LAB # 02

List, tuple, Dictionary, class and object

# OBJECTIVE

Familiarization with Python language using list, tuple, dictionary, class and object.

# THEORY

A *list* is a collection of items in a particular order. You can make a list that includes the letters of the alphabet, the digits from 0–9, or the names of all the people in your family. You can put anything you want into a list, and the items in your list don’t have to be related in any particular way. Because a list usually contains more than one element, it’s a good idea to make the name of your list plural, such as letters, digits, or names.

It can have any number of items and they may be of different types (integer, float, string etc.).In Python, square brackets ([]) indicate a list, and individual elementsin the list are separated by commas.

# empty list

my\_list = []

# list of integers

my\_list = [1, 2, 3]

#list of strings

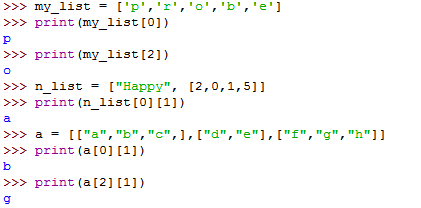
My\_list = [‘abc’,cde’]

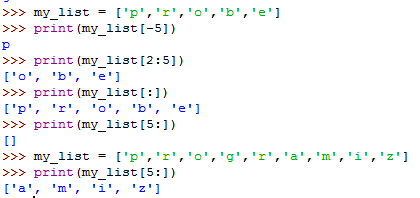
Also, a list can even have another list as an item. This is called nested list.

# nested list

my\_list = [[1,2,3], [8, 4, 6], [4]]

Some simple example of a list:





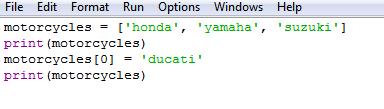
**Changing, Adding, and Removing Elements**

Most lists you create will be dynamic, meaning you’ll build a list and then add and remove elements from it as your program runs its course.

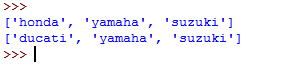
**Modifying Elements in a List:**

The syntax for modifying an element is similar to the syntax for accessingan element in a list. To change an element, use the name of the list followedby the index of the element you want to change, and then provide the newvalue you want that item to have.For example, let’s say we have a list of motorcycles, and the first item inthe list is 'honda'. How would we change the value of this first item?

**Example:**



**Output:**

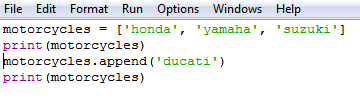
****

**Adding/Appending Elements to a List:**

You might want to add a new element to a list for many reasons. For example, you might want to make new aliens appear in a game, add new data to visualization, or add new registered users to a website you’ve built. Python provides several ways to add new data to existing lists.

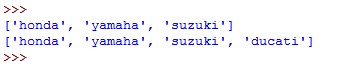
The simplest way to add a new element to a list is to *append* the item to the list. When you append an item to a list, the new element is added to the end of the list. Using the same list we had in the previous example, we’ll add the new element 'ducati' to the end of the list:

**Example:**



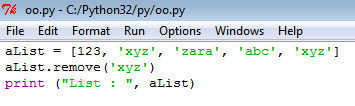
The append () method, adds 'ducati' to the end of the list without affecting any of the other elements in the list:

**Output:**



**Removing Elements from a List:**

**Example#06:**

****

**Output:**

****

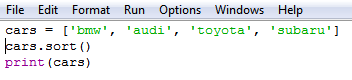
* The remove operation on a list is given a value to remove. It searches the list to find an item with that value and deletes the first matching item it finds. It is an error if there is no matching item.
* The del statement can be used to delete an entire list. If you have a specific list item as your argument to del. It is even possible to delete a "slice" from a list.
* The pop() is to delete the last item from a list as you use the list as a stack. Unlike del, pop returns the value that it popped off the list.

