Practical 2

Programming for Data Analytics

Introduction to Python Data Structures

|  |  |
| --- | --- |
|  |  |

# In this practical, you will …

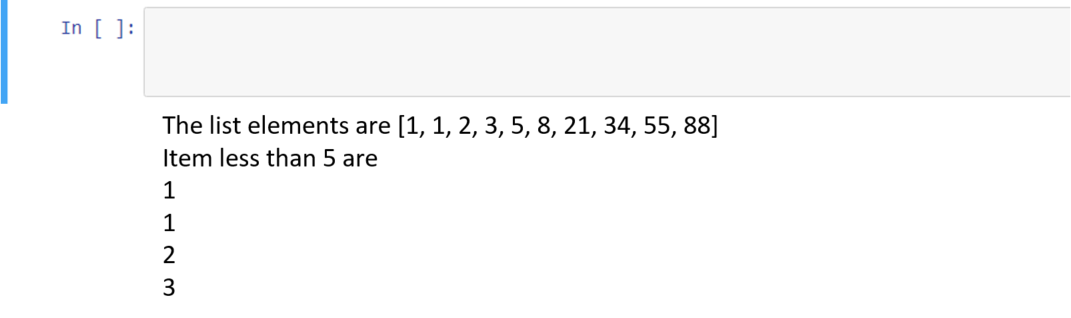
* use the Python list construct
* use the Python Dictionary construct
* write your own Python modules and functions

Python lists

### Task 1-1: Lists (Elements less than 5)

Given the following Python list, write a program that prints all the elements of the list that are less than 5.

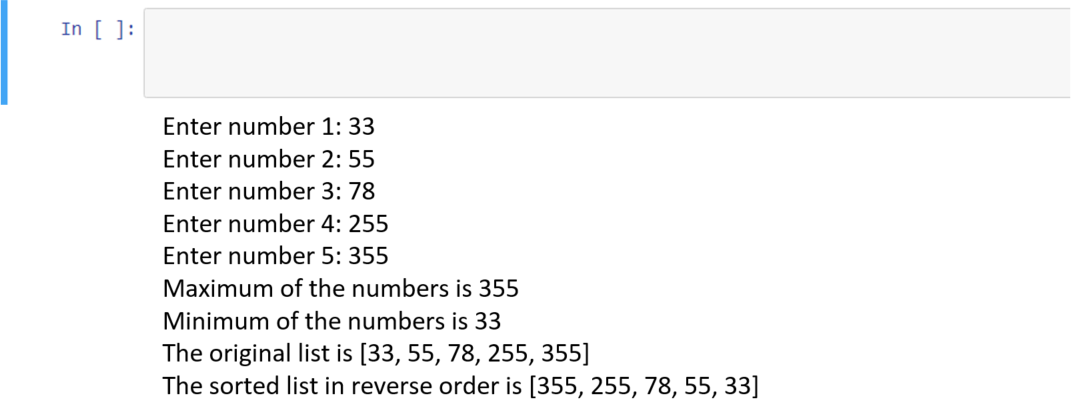
aList = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]



### Task 1-2: Lists (max, min, sorted, reverse)

Write a Python program that prompts the user to enter a list of 5 numbers and prints the maximum and minimum values that were entered. In addition, print the list sorted in reverse order (highest numbers first).

Your output should look similar to that below.



