

## Working with Lists, Tuples, and Dictionaries

---

### Lab overview

In Python, string and numeric data types are often used in groups called *collections*. Three such collections that Python supports are the list, the tuple, and the dictionary.

In this lab, you will:

- Use the list data type
- Use the tuple data type
- Use the dictionary data type\_\_\_\_

## Defining a list

In this activity, you will edit a Python script to hold a collection of fruit names, or a list of fruit.

11. From the navigation pane of the IDE, choose the **.py** file that you created in the previous *Creating your Python exercise file* section.
12. In the file, enter the following code:

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

13. Save and run the file.
14. Confirm that the script runs correctly and that the output displays as you expect it to.

## Accessing a list by position

You can access the contents of a list by position. In this activity, you will print out each item in our list by their position:

15. In programming languages, the list position starts at zero (0). The brackets tell Python which position in the list you want. To access the **apple** string, enter the following code:

```
print(myFruitList[0])
```

16. To access the **banana** string, enter the following:

```
print(myFruitList[1])
```

17. To access the `cherry` string, enter the following code:

```
print(myFruitList[2])
```

18. Save and run the file.

19. Confirm that the script runs correctly and that the output displays as you expect it to.

### Changing the values in a list

The values of a list can be changed. In this activity, you will change `cherry` to `orange`.

20. In Python, list position starts at zero (0), so you must use the numeral 2 to access the third position. Enter the following code:

```
myFruitList[2] = "orange"
```

21. Print the updated list:

```
print(myFruitList)
```

22. Save and run the file.

23. Confirm that the script runs correctly and that the output displays as you expect it to.

```
['apple', 'banana', 'cherry']  
<class 'list'>  
apple  
banana  
cherry  
['apple', 'banana', 'orange']
```

## Exercise 2: Introducing the tuple data type

### Defining a tuple

The tuple is like a list, but it can't be changed. A data type that can't be changed after it's created is said to be *immutable*. To define a tuple, you use parentheses instead of brackets ([]).

24. Create a tuple by entering the following code:

```
myFinalAnswerTuple = ("apple", "banana", "pineapple")
print(myFinalAnswerTuple)
print(type(myFinalAnswerTuple))
```

25. Save and run the file.

26. Confirm that the script runs correctly and that the output displays as you expect it to.

### Accessing a tuple by position

Like a list, the items of a tuple can also be accessed by position:

27. To access the `apple` string, enter the following code:

```
print(myFinalAnswerTuple[0])
```

28. To access the `banana` string, enter the following code:

```
print(myFinalAnswerTuple[1])
```

29. To access the `pineapple` string, enter the following code:

```
print(myFinalAnswerTuple[2])
```

30. Save and run the file.

31. Near the top of the IDE window, choose the **Run** (Play) button.

32. Confirm that the script runs correctly and that the output displays as you expect it to.

```
['apple', 'banana', 'cherry']  
<class 'list'>  
apple  
banana  
cherry  
['apple', 'banana', 'orange']  
('apple', 'banana', 'pineapple')  
<class 'tuple'>  
apple  
banana  
pineapple
```

## Exercise 3: Introducing the dictionary data type

### Defining a dictionary

A dictionary is a list with named positions (keys). Imagine that your list shows people's favorite fruit.

33. Return to the Python script, and enter the following code:

```
myFavoriteFruitDictionary = {  
    "Akua" : "apple",  
    "Saanvi" : "banana",  
    "Paulo" : "pineapple"  
}
```

34. Use the `print()` function to write the dictionary to the shell:

```
print(myFavoriteFruitDictionary)
```

35. Use the `type()` function to write the data type to the shell:

```
print(type(myFavoriteFruitDictionary))
```

36. Save and run the file.

37. Confirm that the script runs correctly and that the output displays as you expect it to.

### Accessing a dictionary by name

In this activity, you will use the name of the individuals to get their favorite fruit, instead of numbers.

38. To access Akua's favorite fruit, enter the following code:

```
print(myFavoriteFruitDictionary["Akua"])
```

39. To access Saanvi's favorite fruit, enter the following code:

```
print(myFavoriteFruitDictionary["Saanvi"])
```

40. To access Paulo's favorite fruit, enter the following code:

```
print(myFavoriteFruitDictionary["Paulo"])
```

41. Save and run the file.

42. Confirm that the script runs correctly and that the output displays as you expect it to.

```
['apple', 'banana', 'cherry']  
<class 'list'>  
apple  
banana  
cherry  
['apple', 'banana', 'orange']  
('apple', 'banana', 'pineapple')  
<class 'tuple'>  
apple  
banana  
pineapple  
{'Akua': 'apple', 'Saanvi': 'banana', 'Paulo': 'pineapple'}  
<class 'dict'>  
apple  
banana  
pineapple
```

Congratulations! You have worked with the list, tuple, and dictionary data types in Python.

---