Salvadora, Angelle D.
CS3C

Lists

▪ **Defining a List**

— used to store multiple items in a single variable. It has 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

▪ **List Syntax**

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

▪ **Accessing List Elements**

— In Python, you can define a list using square brackets [] to enclose comma-separated values. Lists can contain elements of different types, including integers, floats, strings, and even other lists.

Ex.

```
my_list = [ ]
my_list = [value1, value2, value3, ...]
                    or
my_list = [1, 2, 3, 4, 5]
```

▪ **Loop through a List**

— We can use various methods such as using a **for loop**, a list comprehension, or built-in functions like **map()** and **filter()**.

Ex.

```
my_list = [10  20  30  40  50]

# Using a for loop
for item in my_list:
print(item)
```

▪ **List Length**

— The **len()** function takes a list as its argument and returns the number of elements in that list.

Ex.

```
my_list = [10  20  30  40  50]

# Finding the length of the list
length = len
print(length)       #Output: 5
```

▪ **Add Items in the List**

— **Appending Elements:** You can add elements to the end of a list using the append() method.

Ex.

```
my_list.append(6)
print(my_list)     #Output: [1, 2, 3, 4, 5, 6]
```

— **Inserting Elements:** You can insert elements at a specific position in the list using the insert() method.

Ex.

```
my_list.insert(2  "inserted")

print(my_list)       #Output: [1, 2, 'inserted', 3, 4, 5, 6]
```

▪ **Remove Item from a List**

— **Removing Elements:** You can remove elements from a list using the remove() method or by using the del statement.

Ex.

```
my_list.remove(3)
print(my_list)      #Output: [1, 2, 4, 5, 6]
```

▪ **The List () Constructor**

— is a built-in function that allows you to create a new list from an iterable object, such as another list, a tuple, a string, or any other iterable. It can also create an empty list if no argument is provided.

Ex.

```
new_list = list(iterable)
```

▪ **List Methods**

1. **append():** Adds a single element to the end of the list.
2. **extend():** Adds multiple elements (from another iterable) to the end of the list.
3. **insert():** Inserts an element at a specified position in the list.
4. **remove():** Removes the first occurrence of a specified value from the list.
5. **pop():** Removes and returns the element at a specified index (default is the last element).
6. **index():** Returns the index of the first occurrence of a specified value.
7. **count():** Returns the number of occurrences of a specified value in the list.
8. **sort():** Sorts the elements of the list in ascending order (in-place).
9. **reverse():** Reverses the order of the elements in the list (in-place).
10. **copy():** Returns a shallow copy of the list.

▪ **Nested Lists**

— is a list that contains other lists as its elements. This allows for the creation of multidimensional data structures, where each inner list can represent a row or column of data. Nested lists can be used to represent matrices, tables, or any hierarchical data structure.

Ex.

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