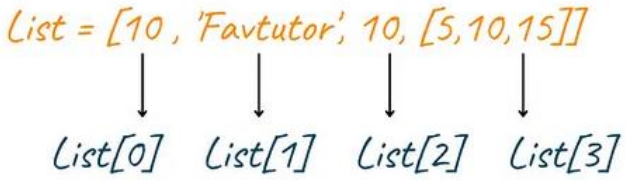
Lab 04: Lists and Tuples

What is a Python List ??

A **Python list** is a collection of data objects that maintains the order of its elements and allows for modifications. Unlike arrays, which are restricted to holding elements of a single type, lists in Python can accommodate a variety of object types, enabling a mixture of data within the same list.



- ✓ *Ordered:* Items have defined order which cannot be changed
- ✓ *Mutable:* Items can be modified anytime
- ✓ *Allow duplicates:* Items with the same value is allowed

List Creation

Definition: List creation refers to the process of initializing or defining a list in Python, which involves enclosing elements within square brackets `[ ]`.

Example:

```
# Creating a list of numbers
numbers = [1, 2, 3, 4, 5]
# Creating a list of strings
fruits = ['apple', 'banana', 'orange']
# Creating a list of mixed data types
mixed_list = [1, 'hello', True, 3.14]
```

Accessing an Element:

Definition: Accessing an element in a list involves retrieving a specific item from the list using its index position.

Example:

```
# Creating a list of strings
fruits = ['apple', 'banana', 'orange']
# Accessing the first element of the list
first_element = fruits[0] print(first_element)
# Output: 'apple'
# Accessing the last element of the list
last_element = fruits[-1] print(last_element)
# Output: 'orange
```

Adding and Removing Elements:

Definition: Adding elements involves inserting new items into a list, while removing elements entails deleting existing items from a list.

Example:

```
fruits = ['apple', 'banana', 'orange']
# Adding an element to the end of the list
fruits.append('grape')
print(fruits)
# Output: ['apple', 'banana', 'orange', 'grape']
# Removing an element by value
fruits.remove('banana')
print(fruits)
# Output: ['apple', 'orange', 'grape']
```

Looping in Lists:

Definition: Looping through lists involves iterating over each element in the list to perform a specific action.

Example:

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

Sort Lists:

Definition: Sorting lists involves arranging the elements of a list in ascending or descending order.

Example:

```
numbers=[2,1,4,5,3]
# Sorting a list of numbers in ascending order
numbers.sort()
print(numbers)  # Output: [1, 2, 3, 4, 5]
```

Copy Lists:

Definition: Copying lists involves creating a new list that contains the same elements as the original list.

Example:

```
# Copying a list using the copy() method
numbers_copy = numbers.copy()
print(numbers_copy)
# Output: [1, 2, 3, 4, 5]
```

Join Lists:

Definition: Joining lists involves combining the elements of multiple lists into a single list.

Example:

```
# Joining two lists using the extend() method
more_fruits = ['pineapple', 'mango']
fruits.extend(more_fruits)
print(fruits)
# Output: ['apple', 'orange', 'grape', 'pineapple', 'mango']
```

Python List Methods

by levelupcoding.co

	Description	Code example	Diagram
append()	Adds an element to the end of the list	letters = ['a', 'b', 'c'] letters.append('d')	
extend()	Adds the elements of a list to the end of another list	letters = ['a', 'b', 'c'] letters.extend(['d', 'e', 'f'])	
insert()	Adds an element at a specific index	letters = ['a', 'b', 'c'] letters.insert(1, 'x')	
remove()	Removes the first occurrence of an element	letters = ['a', 'b', 'c', 'b'] letters.remove('b')	
pop()	Removes and returns the element at a given index	letters = ['a', 'b', 'c', 'd'] letter = letters.pop(1)	
count()	Returns the number of times a value appears in a list	letters = ['c', 'b', 'c', 'c', 'd'] print(letters.count('c'))	
sort() @NikkiSiapno	Sorts the list in ascending order	letters = ['c', 'a', 'd', 'a', 'b'] letters.sort()	
reverse() @ChrisStaud	Reverses the order of elements in the list	letters = ['a', 'b', 'c'] letters.reverse()	

The list() Constructor

It is also possible to use the `list()` constructor when creating a new list.

Example

Using the `list()` constructor to make a List:

```
thislist = list(("apple", "banana", "cherry")) # note the double round-brackets
print(thislist)
```

Python Collections (Arrays)

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.

*Set *items* are unchangeable, but you can remove and/or add items whenever you like.

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

Tuple

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

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

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

Tuples are written with round brackets.

