University of Tripoli
Faculty of Information Technology
Department of Software Engineering

ITSE305 مواضيع مختارة
Python Programming
S2025

Lecture (2): 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 Lists

- Lists are used to store multiple items in a single variable.
- Lists are defined as objects with the data type 'list'
- Lists can be created using the list() constructor
- Lists are created using square brackets:
  - collection = ["apple", "banana", "cherry"]
- Lists are ordered
  - There are some list methods that will change the order
  - In general, the order of the items will not change
  - When adding new items to a list, they will be placed at the end of the list.
- Lists are changeable
  - We can change, add, and remove items in a list after it has been created.

List

- Lists are indexed
  - The first item has index [0], the second item has index [1], … etc.
- Lists allow Duplicates
  - Since lists are indexed, lists can have items with the same value
  - collection = ["apple", "banana", "cherry" , "banana"]
- print(len(collection)) : determine how many items a list has
- List items can be of any data type and contain different data types:
  - collection = ["apple", "banana", "cherry"]
  - collection = [1, 5, 7, 9, 3]
  - collection= [True, False, False]
  - collection = ["abc", 34, True, 40, "male"]

Access List Items

- The first item has index 0.
- Negative indexing means start from the end: -1 refers to the last item, -2 refers to the second last item … etc.
- When specifying a range, the return value will be a new list with the specified items.
  - The **start index is included** but the **end index is excluded**.
  - By leaving out the start value, the range will start at the first item
  - By leaving out the end value, the range will go on to the end of the list
  - Use the in keyword to determine if a specified item is present in a list

Access List Items

```
collection = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(collection[1])
print(collection[-1])
print(collection[2:5])
print(collection[:4])
print(collection[2:])
print(collection[-4:-1])
if "apple" in collection:
  print("Yes, 'apple' is included in the collection")
```

```
banana
mango
['cherry', 'orange', 'kiwi']
['apple', 'banana', 'cherry', 'orange']
['cherry', 'orange', 'kiwi', 'melon', 'mango']
['orange', 'kiwi', 'melon']
Yes, 'apple' is included in the collection
```

Change List Items

- To change the value of a specific item, refer to the index number
  - Collection[2] = "mango"
- To change the value of items within a specific range, define a list with the new values, and refer to the range of index numbers where you want to insert the new values
  - If you insert *more* items than you replace ???
  - If you insert *less* items than you replace ???
- collection.insert(2, "watermelon"): To insert a new list item at the specified index, without replacing any of the existing values

Add List Items

- To add an item to the end of the list, use the append() method
- To insert a list item at a specified index, use the insert() method.
- To append elements from *another list* to the current list, use the extend() method

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

collection.append("orange")
print(collection)

collection.insert(1, "mango")
print(collection)

collection1 = ["pineapple", "papaya"]
collection.extend(collection1)
print(collection)
```

```
['apple', 'banana', 'cherry']
['apple', 'banana', 'cherry', 'orange']
['apple', 'mango', 'banana', 'cherry', 'orange']
['apple', 'mango', 'banana', 'cherry', 'orange', 'pineapple', 'papaya']
```

Remove List Items

- The remove() method removes the specified item
  - If there are more than one item with the specified value, the remove() method removes the first occurrence
- The pop() method removes the specified index
  - Not specifying the index, the pop() method removes the last item.
- The del keyword also removes the specified index
  - The del keyword can also delete the list completely.
- The clear() method empties the list. The list still remains, but it has no content.

Remove List Items

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

collection.remove("apple")
print(collection)

collection.pop(1)
print(collection)

collection.pop()
print(collection)

del collection[0]
print(collection)

collection.clear()
print(collection)

del collection
```

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

Loop Lists

- for loop to loop through the list items
- range() and len() functions to loop through the list items by referring to their index number
- while loop to loop through the list items by referring to their indexes start at 0 with increasing the index by 1 after each iteration

```
lists = ["apple", "banana", "cherry"]
for x in lists:
  print(x)

print("\n")
for i in range(len(lists)):
  print(lists[i])

print("\n")
i = 0
while i < len(lists):
  print(lists[i])
  i = i + 1
```

```
apple
banana
cherry

apple
banana
cherry

apple
banana
cherry
```


Sort Lists

- sort() method: sort the list alphanumerically, case sensitive, ascending by default
- Use the keyword argument reverse = True with sort() method to sort the list descending
- reverse() method: reverses the current sorting order of the elements, regardless of the alphabet

