SET AS DATA TYPE

- Sets are used to store multiple items in a single variable.
- Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.
- A set is a collection which is both *unordered* and *unindexed*.
- Sets are written with curly brackets.

How To create a set as data type

- thisset = {"apple", "banana", "cherry"}
  print(thisset)

Set Items

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

Unordered

Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

Unchangeable

Sets are unchangeable, meaning that we cannot change the items after the set has been created.

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

Duplicates Not Allowed

Sets cannot have two items with the same value.

Example

```
thisset = {"apple", "banana", "cherry", "apple"}
print(thisset)
```

Output

```
{'banana', 'cherry', 'apple'}
```

LENGTH OF A SET

To determine how many items a set has, use the len() method.

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

print(len(thisset))
```

Output

3

- Set items can be of any data type:

Example

```
set1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}
print(set1)
print(set2)
print(set3)
```

output

{'cherry', 'apple', 'banana'}

{1, 3, 5, 7, 9}

{False, True}

A SET WITH DIFFERENT DATA TYPE

- A set with strings, integers and boolean values:
- set1 = {"abc", 34, True, 40, "male"}

```
print(set1)
```

output

{True, 34, 40, 'male', 'abc'}

LIST AS DATA TYPE

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

HOW TO CREATE A LIST

Lists are created using square brackets:

Example

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

Output

```
['apple',
'banana', 'cherry']
```

LIST

List Items

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

List items are indexed, the first item has index [0], the second item has index [1] etc.

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.

Note: There are some list methods that will change the order, but in general: the order of the items will not change.

Changeable

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

Allow Duplicates

Since lists are indexed, lists can have items with the same value:

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

Output

```
['apple', 'banana', 'cherry', 'apple', 'cherry']
```

Print the number of items in the list:

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

output

```
3
```

List with different data types.

```
list1 = ["abc", 34, True, 40, "male"]
print(list1)
```

output

```
['abc', 34, True, 40, 'male']
```

ACCESING LIST ELEMENTS

Accessing list elements in a list is the same way as that of tuble but the only difference is List uses square bracket [] whereas tuples uses round bracket.

All the methods of indexing are the same.

Refer back to tubles slides and try to access the list elements by using all the methods defined there.

DICTIONARY AS DATA TYPE

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered, changeable and does not allow duplicates.
- Dictionaries are written with curly brackets, and have keys and values.

How To create a Dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

output

{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

DICTIONARY ITEMS

Example:- The program below print brand value of the dictiona

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"])
```

output

Ford

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

DICTIONARY

Ordered or Unordered?

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

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

Unordered means that the items does not have a defined order, you cannot refer to an item by using an index.

Changeable

Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.

Duplicates Not Allowed

Dictionaries cannot have two items with the same key:

DICTIONARY OVERWRITTEN

Example:- The program below have duplicates values of years therefore the program prints all the element in the dictionary but year 2020 overwrite the year 1964.

Note:-Duplicate values are not allow in dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(thisdict)
```

Output

{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}

DICTIONARY LENGHT

To determine how many items a dictionary has, use the len() function

Example:- The program below print the length of the dictionary.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(len(thisdict))
```

Output

3

DICTIONARY DATA TYPES

Dictionary can contained different data types, as seen in the below program with string, Boolean, integer and list as the data type.

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

output

{'brand': 'Ford', 'electric': False, 'year': 1964, 'colors': ['red', 'white', 'blue']}

Example:-The program below print the data types of the dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(type(thisdict))
```

Output

<class 'dict'>

Accessing Dictionary Items

You can access the items of a dictionary by referring to its key name, inside square brackets.

Example:- The program below get the value of model key

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]
```

Output

Mustang

We can also use the get method to print the value of the model key

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict.get("model")
print(x)
```

output

Mustang

The program below print all the values of the dictionary.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964}
x = thisdict.values()
print(x)
```

Output

```
dict_values(['Ford', 'Mustang', 1964])
```

ADDING AND ITEM IN THE DICTIONARY

Adding a new item to the original dictionary, and see that the keys list gets updated as well.

Example:- The program below add a new element “colour” of the car which is white in colour.

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

x = car.keys()

print(x) #before the change

car["color"] = "white"

print(x) #after the change
```

Output

```
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year',
'color'])
```

CHANGE IN THE DICTIONATY

Example:-The program below change the value of the year key to 2020.

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

x = car.values()

print(x) #before the change

car["year"] = 2020

print(x) #after the change
```

Output

```
dict_values(['Ford', 'Mustang', 1964])
dict_values(['Ford', 'Mustang', 2020])
```

GET ITEMS IN DICTIONARY

The items() method will return each item in a dictionary, as tuples in a list.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict.items()
print(x)
```

Output

```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

CHECK IF KEY EXIST IN THE DICTIONARY

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

Output

Yes, 'model' is one of the keys in the thisdict dictionary

Class Activities

1.
2.
3.
4.