**Organizing a List:**

**Sorting a List Permanently with the sort() Method:**

Python’s sort() method makes it relatively easy to sort a list. Imagine we have a list of cars and want to change the order of the list to store them alphabetically. To keep the task simple, let’s assume that all the values in the list are lowercase.

**Example#07:**

****

**Output:**

****

## Built-in List Functions & Methods:

Python includes the following list functions −

|  |  |
| --- | --- |
| Sr.No. | Function with Description |
| 1 | [**cmp(list1, list2)**](https://www.tutorialspoint.com/python/list_cmp.htm)  Compares elements of both lists. |
| 2 | [**len(list)**](https://www.tutorialspoint.com/python/list_len.htm)  Gives the total length of the list. |
| 3 | [**max(list)**](https://www.tutorialspoint.com/python/list_max.htm)  Returns item from the list with max value. |
| 4 | [**min(list)**](https://www.tutorialspoint.com/python/list_min.htm)  Returns item from the list with min value. |
| 5 | [**list(seq)**](https://www.tutorialspoint.com/python/list_list.htm)  Converts a tuple into list. |

Python includes following list methods

|  |  |
| --- | --- |
| Sr.No. | Methods with Description |
| 1 | [**list.append(obj)**](https://www.tutorialspoint.com/python/list_append.htm)  Appends object obj to list |
| 2 | [**list.count(obj)**](https://www.tutorialspoint.com/python/list_count.htm)  Returns count of how many times obj occurs in list |
| 4 | [**list.index(obj)**](https://www.tutorialspoint.com/python/list_index.htm)  Returns the lowest index in list that obj appears |
| 5 | [**list.insert(index, obj)**](https://www.tutorialspoint.com/python/list_insert.htm)  Inserts object obj into list at offset index |
| 6 | [**list.pop(obj=list[-1])**](https://www.tutorialspoint.com/python/list_pop.htm)  Removes and returns last object or obj from list |
| 7 | [**list.remove(obj)**](https://www.tutorialspoint.com/python/list_remove.htm)  Removes object obj from list |
| 8 | [**list.reverse()**](https://www.tutorialspoint.com/python/list_reverse.htm)  Reverses objects of list in place |
| 9 | [**list.sort([func])**](https://www.tutorialspoint.com/python/list_sort.htm)  Sorts objects of list, use compare func if given |

**Tuples:**

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

tup1 = ('physics', 'chemistry', 1997, 2000);

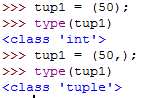
tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";

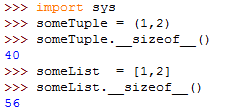
tup4 = ( );

To write a tuple containing a single value you have to include a comma, even though there is only one value –

tup1 = (50,);

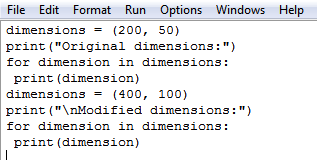
****

**Difference between Tuple and List:**



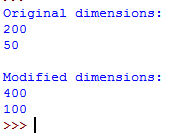
Due to the smaller size of a tuple operation, it becomes a bit faster, but not that much to mention about until you have a huge number of elements.

**Example:**

****

The block at line 1, defines the original tuple and prints the initial dimensions. At line 5, we store a new tuple in the variable dimensions. We then print the new dimensions at line 6. Python doesn’t raise any errors this time, because overwriting a variable is valid:

**Output:**

****

**Dictionary:**

A dictionaryin Python is a collection of key-value pairs. Each keyis connected to a value, and you can use a key to access the value associated with that key. A key’s value can be a number, a string, a list, or even another dictionary.Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair. Dictionaries can store an almost limitless amount of information.

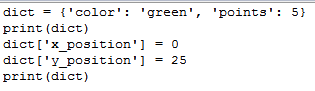
Creating a dictionary is as simple as placing items inside curly braces {} separated by comma. An item has a key and the corresponding value expressed as a pair, key: value.

dict = {'color': 'green', 'points': 5}

**Adding New Key-Value Pairs:**

Dictionaries are dynamic structures, and you can add new key-value pairsto a dictionary at any time. For example, to add a new key-value pair, you would give the name of the dictionary followed by the new key in square brackets along with the new value.