Sort Lists

```
lists = ["orange", "mango", "kiwi", "Pineapple", "banana"]
lists.sort()
print(lists)
print("\n")

lists = ["orange", "mango", "kiwi", "Pineapple", "banana"]
lists.sort(reverse = True)
print(lists)
print("\n")

lists = ["orange", "mango", "kiwi", "Pineapple", "banana"]
lists.reverse()
print(lists)
```

```
['Pineapple', 'banana', 'kiwi', 'mango', 'orange']

['orange', 'mango', 'kiwi', 'banana', 'Pineapple']

['banana', 'Pineapple', 'kiwi', 'mango', 'orange']
```


Copy Lists

- The built-in List method copy()
- The built-in method list()
- the : (slice) operator.

- Why not using list2 = list1 ??

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

list1 = lists.copy()
print(list1)

list2 = list(lists)
print(list2)

list3 = lists[:]
print(list3)
```

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

Join Lists

- Using the + operator.
- Appending all the items from list2 into list1, one by one
- Using the extend() method

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)

list3 = list1.copy()
for x in list2:
  list3.append(x)
print(list3)

list3 = list1.copy()
list3.extend(list2)
print(list3)
```

```
['a', 'b', 'c', 1, 2, 3]
['a', 'b', 'c', 1, 2, 3]
['a', 'b', 'c', 1, 2, 3]
```


Other List Methods

- count(): Returns the number of elements with the specified value
- index(): Returns the index of the first element with the specified value

Python Tuples

- Tuples are used to store multiple items in a single variable.
- A tuple is a collection which is ordered, indexed and **unchangeable**.
- Tuples are written with round brackets.
- Tuples items are ordered
  - the items have a defined order, and that order will not change
- Tuples are unchangeable
  - Cannot be changed, added or removed after the tuple has been created.
- len() function to determine how many items a tuple has

Python Tuples

- Tuple items can be of any data type
  - tuple1 = ("apple", "banana", "cherry")
  - tuple2 = (1, 5, 7, 9, 3)
  - tuple3 = (True, False, False)
  - tuple4 = ("abc", 34, True, 40, "male")
- 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",)
- Use the tuple() constructor to make a tuple

Access Tuple Items

- Accessing tuple items by referring to the index number
  - The first item has index 0.
  - Negative indexing means start from the end, -1 refers to the last item, -2 refers to the second last item … etc.
- Specifying a range of indexes by specifying where to start and where to end the range
- Using the in keyword to determine if a specified item is present in a tuple

Update Tuples

- Tuples are unchangeable (**immutable**), meaning that you cannot change, add, or remove items once the tuple is created. But there are some workarounds.
  - Change items: You can convert the tuple into a list, change the list, and convert the list back into a tuple.
  - Add items: you can convert the tuple into a list, add your item(s), and convert it back into a tuple.
  - Remove items: you can convert the tuple into a list, remove your item(s), and convert it back into a tuple.
- Deleting the tuple completely using the del keyword
- **Note:** When creating a tuple with only one item, remember to include a comma after the item, otherwise it will not be identified as a tuple.

Update Tuples

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

```
("apple", "kiwi", "cherry")
```


Loop Tuples

- for loop
- the range() and len() functions
- while loop with the len() function

Join Tuples

- using the + operator
- using the * operator to multiply the content of a tuple a given number of times

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2

print(mytuple)
```

```
('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')
```


Other Tuple Methods

- 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

Exercises

1) Given a Python list, write a program to remove all occurrences of item 20, given:

```
list1 = [5, 20, 15, 20, 25, 50, 20]
```

2) You have given a nested list. Write a program to extend it by adding the sublist ["h", "i", "j"] in such a way that it will look like the following list:

```
list1 = ["a", "b", ["c", ["d", "e", ["f", "g"], "k"], "l"], "m", "n"]
```

3) Given a list of numbers. write a program to turn every item of a list into its square, given:

```
numbers = [1, 2, 3, 4, 5, 6, 7]
```


Exercises

4) Sort a tuple of tuples by 2nd item, given:

```
tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))
```

5) Swap two tuples in Python, given:

```
tuple1 = (11, 22)
tuple2 = (99, 88)
```

6) Reverse the tuple, given:

```
tuple1 = (10, 20, 30, 40, 50)
```

The END

by: Fatima Ben Lashihar