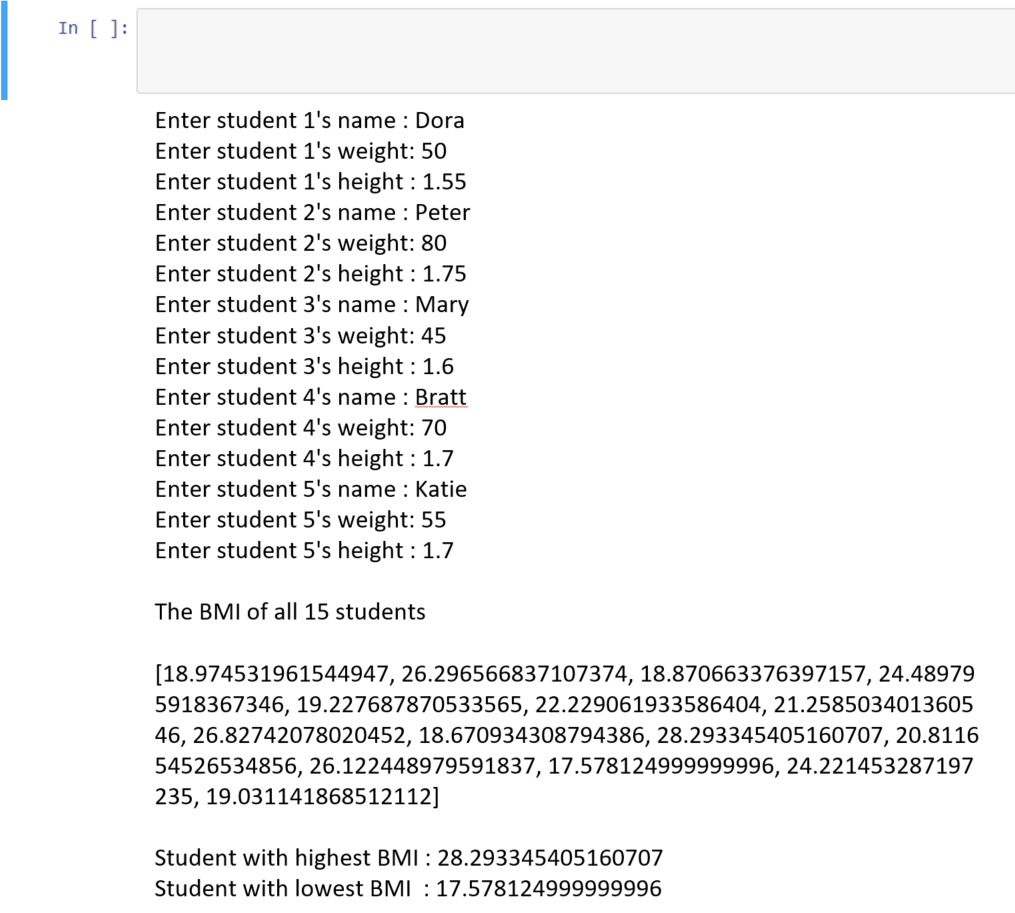
### Task 1-3: BMI

Write a Python program as follows:

Declare three Python lists at the beginning of the program with the following data

1. *list\_students1* – contains names of 10 students (Ann, Brandon, Christine, Daniel, Eileen, Francis, Gloria, Henry, Irene, John)
2. *list\_weights1* – contains the weights of the 10 students in kilogrammes (45,90,52,75, 48, 65, 60, 85, 49, 100)
3. *list\_heights1* – contains the heights of the 10 students in metres (1.54, 1.85, 1.66, 1.75, 1.58, 1.71, 1.68, 1.78, 1.62, 1.88)

Your program should:

1. Use a **for** loop to prompt the user to enter the names, weights and heights of 5 more students, storing the inputs into the *list\_students2*, *list\_weights2* and *list\_heights2* variables respectively
2. Combine *list\_students1* and *list\_students2* into a single list named *list\_students\_all*
3. Combine *list\_weights1* and *list\_weights2* into a single list named *list\_weights\_all*
4. Combine *list\_heights1* and *list\_heights2* into a single list named *list\_heights\_all*
5. Use a **for** loop to calculate the BMIs of the 15 students, and store them in a list variable named *list\_bmis\_all*. (BMI is obtained by weight/height2)
6. Print the student with the highest BMI and lowest BMI as shown below

### Task 1-4: List Slicing

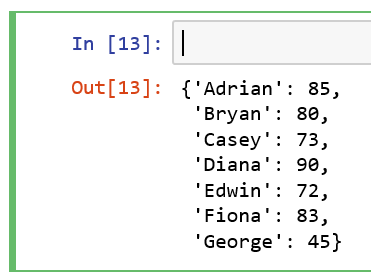
Given the three Python lists below, provide the code that will retrieve the required elements in each list.

list\_1 = [300,50,80,90,199,800,74,33]

list\_2 = ['Apple','Banana','Durian','Grapes','Papaya','Watermelon']

list\_3 = [0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8,9.9]

1. The numbers 800, 74 and 33 in list\_1
2. The last two elements in list\_2
3. All the elements in list\_3 except the first and the last