**Example:**



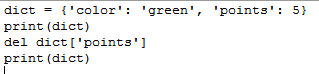
**Output:**

****

**Removing Key-value pair:**

When you no longer need a piece of information that’s stored in a dictionary, you can use the del statement to completely remove a key-value pair. All ‘del’ needs is the name of the dictionary and the key that you want to remove.

**Example:**

****

**Output:**

****

**Example:**

****

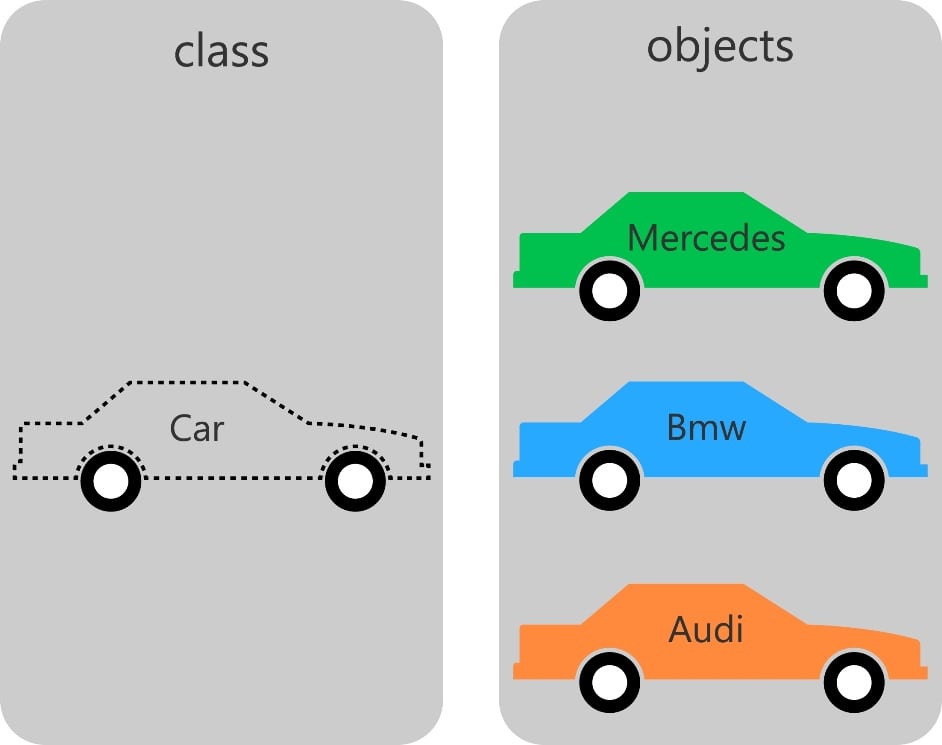
* The method items () returns a list of dict's (key, value) tuple pairs. The syntax of items () method is:dictionary.items ().
* The key-value pairs are not returned in the order in which they were stored, even when looping through a dictionary. Python doesn’t care about the order in which key-value pairs are stored; it tracks only the connections between individual keys and their values.

**Output:**

****

**Class and Object:**

* Python is an object oriented programming language. Unlike procedure oriented programming, where the main emphasis is on functions, object oriented programming stress on objects.
* A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.Class is a blueprint for the object.
* Object is simply a collection of data (variables) and methods (functions) that act on those data.



**Defining a Class in Python:**

Like function definitions begin with the keyword [def](https://www.programiz.com/python-programming/keyword-list#def), in Python, we define a class using the keyword [class](https://www.programiz.com/python-programming/keyword-list#class).

The first string is called docstring and has a brief description about the class. Although not mandatory, this is recommended.

