



The Islamic University of Gaza
Faculty of Engineering
Computer Engineering Department



Introduction to Computers

LAB #7

Lists

Eng. Hashem M. Hejazy

Eng. Mohammed O. AlHabbash

Fall 2026

Introduction

Programs commonly need to store a large number of values. Suppose, for instance, that you need to read 100 numbers and compute their average. Your program first reads the numbers and computes their average. In order to accomplish this task, the numbers must all be stored in variables. To do this, you would have to create 100 variables and repeatedly write almost identical code 100 times. Writing a program this way is impractical. So, how do you solve this problem?

An efficient, organized approach is needed. Python provides a type called a list that stores a sequential collection of elements. In our example, you can store all 100 numbers in a list and access them through a single list variable.

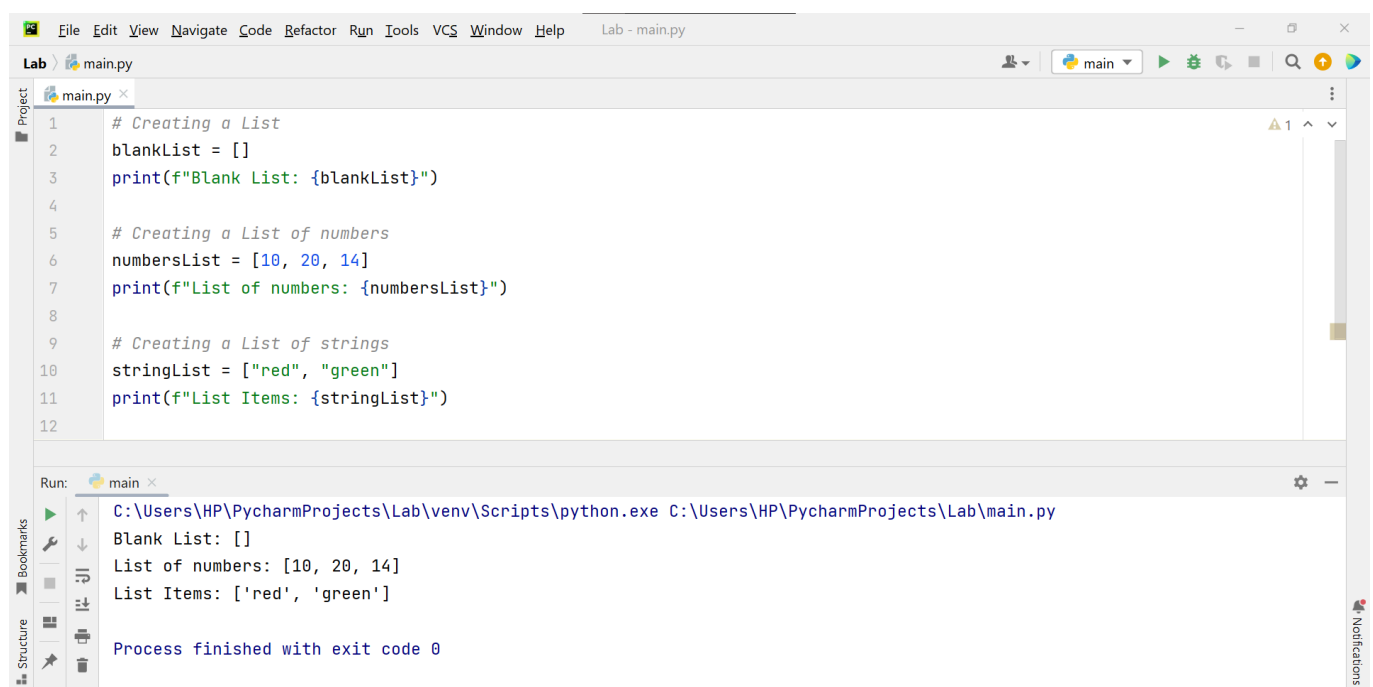
List Basics

An ordered sequence of items is called List. It is a very flexible data type in Python. There is no need for the value in the list to be of the same data type. The List is the data type that is a highly used data type in Python.

