Slide Type Slide

1 ICS 104 - Introduction to Programming in Python and C

1.1 Lists, Tuples and Dictionaries

Slide Type Slide

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.

Slide Type Slide

3 Common Functions and Operators Used with Lists

Slide Type -

Operation	Description
`l[from : to]`	Creates a sublist from a subsequence of elements in list *l* starting at position `from` and going through but not including the element at position `to`. Both `from` and `to` are optional. (See Special Topic 6.2.)
`sum(l)`	Computes the sum of the values in list *l*.
`min(l)` `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.

Slide Type Slide

4 Common List Methods

Slide Type -

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

Slide Type Slide

5 Worked Example

Slide Type -

- **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.

Slide Type Fragment

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

Slide Type -

-
  - 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.

Slide Type Slide

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

Slide Type -

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

Slide Type Slide

- **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)

Slide Type Slide

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

Slide Type Fragment

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)	**Error**	That is not a legal input.

Slide Type Slide

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

In [5]:

Slide Type -

```
##
#  This program computes a final score for a series of quiz scores: the sum after
#  dropping the two lowest scores. The program uses a list.
#

def main() :
    scores = readFloats()
    if len(scores) > 1 :
        removeMinimum(scores)
        removeMinimum(scores)
        total = sum(scores)
        print("Final score:", total)
    else :
        print("At least two scores are required.")

## Reads a sequence of floating-point numbers.
#  @return a list containing the numbers
#
def readFloats() :
   # Create an empty list.
    values = []

   # Read the input values into a list.
    print("Please enter values, Q to quit:")
    userInput = input("")
    while userInput.upper() != "Q" :
        values.append(float(userInput))
        userInput = input("")

    return values

## Removes the minimum value from a list.
#  @param values a list of size >= 1
#
def removeMinimum(values) :
    values.remove(min(values))

# Start the program.
main()
```

```
Please enter values, Q to quit:
1
2
3
4
q
Final score: 7.0
```

Slide Type Slide

Excercises

Slide Type Fragment

- **Excercise # 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]
```

Slide Type: Fragment

- **Excercise # 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.
- Define a function `def generate_random(numOfElts)` that receives the number of elements to generate randomly. The function **returns** a list of random numbers (0 - 99) of the specified size (not only 20).
- 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 [17]:

Slide Type: -

```python
# Exercise # 1 - Source Code
from random import randint
def main():
    numOfEltsk = input("Enter a list size: ")
    numOfElts = int(numOfEltsk)
    lisT = generate_random(numOfElts)
    print(lisT)
    lisT.sort()
    print(lisT)
def generate_random(numOfElts):
    thelist = []
    for i in range(numOfElts):
        i = randint(0,99)
        thelist.append(i)
    return thelist

main()
```

```
Enter a list size: 20
[47, 37, 95, 71, 71, 55, 75, 58, 66, 84, 58, 70, 13, 2, 79, 26, 31, 57, 45, 99]
[2, 13, 26, 31, 37, 45, 47, 55, 57, 58, 58, 66, 70, 71, 71, 75, 79, 84, 95, 99]
```

Slide Type: Slide

4### - **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 entered 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.
- Define `def readInts()` function to read integers until user enters q. and **returns a list of entered integers**
- Define `def printStars(values)` function that **prints the bar chart using the list of `values`**
- 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 [1]:

Slide Type -

```python
# Exercise # 2 - Source Code
def main():
    printStars(readInts())


def readInts():
    values = []
    number = input("please enter positive values, followed by Q to quit:  ")
    while number != "q" and number != "Q":
        values.append(int(number))
        number = input("please enter positive values, followed by Q to quit:  ")
    return values

def printStars(values):
    MAX_VALUE = max(values)
    for i in values:
        print("*"*round((i/MAX_VALUE)*40))

main()

```

```
please enter positive values, followed by Q to quit:  4
please enter positive values, followed by Q to quit:  5
please enter positive values, followed by Q to quit:  2
please enter positive values, followed by Q to quit:  1
please enter positive values, followed by Q to quit:  5
please enter positive values, followed by Q to quit:  q
**********
**********
**********
********
**********
```

Slide Type Slide

5.1 - 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 [2]:

Slide Type -

```python
# Exercise # 3 - Source Code
def main():

    sales = []
    customers=[]
    BEST_CUSTOMER = nameOfBestCustomer(sales, customers)

def nameOfBestCustomer(sales, customers):
    sales = []
    customers=[]
    sale = int(input("Enter the amount of sale (0 to end) "))
    if sale > 0:
        while sale != 0:
            sales.append(sale)
            customer = input("Enter the name of the customer who paid "+str(sale)+": ")
            customers.append(customer)
            sale = int(input("Enter the amount of sale (0 to end) "))
        MAX_SALE = max(sales)
        postionOfMAX = sales.index(MAX_SALE)
        MAX_NAME = customers[postionOfMAX]
        print("The best customer is "+MAX_NAME)
    else:
        print("No sales Today")

main()
```

```
Enter the amount of sale (0 to end) 4
Enter the name of the customer who paid4: omar
Enter the amount of sale (0 to end) 50
Enter the name of the customer who paid50: ali
Enter the amount of sale (0 to end) 70
Enter the name of the customer who paid70: kk
Enter the amount of sale (0 to end) 0
The best customer is kk
```

Slide Type Slide

- **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.
- Define `def sieveOfEratosthenes(n)` that receives the maximum value and **returns a list of all prime numbers less than or equal to this value**
- in main() print the list of prime numbers.

Slide Type -

Slide Type -

- 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]:

Slide Type -

```python
#THIS CODE IS NOT WHAT THE QUESTION WANTS BUT IT WORKS :) ,,,



# Exercise # 4 - Source Code
def main():
    NUMBER = int(input("Enter a positive integer: "))
    if NUMBER <=0 :
        print("Must to enter a positive values")
    elif NUMBER == 1 :
        print("there are no prime numbers up to 1 ")
    elif NUMBER == 2:
        print("prime number is 2")
    elif NUMBER > 2:
        print("prime numbers up to",NUMBER,"are: ")
        print(sieveOfEratosthenes(NUMBER))


def sieveOfEratosthenes(n):
    primes = []
    for i in range(2,n+1):
        primes.append(i)
        if i>2 and (i%2) == 0 or i>3 and (i%3) == 0 or i**(1/2) in primes:
            primes.remove(i)
    return primes
main()

```

```
Enter a positive integer: 15
prime numbers up to 15 are:
[2, 3, 5, 7, 11, 13]
```

In [ ]:

Slide Type