Python Dictionary

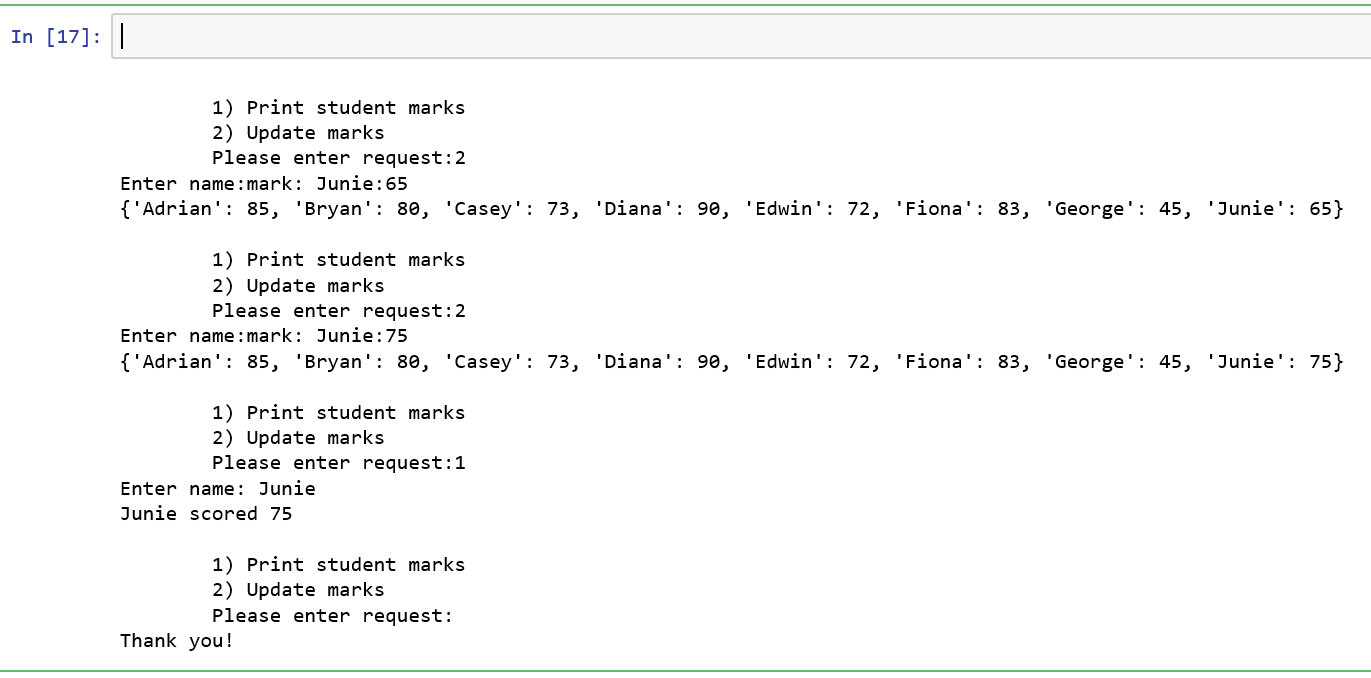
### Task 2-1: Populate a Dictionary

Given the Python string below, provide the code that will first split the string by ‘|’ to retrieve the name-mark pairs of each student. Then for each pair, split using the ‘:’ to store the name and marks into a dictionary and print its contents. s='Adrian:85|Bryan:80|Casey:73|Diana:90|Edwin:72|Fiona:83|George:45'