Creating Lists

Lists in Python can be created by just placing the sequence inside the square brackets[].

Example 1: Creating a list in Python



The screenshot displays the PyCharm IDE interface. The main editor window shows a Python file named `main.py` with the following code:

```
1 # Creating a List
2 blankList = []
3 print(f"Blank List: {blankList}")
4
5 # Creating a List of numbers
6 numbersList = [10, 20, 14]
7 print(f"List of numbers: {numbersList}")
8
9 # Creating a List of strings
10 stringList = ["red", "green"]
11 print(f"List Items: {stringList}")
12
```

Below the editor, the Run console shows the output of the program:

```
Run: main
C:\Users\HP\PycharmProjects\Lab\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\Lab\main.py
Blank List: []
List of numbers: [10, 20, 14]
List Items: ['red', 'green']
Process finished with exit code 0
```

Notes:

- The elements in a list are separated by commas and are enclosed by a pair of brackets ([]).
- A List can contain the elements of the same type or mixed types. For example, the following list is fine:

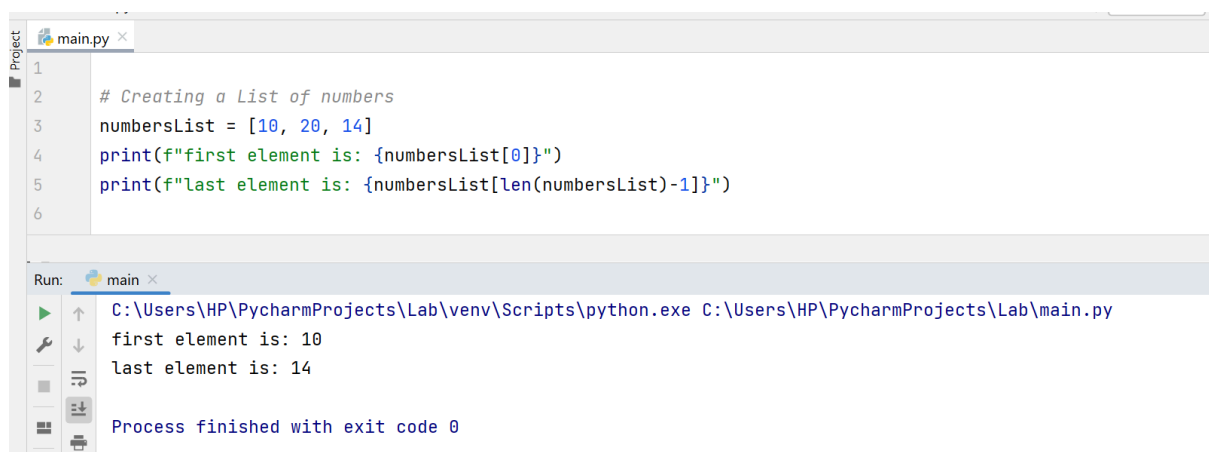
```
list = [2, "three", 4]
```

Accessing elements from the List

An element in a list can be accessed through the index operator, using the following syntax:

list[index]

List indexes are 0 based; that is, they range from 0 to len(list)-1



The screenshot shows a PyCharm IDE window with a file named 'main.py'. The code in the editor is as follows:

```
1
2 # Creating a List of numbers
3 numbersList = [10, 20, 14]
4 print(f"first element is: {numbersList[0]}")
5 print(f"last element is: {numbersList[len(numbersList)-1]}")
6
```

Below the editor, the 'Run' console shows the output of the script:

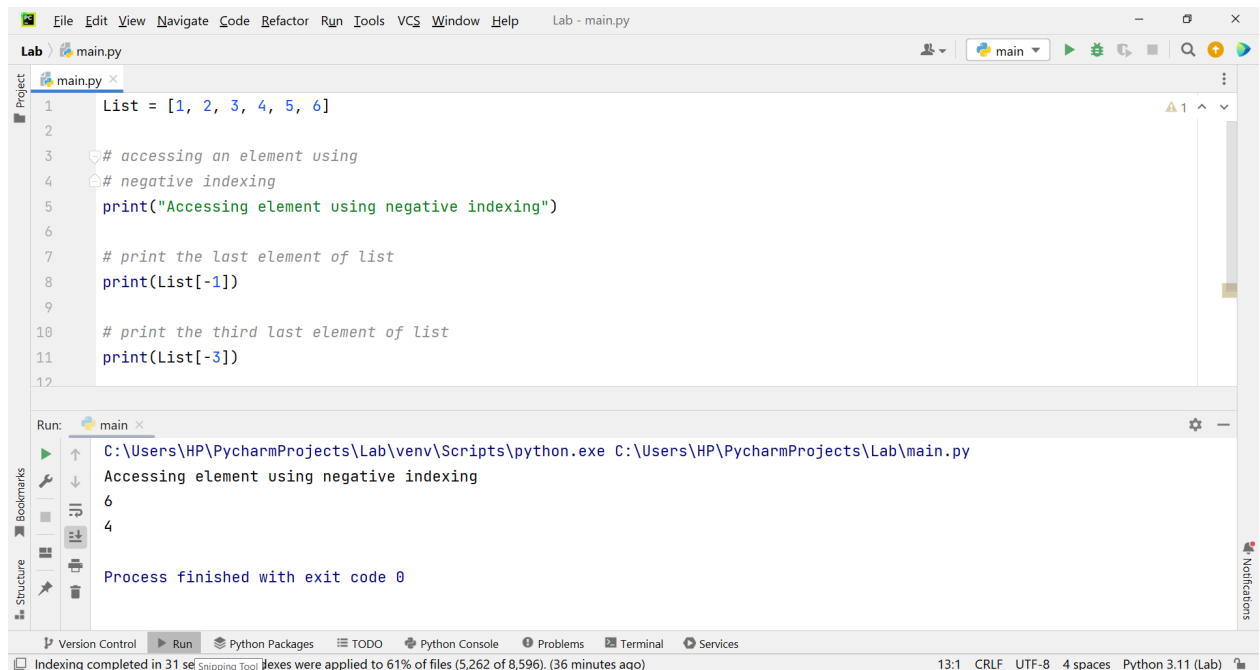
```
Run: main
C:\Users\HP\PycharmProjects\Lab\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\Lab\main.py
first element is: 10
last element is: 14
Process finished with exit code 0
```

Caution

- Accessing a list out of bounds is a common programming error that results in a runtime `IndexError`. To avoid this error, make sure that you do not use an index beyond `len(myList) - 1`.
- Programmers often mistakenly reference the first element in a list with index 1, but it should be 0.
- In many other programming languages, you would use a type called an array to store a sequence of data. An array has a fixed size. A Python list's size is flexible. It can grow and shrink on demand.

Negative indexing

In Python, negative sequence indexes represent positions from the end of the array. Instead of having to compute the offset as in `List[len(List)-3]`, it is enough to just write `List[-3]`. Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second-last item, etc.



```
1 List = [1, 2, 3, 4, 5, 6]
2
3 # accessing an element using
4 # negative indexing
5 print("Accessing element using negative indexing")
6
7 # print the last element of list
8 print(List[-1])
9
10 # print the third last element of list
11 print(List[-3])
12
```