Here is a simple class definition.

class MyNewClass:

'''This is a docstring. I have created a new class'''

pass

A class creates a new local namespace where all its attributes are defined. Attributes may be data or functions.

There are also special attributes in it that begins with double underscores (\_\_). For example, \_\_doc\_\_ gives us the docstring of that class.

**Output:**

****

**Creating an Object in Python:**

We saw that the class object could be used to access different attributes.

It can also be used to create new object instances (instantiation) of that class. The procedure to create an object is similar to a [function](https://www.programiz.com/python-programming/function)call.

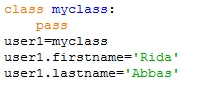
>>>ob =Class()

This will create a new instance object named ob. We can access attributes of objects using the object name prefix.

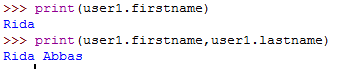
Attributes may be data or method. Method of an object are corresponding functions of that class. Any function object that is a class attribute defines a method for objects of that class.

This means to say, since MyClass.func is a function object (attribute of class), ob.func will be a method object.

**Example:**

****

**Output:**

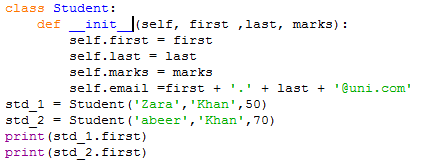
****

**Class Features:**

**Initialization (\_\_init\_\_):**

The \_\_init\_\_ method is run as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object. Notice the double underscore both in the beginning and at the end in the name.

**Example:**

****

**Lab#2 Exercise:**

1. Store the names of a few of your friends in a list called names. Print each person’s name by accessing each element in the list, one at a time

**Code:**

a=['Faheela','Rania','Hania','Sania']

for n in a:

print(n)

**Output:**

Faheela

Rania

Hania

Sania

1. If you could invite anyone, living or deceased, to dinner, who would you invite? Make a list that includes at least three people you’d like to invite to dinner. Then use your list to print a message to each person, inviting them to dinner.

**Code:**

a=input("Enter Your first guest name ")

b=input("Enter Your Second guest name ")

c=input("Enter Your Third guest name ")

d=[a,b,c]

for n in d:

print("Hey "+n+"! you are invited ")

**Output:**

Enter Your first guest name Rania

Enter Your Second guest name Sania

Enter Your Third guest name Hania

Hey Rania! you are invited for party

Hey Sania! you are invited for party

Hey Hania! you are invited for party

1. **Changing Guest List:** You just heard that one of your guests can’t make the dinner, so you need to send out a new set of invitations. You’ll have to think of someone else to invite.

* Modify your list, replacing the name of the guest who can’t make it with the name of the new person you are inviting.
* Print a second set of invitation messages, one for each person who is still in your list.

**Code:**

a=input("Enter Your first guest name ")

b=input("Enter Your Second guest name ")

c=input("Enter Your Third guest name ")

d=[a,b,c]

print('Sorry friends ! '+a+' IS BUSY ')

d.remove(a)

a=input("Enter Your new guest name ")

d.insert(0,a)

for n in d:

print("Hey "+n+"! you are invited for party ")

**Output:**

Enter Your first guest name Rania

Enter Your Second guest name Hania

Enter Your Third guest name Sania

Sorry friends ! Rania IS BUSY

Enter Your new guest name Hamid

Hey Hania! you are invited for party

Hey Sania! you are invited for party

Hey Hamid! you are invited for party

1. Create a Class “Emplyee”, it’s a common base class for all the employee. Then initialize employee’s parameter like empName and salary and create function like displayCount() contain total number of employee in your knowledge base and displayEmployee() contain empName and their salary.
2. Implement graph using adjacency list using list or dictionary , make a class such as Vertex and Graph then make some function such asadd\_nodes , add\_edges, add\_neighbors, add\_vertex, add\_vertices and suppose whatever you want to need it.