Example

Get your own Python Server

Create a Tuple:

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

Change Tuple Values

Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called.

But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

Example

Get your own Python Ser

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)
```

Add Items

Since tuples are immutable, they do not have a built-in `append()` method, but there are other ways to add items to a tuple.

1. **Convert into a list**: Just like the workaround for *changing* a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.

Example

Convert the tuple into a list, add "orange", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
```

2. **Add tuple to a tuple**. You are allowed to add tuples to tuples, so if you want to add one item, (or many), create a new tuple with the item(s), and add it to the existing tuple:

Example

Create a new tuple with the value "orange", and add that tuple:

```
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y

print(thistuple)
```

Remove Items

Note: You cannot remove items in a tuple.

Tuples are **unchangeable**, so you cannot remove items from it, but you can use the same workaround as we used for changing and adding tuple items:

Example

Convert the tuple into a list, remove "apple", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
```

Tuple methods and built in functions

Method	Description	Example

len()	Returns the length or the number of elements of the tuple passed as the argument	>>> tuple1 = (10,20,30,40,50)
		>>> len(tuple1)
		5

tuple()	Creates an empty tuple if no argument is passed
	
	Creates a tuple if a sequence is passed as argument
		>>> tuple1 = tuple()
		>>> tuple1
		( )
		>>> tuple1 = tuple('aeiou')#string
		>>> tuple1
		('a', 'e', 'i', 'o', 'u')
		
		>>> tuple2 = tuple([1,2,3]) #list
		>>> tuple2
		(1, 2, 3)
		
		>>> tuple3 = tuple(range(5))
		>>> tuple3
		(0, 1, 2, 3, 4)

count()	Returns the number of times the given element appears in the tuple	>>> tuple1 = (10,20,30,10,40,10,50)
		>>> tuple1.count(10)
		3
		>>> tuple1.count(90)
		0

index()	Returns the index of the first occurrence of the element in the given tuple	>>> tuple1 = (10,20,30,40,50)
		>>> tuple1.index(30)
		2
		>>> tuple1.index(90)
		ValueError: tuple.index(x): x not in tuple

sorted()	Takes elements in the tuple and returns a new sorted list. It should be noted that, sorted() does not make any change to the original tuple	>>> tuple1 = ("Rama","Heena","Raj", "Mohsin","Aditya")
		>>> sorted(tuple1)
		['Aditya', 'Heena', 'Mohsin', 'Raj', 'Rama']

min()
 	
max()
 	
sum()
	Returns minimum or smallest element of the tuple
	
	Returns maximum or largest element of the tuple
	
	Returns sum of the elements of the tuple
		>>> tuple1 = (19,12,56,18,9,87,34)
		>>> min(tuple1)
		9
		>>> max(tuple1)
		87
		>>> sum(tuple1)
		235

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 list
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 first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Loop Through the Index Numbers

You can also loop through the tuple items by referring to their index number.

Use the `range()` and `len()` functions to create a suitable iterable.

Example

Print all items by referring to their index number:

```
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):
  print(thistuple[i])
```

Using a While Loop

You can loop through the tuple items by using a `while` loop.

Use the `len()` function to determine the length of the tuple, then start at 0 and loop your way through the tuple items by referring to their indexes.

Remember to increase the index by 1 after each iteration.

Example

Print all items, using a `while` loop to go through all the index numbers:

```
thistuple = ("apple", "banana", "cherry")
i = 0
while i < len(thistuple):
  print(thistuple[i])
  i = i + 1
```

Exercise 01:

Create a list L = [10, 20, 30, 40, 50].

- Print the first element
- Print the last element
- Print the length of the list

Exercise 02:

Create a list containing squares of numbers from 1 to 10.

Exercise 03:

Given L = [3, 8, 1, 15, 7], compute:

- Maximum
- Minimum
- Sum

Exercise 04:

Create a tuple T = (4, 8, 15, 16)

- Print the second element
- Print the last element
- Print the length

Exercise 05:

Given T = (1, 2, 3, 4), convert it to a list and append the number 5.

Exercise 06:

Given

L = [1, 2, 2, 3, 3, 3, 4]

Remove duplicates and return a **tuple**.

Exercise 07:

Write a program that takes two arrays of 10 integers, a and b, as input. c is an array of 20 integers. The program should combine arrays a and b into c. The first 10 elements of c will be copied from array a, and the last 10 from array b. The program should then display array c.

Exercise 08:

Write a program that prompts the user to enter 10 integers, which will be stored in an array. The program should sort the array in ascending order and then display the resulting array.