### Task 2-2: Process a Dictionary

Allow user to

1. Enter a name to print the student’s marks.
2. Update the marks for a student (add if not exist, overwrite if exist)



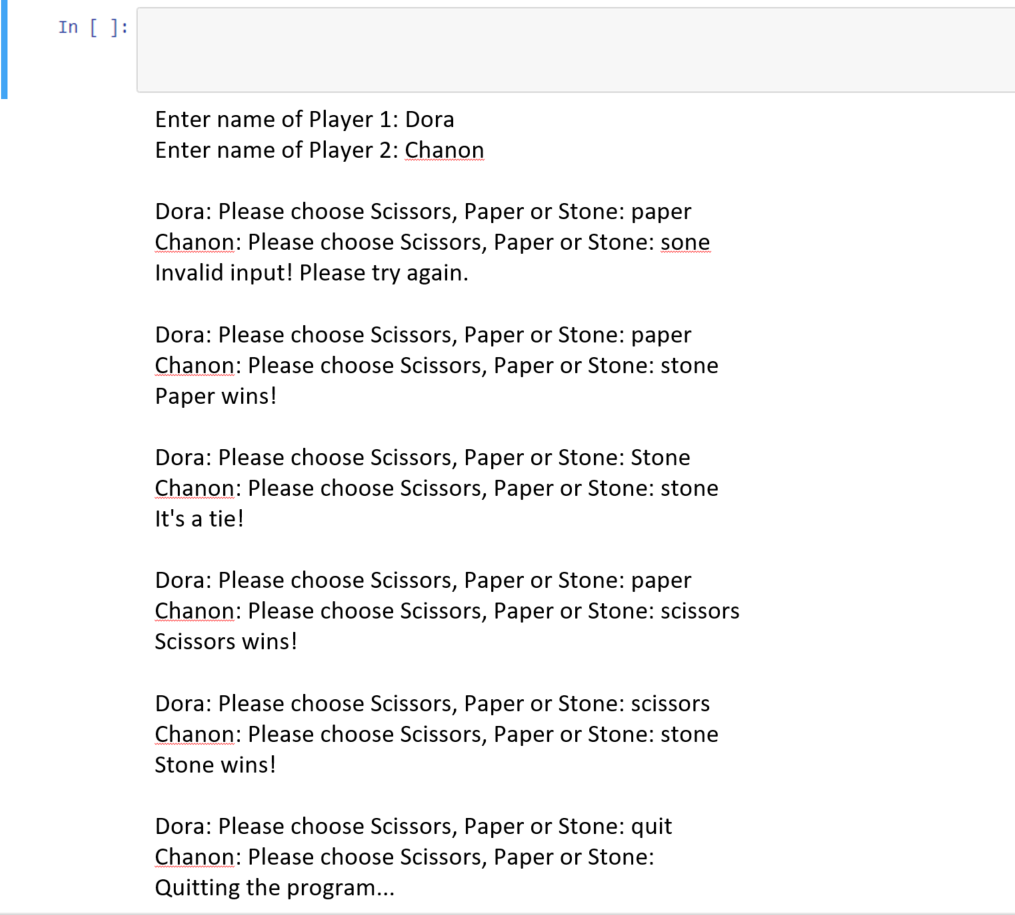
Python modules and functions

### Task 3-1: Scissors, Paper, Stone

Write a Python function compare(input1, input2) that takes in two string inputs which can be one of these values: Scissors, Paper or Stone. Your function should return two outputs: a message and a True or False value as follows:

|  |  |
| --- | --- |
| * If the two inputs are the same, e.g. both are “Scissors”, return “It’s a tie”, False * If input1 is “Scissors” and input2 is “Paper”, return “Scissors wins!”, False * If input1 is “Scissors” and input2 is “Stone”, return “Stone wins!”, False * If input1 is “Paper” and input2 is “Scissors”, return “Scissors wins!”, False * If input1 is “Paper” and input2 is “Stone”, return “Paper wins!”, False | * If input1 is “Stone” and input2 is “Scissors”, return “Stone wins!”, False * If input1 is “Stone” and input2 is “Paper”, return “Paper wins!”, False * If input1 is “Quit” or input2 is “Quit”, return “Quitting the program”, True * If either of input1 or input2 is not one of these values: Scissors, Paper, Stone, Quit, then return “Invalid input. Please try again”, False |

Using this function, create a Python program that simulates a two-player Scissors-Paper-Stone game that keeps repeating until any one of the user keys in a “quit” as his choice. A sample run is shown below.



### Task 3-2: Odd and Even marks

Write a Python function that takes in a list of numbers, and outputs them into two lists, one consisting of even numbers, and the other odd numbers.

Your program should:

1. Generate a list with 20 random numbers in the range of 1 to 100 and store them in the variable original\_list. Hint: You can generate a list of numbers using list comprehension and the random.randint() function

Example:

# Generate 1000 numbers from 1 to 10

numbers = [random.randint(1,10) for x in range(0,1000)]

1. Define a function called ***oddandeven(numbers\_list)*** that takes in one parameter of type list containing a list of numbers and returns two outputs of two lists, one that contains even numbers, and the other that contains odd numbers
2. After writing your function, call the ***oddandeven(numbers\_list)*** function, passing in original\_list as the parameter, and storing the outputs of the function as even\_numbers and odd\_numbers respectively
3. Print the contents of even\_numbers and odd\_numbers as shown below

Original List: [13, 19, 35, 27, 10, 96, 3, 73, 12, 68, 72, 10, 26, 68, 3, 19, 54, 75, 75, 73]

Odd [13, 19, 35, 27, 3, 73, 3, 19, 75, 75, 73]

Even [10, 96, 12, 68, 72, 10, 26, 68, 54]

1. Using the original\_list list generated in step 1., redo the question, this time without using the ***oddandeven(numbers\_list)*** function. Instead, use **filter** to separate the elements in the list. The output is the same as for step 4.
2. Now let’s say this set of random marks represent a continual assessment mark for a module, and takes up 20% of the final grade. Hence, using the original\_list list generated in step 1., apply **map** to convert the marks to 20%, (eg 100 marks becomes 20, 50 marks becomes 10 etc) and print it.

CA1 marks: [2.6, 3.8, 7.0, 5.4, 2.0, 19.2, 0.6, 14.6, 2.4, 13.6, 14.4, 2.0, 5.2, 13.6, 0.6, 3.8, 10.8, 15.0, 15.0, 14.6]

1. Finally, use **reduce** from functools to compute for the average marks for the marks in the original\_list. Note, there are many ways to compute for average here but let us try the **reduce** method.

Average mark: 41.55

**-- End --**