

# 1 ICS 104 - Introduction to Programming in Python and C

## 1.1 Lists, Tuples and Dictionaries

## 2 Lab Objectives

- To be familiar with common functions, operators and methods used with lists
- To implement algorithms using lists and sets.
- To structure programs using functions.

## 3 Common Functions and Operators Used with Lists

Operation	Description
$l[\text{from} : \text{to}]$	Creates a sublist from a subsequence of elements in list $l$ starting at position $\text{from}$ and going through but not including the element at position $\text{to}$ . Both $\text{from}$ and $\text{to}$ are optional. (See Special Topic 6.2.)
$\text{sum}(l)$	Computes the sum of the values in list $l$ .
$\text{min}(l)$ $\text{max}(l)$	Returns the minimum or maximum value in list $l$ .
$l_1 == l_2$	Tests whether two lists have the same elements, in the same order.

## 4 Common List Methods

Method	Description
<code>l.pop()</code> <code>l.pop(position)</code>	Removes the last element from the list or from the given position. All elements following the given position are moved up one place.
<code>l.insert(position, element)</code>	Inserts the element at the given position in the list. All elements at and following the given position are moved down.
<code>l.append(element)</code>	Appends the element to the end of the list.
<code>l.index(element)</code>	Returns the position of the given element in the list. The element must be in the list.
<code>l.remove(element)</code>	Removes the given element from the list and moves all elements following it up one position.
<code>l.sort()</code>	Sorts the elements in the list from smallest to largest.

## 5 Worked Example

- **Problem Statement:** A final quiz score is computed by adding all the scores, except for the lowest two.
  - For example, if the scores are 8 4 7 8.5 9.5 7 5 10, then the final score is 50.
- Write a program to compute a final score in this way.

- **Step 1:** Decompose your task into steps (What needs to be done).

- - Read the data into a list.
  - Process the data in one or more steps.
    - Remove the minimum.
    - Remove the minimum again.
    - Calculate the sum.
  - Display the results.

- **Step 2:** Determine which algorithms you need. (How are you going to achieve these steps)

- For finding the minimum,
  - You can either write code for that, or use the built-in functions and methods.

- **Step 3:** Use functions to structure the program
- Obviously, we need two functions:
  - A function to read the input ( `readFloats` )
  - A function to find the minimum and remove it from the list ( `removeMinimum` )
  - A `main` function that will call the above ones and solve the problem.
    - `scores = readFloats()`
    - `removeMinimum(scores)`
    - `removeMinimum(scores)`
    - `total = sum(scores)`
    - `print("Final score:", total)`

- **Step 4:** Develop test cases for the program.
  - Think of all different scenarios.
  - This will help you during the implementation of the program.

Test Case	Expected Output	Comment
8 4 7 8.5 9.5 7 5 10	50	See Step 1.
8 7 7 7 9	24	Only two instances of the low score should be removed.
8 7	0	After removing the low scores, no score remains.
(no inputs)	<b>Error</b>	That is not a legal input.

- **Step 5:** Implement the program.

In [17]:

```
1  ##
2  # This program computes a final score for a series of quiz scores: the sum after
3  # dropping the two lowest scores. The program uses a list.
4  #
5
6  def main() :
7      scores = readFloats()
8      if len(scores) > 1 :
9          removeMinimum(scores)
10         removeMinimum(scores)
11         total = sum(scores)
12         print("Final score:", total)
13     else :
14         print("At least two scores are required.")
15
16  ## Reads a sequence of floating-point numbers.
17  # @return a list containing the numbers
18  #
19  def readFloats() :
20      # Create an empty list.
21      values = []
22
23      # Read the input values into a list.
24      print("Please enter values, Q to quit:")
25      userInput = input("")
26      while userInput.upper() != "Q" :
27          values.append(float(userInput))
28          userInput = input("")
29
30      return values
31
32  ## Removes the minimum value from a list.
33  # @param values a list of size >= 1
34  #
35  def removeMinimum(values) :
36      smallestPosition = 0
37      for i in range(1, len(values)) :
38          if values[i] < values[smallestPosition] :
39              smallestPosition = i
40
41      values.pop(smallestPosition)
42
```

```
43 # Start the program.
44 main()
Please enter values, Q to quit:
8
4
7
8.5
9.5
7
5
10
q
Final score: 50.0
```

