Recap

Yesterday

Python Keywords

Python Comments

Python Input

Python Conditional Statements

Logical Operators

Pass Statement

Today

Python Lists

Python Tuples

Python Sets

Python Dictionaries

Python Data Collections

Python Lists

Lists are used to store multiple items in a single variable.

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

List items are **ordered**, **changeable**, and **allow duplicate** values.

List items are **indexed**

Python Lists

ORDERED

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

Python Lists

Changeable

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

Python Lists

Add Item to List

append()

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

Python Lists

Insert Item to List

`insert()`

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

Python Lists

Extend List

`extend()`

To append elements from *another list* to the current list

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

Python Lists

Remove item from List

`remove()`

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

Python Lists

Remove item from List

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

?
```

Python Lists

Remove Specified Index

```
pop()
```

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

Python Lists

Remove Specified Index

```
pop()
```

```
thislist = ["apple", "banana", "cherry"]
thislist.pop() #without any index ?
print(thislist)
```

?

Python Lists

Remove item from list

```
del
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

Python Lists

Delete the list

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

Python Lists

Length of a list

Len()

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

Python Lists Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current lis
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python Tuples

Tuples are used to store multiple items in a single variable.

A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with round brackets.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Python Tuples with one item

To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.

```
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple”)
print(type(thistuple))
```

Python Tuples

```
tuple1 = ("apple", "banana", "cherry")

tuple2 = (1, 5, 7, 9, 3)

tuple3 = (True, False, False)

tuple4 = ("abc", 34, True, 40, "male")
```

Python Tuples

Can we change a Tuple Value?

Python Tuples – How to change?

Convert the tuple into a list to be able to change it

```
x = ("apple", "banana", "cherry")

y = list(x)

y[1] = "kiwi“

x = tuple(y)

print(x)
```

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Python Tuples Methods

Python Sets

Sets are used to store multiple items in a single variable.

A set is a collection which is ***unordered***, ***unchangeable****, and ***unindexed***.

Sets are written with curly brackets.

```
{ }
```

Python Sets

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

print(thisset)
```

Python Sets

Set items are unordered, unchangeable, and do not allow duplicate values.

Once a set is created, you cannot change its items, but you can remove items and add new items.

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

print(thisset)
```

Python Sets

What will be output?

```
thisset = {"apple", "banana", "cherry", True, 1, 2}

print(thisset)
```

Python Sets Add and Remove Items

add() , remove()

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

thisset.add("orange")

print(thisset)

thisset.remove("banana")

print(thisset)
```

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Python Sets

Python Dictionaries

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered*, changeable and do not allow duplicates.

Dictionaries are written with curly brackets, and have keys and values

```
thisdict = {
  “name": “Muhammed Aziz",
  “designation": “Instructor",
  "year": 2024
}
print(thisdict)
```

Python Dictionaries

Access

```
thisdict = {
  “name": “Muhammed Aziz",
  “designation": “Instructor",
  "year": 2024
}
print(thisdict[“name”])
```

Python Dictionaries

As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

Changeable

```
thisdict[“name”] = “New Name”
```

Duplicates Not Allowed

```
thisdict = {
  “name": “Muhammed Aziz",
  “designation": “Instructor",
  "year": 2024,
  "year": 2024
}
print(thisdict[“name”])
```

Python Dictionaries

As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

```
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}
```

Python Dictionaries

Add Item

```
thisdict = {
  “name": “Muhammed Aziz",
  “designation": “Instructor",
  "year": 2024
}
```

```
thisdict[“organization"] = “skill city"
print(thisdict)
```

Python Dictionaries

Remove Item

pop()

```
thisdict = {
  “name": “Muhammed Aziz",
  “designation": “Instructor",
  "year": 2024
}


thisdict.pop(“year")
print(thisdict)
```

Python Dictionaries

Remove Item

popitem()

```
thisdict = {
  “name": “Muhammed Aziz",
  “designation": “Instructor",
  "year": 2024
}


thisdict.popitem()
print(thisdict)
```

Python Nested Dictionaries

A dictionary can contain dictionaries, this is called nested dictionaries.

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```

clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs

Python Dictionaries

Python Data Collections

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- **Dictionary** is a collection which is ordered** and changeable. No duplicate members.

Activity

Write a program that asks for a data collection type from user and creates that data collection with minimum 3 data items and print it.