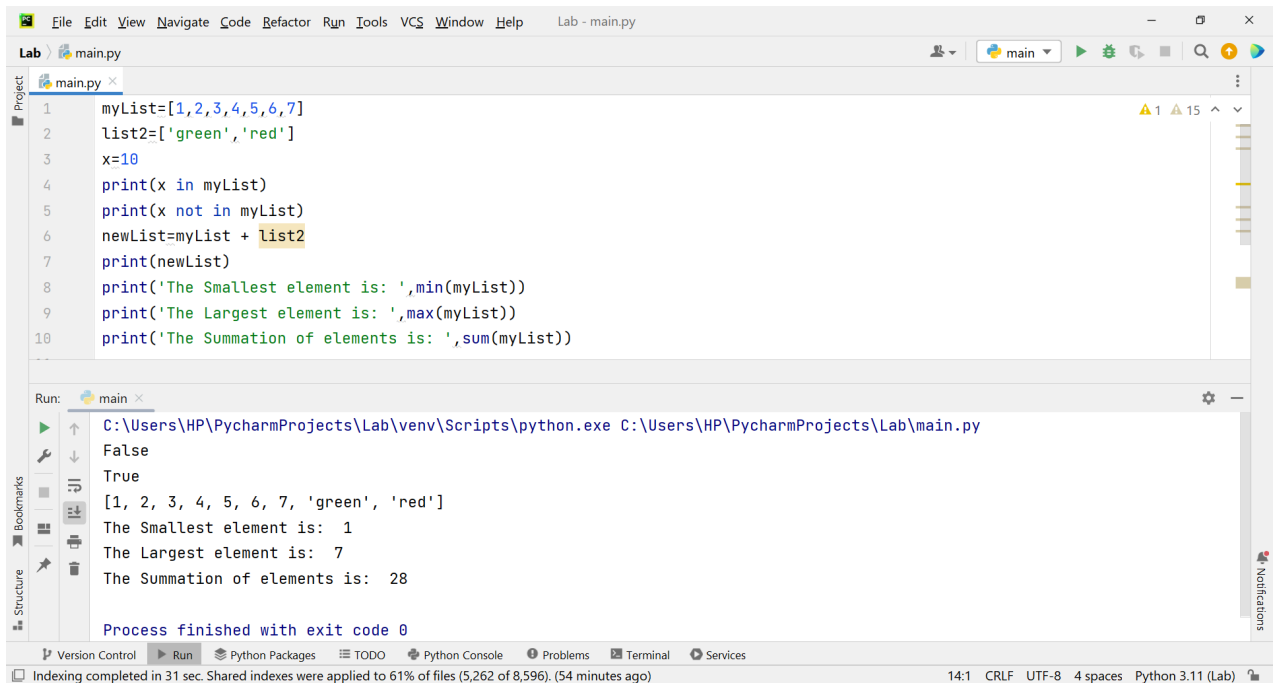
Run: main

```
C:\Users\HP\PycharmProjects\Lab\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\Lab\main.py
Accessing element using negative indexing
6
4
Process finished with exit code 0
```

Common Operations in list

s: is the list name, **x:** is an element

Operation	Description
<code>x in s</code>	True if element x is in sequence s.
<code>x not in s</code>	True if element x is not in sequence s.
<code>s1 + s2</code>	Concatenates two sequences s1 and s2.
<code>s[i]</code>	element number i in sequence s.
<code>s[i : j]</code>	Slice of sequence s from index i to j-1.
<code>min(s)</code>	Smallest element in sequence s.
<code>max(s)</code>	Largest element in sequence s.
<code>sum(s)</code>	Sum of all numbers in sequence s.