## 6 Exercises

- **Exercise # 1** Write a program that generates a sequence of 20 random values between 0 and 99 in a list, prints the sequence, sorts it, and prints the sorted sequence. Use the list `sort` method. Make sure that you use functions in your solution, including a `main` function.
- Following is a sample run

original sequence:

```
[28, 46, 31, 75, 20, 7, 9, 89, 9, 88, 6, 80, 37, 54, 11, 58, 12, 72, 55, 96]
```

sorted sequence:

```
[6, 7, 9, 9, 11, 12, 20, 28, 31, 37, 46, 54, 55, 58, 72, 75, 80, 88, 89, 96]
```

In [5]:

```

1  # Exercise # 1 - Source Code
2
3  from random import randint
4  def main():
5      lists = randomList()
6      sortedList(lists)
7
8  def randomList():
9      originalSequence = []
10     for i in range(20):
11         originalSequence.append(randint(0,99))
12     print("original sequence:")
13     print(originalSequence)
14     return originalSequence
15
16 def sortedList(randList):
17     randList.sort()
18     print("sorted sequence:")
19     print(randList)
20
21
22 main()

```

original sequence:

[1, 91, 26, 80, 97, 6, 34, 44, 52, 92, 33, 21, 96, 64, 12, 91, 76, 72, 94, 29]

sorted sequence:

[1, 6, 12, 21, 26, 29, 33, 34, 44, 52, 64, 72, 76, 80, 91, 91, 92, 94, 96, 97]

- **Excercise # 2** Write a program that reads a sequence of input values and displays a bar chart of the values, using asterisks, like this:

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

- You may assume that all values are positive. You will keep reading input until the letter `Q` is entered. In order to solve this problem, you need to first figure out the maximum value. That value's bar should be drawn with 40 asterisks. Shorter bars should use proportionally fewer asterisks.
- Make sure that you use functions in your solution, including a `main` function.
- Following are some sample runs

Please enter positive values, followed by Q to quit:

4

8

9

1

6

3

q

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

Please enter positive values, followed by Q to quit:

1

2

3

4

5

6

7

8

9

10

q

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

Please enter positive values, followed by Q to quit:

5

6

4

Q

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

Please enter positive values, followed by Q to quit:

2

Q

\*\*\*\*\*

Please enter positive values, followed by Q to quit:

1

2

q

\*\*\*\*\*

\*\*\*\*\*

In [6]:

```

1  # Exercise # 2 - Source Code
2
3  def main():
4      userNumbers = enteredValues()
5      maxValue = maximumValue(userNumbers)
6      userNumbers = proportionList(maxValue,userNumbers)
7      asteriks(userNumbers)
8
9  def enteredValues():
10     values = []
11     userInput = 0
12     while userInput != "Q" and userInput != "q":
13         userInput = input("Please enter positive values, followed by Q to quit ")
14         if userInput != "Q" and userInput != "q":
15             values.append(int(userInput))
16     return values
17
18 def maximumValue(lists):
19     maximum = max(lists)
20     return maximum
21
22 def proportionList(maximum,lists):
23     for i in range(len(lists)):
24         lists.append(int((40/maximum*(lists[0]))))
25         lists.pop(0)
26     return lists
27
28 def asteriks(lists):
29     for i in range(len(lists)):
30         print("*"*(lists[i]))
31
32 main()

```

```

Please enter positive values, followed by Q to quit 1
Please enter positive values, followed by Q to quit 2
Please enter positive values, followed by Q to quit q
*****
*****

```

- **Excercise # 3** A supermarket wants to reward its best customer of each day, showing the customer's name on a screen in the supermarket. For that purpose, the customer's purchase amount is stored in a list and the customer's name is stored in a

corresponding list.

