University of Tripoli
Faculty of Information Technology
Department of Software Engineering

# ITSE305 مواضيع مختارة
# Python Programming
# S2025

Lecture (3): Python Basics

## Python Collections (Arrays)

- There are four collection data types in Python:
  - **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.

## 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.
- 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
  - Cannot change the items after the set has been created. you can remove items and add new items.



## Python Sets

- Duplicates Not Allowed
  - Sets cannot have two items with the same value.
  - The values True and 1 are considered the same value in sets, and are treated as duplicates. As well as False and 0.
- len() function to determine how many items a set has.
- Set items can be of any data type and can contain different data types
- the set() constructor is used to make a set

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

## Access Set Items

- It cannot be accessed items in a set by referring to an index or a key.
- It can loop through the set items using a for loop
- The in keyword is used to ask for a specified value in a set



## Add Set Items

- Once a set is created, its items cannot be changed, but new items can be added
- To add an item to a set:
  - The add() method
  - the update() method (Note: The object in the update() method can be any iterable object )

## Add Set Items

```
collection = {"apple", "banana", "cherry"}
collection.add("orange")
print(collection)

tropical = {"pineapple", "mango", "papaya"}
collection.update(tropical)
print(collection)

mylist = ["kiwi", "orange"]
collection.update(mylist)
print(collection)
```

```
{'apple', 'banana', 'cherry', 'orange'}
{'banana', 'cherry', 'apple', 'papaya', 'pineapple', 'orange', 'mango'}
{'banana', 'cherry', 'kiwi', 'apple', 'papaya', 'pineapple', 'orange', 'mango'}
```



## Remove Set Items

- The remove() method (the item to be removed must be exist)
- The discard() method
- The pop() method (the return value is the removed item)
- The clear() method
- The del keyword

- Using pop() method, which item will be removed?

## Loop and Join Sets

- The for loop to loop through the set items
- To join two or more sets in Python:
  - The union() method or | operator joins all items from two sets or more.
  - The update() methods joins all items from both sets.
  - The intersection() method or & operator keeps ONLY the duplicates.
  - The difference() method or - operator keeps the items from the first set that are not in the other set(s).
  - The symmetric_difference() method or ^ operator keep all items EXCEPT the duplicates.
- **Note:**
  - All operators only allow to join sets with sets, and not with other data types.
  - The update() changes the original set, and does not return a new set.
  - Both union() and update() will exclude any duplicate items.



## 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.
- As of Python version 3.7, dictionaries are *ordered*. But, in Python 3.6 and earlier, dictionaries are *unordered*.
- Dictionaries are written with curly brackets
- 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

## Python Dictionaries

- len() function To determine how many items a dictionary has
- The values in dictionary items can be of any data type
- the dict() constructor is used to make a dictionary.



## Access Dictionary Items

- Accessing items by referring to its key name, inside square brackets or using the get() method
- The keys() method will return a list of all the keys in the dictionary.
- The values() method will return a list of all the values in the dictionary.
- The items() method will return each item in a dictionary, as tuples in a list.
- in keyword To determine if a specified key is present in a dictionary

## Change Dictionary Items

- You can change the value of a specific item by referring to its key name
- The update() method will update the dictionary with the items from the given argument that is a key:value pairs.



## Add Dictionary Items

- using a new index key and assigning a value to it
- The update() method will update the dictionary with the items from a given argument. If the item does not exist, the item will be added.

## Remove Dictionary Items

- The pop() method removes the item with the specified key name
- The del keyword removes the item with the specified key name or delete the dictionary completely
- The clear() method empties the dictionary



## Loop Dictionaries

- for loop to loop through a dictionary, the return value are the *keys* of the dictionary

## Copy Dictionaries

- copy() function
- dict() function



## Exercises

1) Update set1 by adding items from set2, except common, given:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
```

2) Check if two sets have any elements in common. If yes, display the common elements, given:

```
set1 = {10, 20, 30, 40, 50}
set2 = {60, 70, 80, 90, 10}
```

3) Update the first set with items that don't exist in the second set, given:

```
set1 = {10, 20, 30}
set2 = {20, 40, 50}
```

4) Get the key of a minimum value from the following dictionary, given:

```
sample_dict = {
  'Physics': 82,
  'Math': 65,
  'history': 75
}
```

## Exercises

5) Get the key of a minimum value from the following dictionary, given:

```
sample_dict = {
  'Physics': 82,
  'Math': 65,
  'history': 75
}
```

6) Print the value of key 'history' from the below dict, given:

```
sampleDict = {
    "class": {
        "student": {
            "name": "Mike",
            "marks": {
                "physics": 70,
                "history": 80
            }
        }
    }
}
```



# The END