The screenshot shows the PyCharm IDE interface. The top toolbar includes menus like File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The main editor window displays a file named `main.py` with the following Python code:

```
1 myList=[1,2,3,4,5,6,7]
2 list2=['green','red']
3 x=10
4 print(x in myList)
5 print(x not in myList)
6 newList=myList + list2
7 print(newList)
8 print('The Smallest element is: ',min(myList))
9 print('The Largest element is: ',max(myList))
10 print('The Summation of elements is: ',sum(myList))
```

Below the editor, the 'Run' window shows the execution command and output:

```
Run: C:\Users\HP\PycharmProjects\Lab\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\Lab\main.py
False
True
[1, 2, 3, 4, 5, 6, 7, 'green', 'red']
The Smallest element is: 1
The Largest element is: 7
The Summation of elements is: 28
Process finished with exit code 0
```

The bottom status bar indicates 'Indexing completed in 31 sec. Shared indexes were applied to 61% of files (5,262 of 8,596). (54 minutes ago)' and shows settings for '14:1 CRLF UTF-8 4 spaces Python 3.11 (Lab)'.

Comparing Lists

You can compare lists using the comparison operators (`>`, `>=`, `<`, `<=`, `==`, and `!=`). For this to work, the two lists must contain the same type of elements. The first two elements are compared, and if they differ this determines the outcome of the comparison; if they are equal, the next two elements are compared, and so on, until either list is exhausted.

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The main editor window displays a file named `main.py` with the following Python code:

```

1 list1=['green','red','blue']
2 list2=['green','red','blue']
3 print(f"Is list1 Equal list2? {list1==list2}")
4 print(f"Is list1 Not Equal list2? {list1!=list2}")
5 print(f"Is list1 >= list2? {list1>list2}")
6 print(f"Is list1 < list2? {list1<list2}")
7

```

The left sidebar shows the Project view with the following structure:

- Lab C:\Users\HP\PycharmProjects\Lab
 - venv library root
 - main.py
- External Libraries
- Scratches and Consoles

The bottom Run window shows the execution output for the command:

```

C:\Users\HP\PycharmProjects\Lab\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\Lab\main.py

```

The output is:

```

Is list1 Equal list2? True
Is list1 Not Equal list2? False
Is list1 >= list2? False
Is list1 < list2? False

```

List Methods

Method	Description
count	Returns the number of times element x appears in the list.
append(x)	Adds an element x to the end of the list.
insert(index,x)	Inserts an element x at a given index. Note that the first element in the list has index 0.
pop(i)	Removes the element at the given position and returns it. The parameter i is optional. If it is not specified, list.pop() removes and returns the last element in the list.
remove (x)	Removes the first occurrence of element x from the list.
reverse	Reverses the elements in the list.
sort	Sorts the elements in the list in ascending order.

Project

main.py

```
1 list1=[5,0,3,3,7,-10]
2 print(f"Number 3 Repeted {list1.count(3)} Times")
3
4 #Adding elements to the list
5 list1.append(16)
6 print(f"list1 after adding an element at the end {list1}")
7 list1.insert(2,-3)
8 print(f"list1 after adding an element at index 2 {list1}")
9
10 #Remove elements from the list
11 list1.pop()
12 print(f"list1 after remove the last element {list1}")
13 list1.pop(1)
14 print(f"list1 after remove element at index 1 {list1}")
15 list1.remove(7)
16 print(f"list1 after remove the value 7 {list1}")
17
18 #reverse the list and sort
19 list1.reverse()
20 print(f"list1 after reverse {list1}")
21 list1.sort()
22 print(f"list1 after sorting {list1}")
```

Run: main

C:\Users\HP\PycharmProjects\Lab\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\Lab\main.py

Number 3 Repeted 2 Times

list1 after adding an element at the end [5, 0, 3, 3, 7, -10, 16]

list1 after adding an element at index 2 [5, 0, -3, 3, 3, 7, -10, 16]

list1 after remove the last element [5, 0, -3, 3, 3, 7, -10]

list1 after remove element at index 1 [5, -3, 3, 3, 7, -10]

list1 after remove the value 7 [5, -3, 3, 3, -10]

list1 after reverse [-10, 3, 3, -3, 5]

list1 after sorting [-10, -3, 3, 3, 5]