- Implement a function `nameOfBestCustomer` that returns the name of the customer with the largest sale.

```
def nameOfBestCustomer(sales, customers)
```

- Write a program that prompts the cashier to enter all prices and names, adds them to two lists, calls the function that you implemented, and displays the result. Use a price of 0 as a sentinel.
- Make sure that you use functions in your solution, including a `main` function.
- Following are some sample runs

```
Enter the amount of sale (0 to end): 12
Enter the name of the customer who paid 12.0: Ahmad Salem
Enter the amount of sale (0 to end): 14
Enter the name of the customer who paid 14.0: Saleem al-Hilali
Enter the amount of sale (0 to end): 20
Enter the name of the customer who paid 20.0: Majid Ahmad
Enter the amount of sale (0 to end): 10
Enter the name of the customer who paid 10.0: Mustafa Salem
Enter the amount of sale (0 to end): 0
The best customer is Majid Ahmad
```

---

```
Enter the amount of sale (0 to end): 0
No Sales Today
```

In [7]:

```
1  # Exercise # 3 - Source Code
2
3  count = 0
4
5  def main():
6      lists = listsMaker()
7      if count != 0:
8          maxValue = maximumValue(lists)
9          bestCustomer(maxValue,lists)
10
11 def listsMaker():
12     prices = []
13     people = []
14     userInput = 1
15     while userInput != "0":
16         userInput = input("Enter the amount of sale (0 to end): ")
17         if userInput != "0":
18             prices.append(int(userInput))
19             userInput = float(userInput)
20             peopleInput = input("Enter the name of the customer who paid %.2f: "%userInput)
21             people.append(peopleInput)
22             global count
23             count += 1
24         if userInput == "0" and count == 0:
25             print("No Sales Today")
26     lists = prices + people
27     return lists
28 def maximumValue(lists):
29     maximum = max(lists[0:(len(lists)//2)])
30     return maximum
31 def bestCustomer(maximum,lists):
32     position = lists.index(maximum)
33     customer = lists[position+len(lists)//2]
34     print("The best customer is %s"%customer)
35
36 main()
37
38
```

Enter the amount of sale (0 to end): 0  
No Sales Today

- **Excercise # 4** Implement the sieve of Eratosthenes: a function for computing prime numbers, known to the ancient Greeks. Choose an integer  $n$ . This function will compute all prime numbers up to, and including,  $n$ .
  - First insert all numbers from 1 to  $n$  into a set.
  - Then erase all multiples of 2 (except 2); that is, 4, 6, 8, 10, 12, ... .
  - Erase all multiples of 3, that is, 6, 9, 12, 15, ... .
  - Go up to  $\sqrt{n}$ .
- The remaining numbers are all primes.
- Make sure that you use functions in your solution, including a `main` function.



© martin mcelligott/iStockphoto.

- Following are some sample runs

```
Enter a positive integer: 1
There are no prime numbers up to 1
```

```
Enter a positive integer: 9
prime numbers up to 9 are:
[2, 3, 5, 7]
```

```
Enter a positive integer: 29
prime numbers up to 29 are:
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

In [8]:

```
1  # Exercise # 4 - Source Code
2
3  def prime(a):
4      Numbers=[]
5      for i in range(2,a+1):
6          Numbers.append(i)
7      if Numbers.index(2)==0 and len(Numbers)==1:
8          return ""
9      else:
10         for k in range(4,len(Numbers)+1,2):
11             Numbers.remove(k)
12         for ele in Numbers:
13             if ele % 3==0 and ele!=3:
14                 Numbers.remove(ele)
15         for l in Numbers:
16             if round( l ** 0.5) ** 2 == l:
17                 Numbers.remove(l)
18         return Numbers
19
20 def main():
21     askNumbers=int(input("Enter a positive integer: "))
22     if askNumbers==1:
23         print("There are no prime numbers up to 1")
24     else:
25         k=prime(askNumbers)
26         print(("Prime numbers up to "+str(askNumbers)+" are: "))
27         print(k)
28
29 main()
```

Enter a positive integer: 29

Prime numbers up to 29 are:

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]