**AI Day 09 Notes**

**Syed Mansoor ul Hassan Bukhari**

**Python Lists:**

1. **Definition**: Lists in Python are used to store multiple items in a single variable. They are ordered, changeable, and allow duplicate values. Lists are written with square brackets.

thislist = ["apple", "banana", "cherry"]

print(thislist)

1. **List Items**: List items are ordered, changeable, and allow duplicate values. They are indexed, with the first item having an index of 0.

thislist = ["apple", "banana", "cherry"]

print(thislist[1]) # Output: banana

1. **Order**: Lists are ordered, meaning the items have a defined order that will not change. You can access items by referring to their index.

thislist = ["apple", "banana", "cherry"]

print(thislist[0]) # Output: apple

1. **Changeability**: Lists are mutable, meaning you can change, add, or remove items after the list has been created.

thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist) # Output: ['apple', 'blackcurrant', 'cherry']

1. **Duplication**: Lists can have items with the same value.

thislist = ["apple", "banana", "cherry", "apple"]

print(thislist) # Output: ['apple', 'banana', 'cherry', 'apple']

1. **List Length**: The len() function is used to determine how many items a list has.

thislist = ["apple", "banana", "cherry"]

print(len(thislist)) # Output: 3

1. **Data Types**: The values in list items can be of any data type.

thislist = ["apple", 1, True, 3.14]

print(thislist)

1. **The list() Constructor**: It is also possible to use the list() constructor to make a list.

thislist = list(("apple", "banana", "cherry"))

print(thislist)

1. **Accessing Items**: Items of a list can be accessed by referring to its index number, inside square brackets.

thislist = ["apple", "banana", "cherry"]

print(thislist[1]) # Output: banana

1. **Checking if Item Exists**: To determine if a specified item is present in a list, use the in keyword.

thislist = ["apple", "banana", "cherry"]

if "banana" in thislist:

print("Yes, 'banana' is in the list")

1. **Updating List**: You can change the value of a specific item by referring to its index number.

thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist) # Output: ['apple', 'blackcurrant', 'cherry']

1. **Adding Items**: Adding an item to the list is done using the append() method.

thislist = ["apple", "banana", "cherry"]

thislist.append("orange")

print(thislist) # Output: ['apple', 'banana', 'cherry', 'orange']

1. **Removing Items**: The remove() method removes the specified item.

thislist = ["apple", "banana", "cherry"]

thislist.remove("banana")

print(thislist) # Output: ['apple', 'cherry']

1. **Looping Through a List**: You can loop through a list by using a for loop.

thislist = ["apple", "banana", "cherry"]

for x in thislist:

print(x)

1. **Nested Lists**: A list can contain other lists, this is called nested lists.

mylist = [["apple", "banana"], ["cherry", "date"]]

print(mylist)

1. **Clearing List**: The clear() method empties the list.

thislist = ["apple", "banana", "cherry"]

thislist.clear()

print(thislist) # Output: []

1. **Deleting List**: The del keyword removes the list completely.

thislist = ["apple", "banana", "cherry"]

del thislist

**Python Sets:**

1. **Definition**: Sets in Python are used to store multiple items in a single variable. They are unordered, unchangeable (but you can add or remove items), and do not allow duplicate values. Sets are written with curly brackets.

thisset = {"apple", "banana", "cherry"}

print(thisset)

1. **Set Items**: Set items are unordered, unchangeable, and do not allow duplicate values. They are not indexed.

thisset = {"apple", "banana", "cherry"}

for x in thisset:

print(x)

1. **Order**: Sets are unordered, meaning the items do not have a defined order and you cannot refer to an item by using an index.

thisset = {"apple", "banana", "cherry"}

print(thisset)

1. **Changeability**: Sets are unchangeable, meaning you cannot change the items after the set has been created, but you can add or remove items.

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset) # Output: {'apple', 'banana', 'cherry', 'orange'}

1. **Duplication**: Sets cannot have two items with the same value.

thisset = {"apple", "banana", "cherry", "apple"}

print(thisset) # Output: {'apple', 'banana', 'cherry'}

1. **Set Length**: The len() function is used to determine how many items a set has.

thisset = {"apple", "banana", "cherry"}

print(len(thisset)) # Output: 3

1. **Data Types**: The values in set items can be of any data type.

thisset = {"apple", 1, True, 3.14}

print(thisset)

1. **The set() Constructor**: It is also possible to use the set() constructor to make a set.

thisset = set(("apple", "banana", "cherry"))

print(thisset)

1. **Accessing Items**: You cannot access items in a set by referring to an index or a key, but you can loop through the set items using a for loop.

thisset = {"apple", "banana", "cherry"}

for x in thisset:

print(x)

1. **Checking if Item Exists**: To determine if a specified item is present in a set, use the in keyword.

thisset = {"apple", "banana", "cherry"}

if "banana" in thisset:

print("Yes, 'banana' is in the set")

1. **Updating Set**: You cannot change the items in a set, but you can add new items.

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset) # Output: {'apple', 'banana', 'cherry', 'orange'}

1. **Adding Items**: Adding an item to the set is done using the add() method.

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset) # Output: {'apple', 'banana', 'cherry', 'orange'}

1. **Removing Items**: The remove() method removes the specified item.

thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset) # Output: {'apple', 'cherry'}

1. **Looping Through a Set**: You can loop through a set by using a for loop.

thisset = {"apple", "banana", "cherry"}

for x in thisset:

print(x)

1. **Nested Sets**: Sets cannot contain other sets, but they can contain other iterable objects like tuples.

thisset = {"apple", "banana", ("cherry", "date")}

print(thisset)

1. **Clearing Set**: The clear() method empties the set.

thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset) # Output: set()

1. **Deleting Set**: The del keyword removes the set completely.

thisset = {"apple", "banana", "cherry"}

del thisset