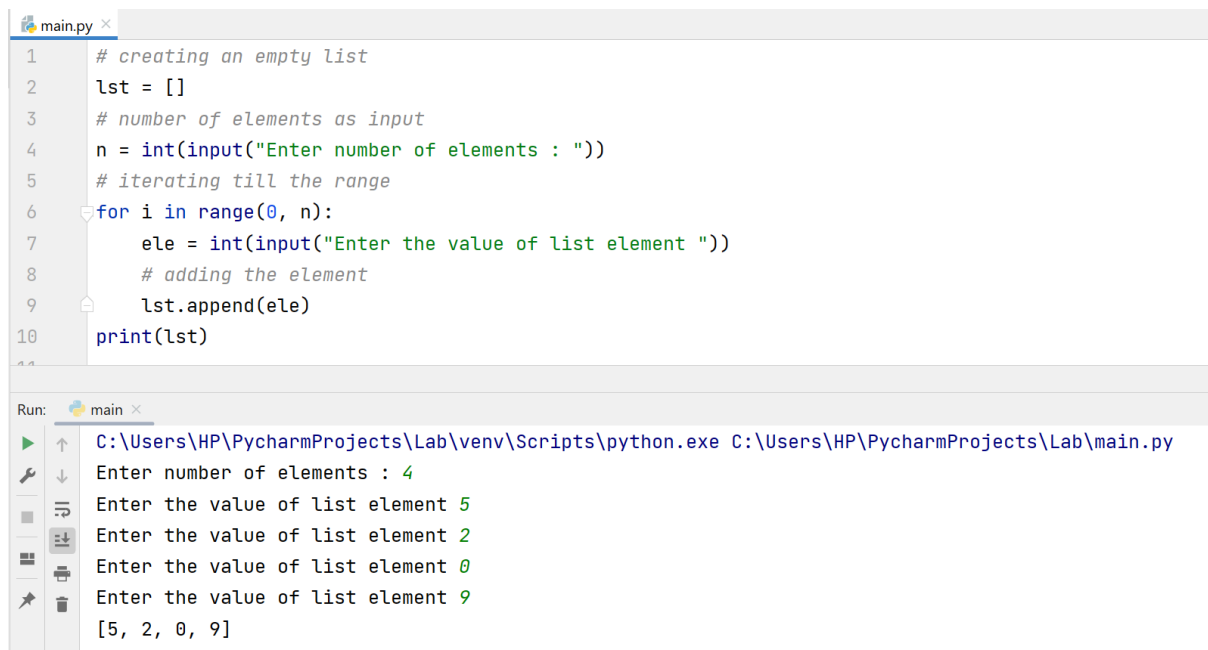
Process finished with exit code 0

Bookmarks

Structure

Get a list as input from user in Python

We can get a list of values using the for loop in python. For this, we can first create an empty list and a count variable. After that, we will ask the user for the total count of values in the list. After getting the total count of values, we can run the input() function count number of times using the for loop and add the input to the list using the append() method as follows.



```
1  # creating an empty list
2  lst = []
3  # number of elements as input
4  n = int(input("Enter number of elements : "))
5  # iterating till the range
6  for i in range(0, n):
7      ele = int(input("Enter the value of list element "))
8      # adding the element
9      lst.append(ele)
10 print(lst)
```

Run: main

```
C:\Users\HP\PycharmProjects\Lab\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\Lab\main.py
Enter number of elements : 4
Enter the value of list element 5
Enter the value of list element 2
Enter the value of list element 0
Enter the value of list element 9
[5, 2, 0, 9]
```

Splitting a String into a List

The str class contains the split method, which is useful for splitting items in a string into a list. For example, the following statement:

```
items = "Jane John Peter Susan".split()
```

splits the string Jane John Peter Susan into the list ['Jane', 'John', 'Peter', 'Susan']. In this case the items are delimited by spaces in the string. You can use a nonspace delimiter. For example, the following statement:

```
items = "09/20/2012".split("/")
```

splits the string 09/20/2012 into the list ['09', '20', '2012'].

