append(): Inserts an element at the end of the list.

```
my_list = [1, 2, 3]
my_list.append(4)
print(my_list)  # Output: [1, 2, 3, 4]
```

insert(): Inserts an element at a certain place in the list.

```
my_list = [1, 2, 3]
my_list.insert(1, 5)  # Insert 5 at index 1
print(my_list)  # Output: [1, 5, 2, 3]
```

extend(): Appends entries from another list to the end of the current one.

```
my_list = [1, 2, 3]
other_list = [4, 5]
my_list.extend(other_list)
print(my_list)  # Output: [1, 2, 3, 4, 5]
```

Remove(): removes the first occurrence of a specified element from the list.

```
my_list = [1, 2, 3, 2]
my_list.remove(2)  # Remove the first occurrence of 2
print(my_list)  # Output: [1, 3, 2]
```

pop(): removes the element at the provided location (or the final element if no position is specified) and returns it.

```
my_list = [1, 2, 3]
popped_element = my_list.pop(1)  # Remove and return the element at index 1
print(popped_element)  # Output: 2
print(my_list)  # Output: [1, 3]
```

index(): returns the index of the first occurrence of a given element in the list.

```
my_list = [1, 2, 3, 2]
index = my_list.index(2)  # Get the index of the first occurrence of 2
print(index)  # Output: 1
```

count(): returns the number of times a specified element appears in the list.

```
my_list = [1, 2, 3, 2]
count = my_list.count(2)  # Count the occurrences of 2
print(count)  # Output: 2
```

sort(): returns the list in ascending order.

```
my_list = [3, 1, 2]
my_list.sort()
print(my_list)  # Output: [1, 2, 3]
```

reverse(): Returns the list's entries in reverse order.

```
my_list = [1, 2, 3]
my_list.reverse()
print(my_list)  # Output: [3, 2, 1]
```

NESTED LISTS

In Python, a nested list is one that contains entries from other lists. This lets you build a multidimensional structure, with each inner list representing a row or column of data. This is an example of a nested list.

Example:

```
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```
