YZV102E/104E - Introduction to Programming for Data Science (Python) Lab 4

20.03.2025
Res. Asst. Erhan Biçer (bicer21@itu.edu.tr)
Res. Asst. Uğur Önal (onalug@itu.edu.tr)
Res. Asst. Barış Bilen ()
Res. Asst. Püren Tap ()
Res. Asst. Sümeyye Öztürk ()

1 Exercise 1

In this part, we will complete the following tasks;

1. Define a function named as $f1$ that takes 2 integer inputs. The function $f1$, returns two outputs. The first output is the sum of the inputs and second output is the multiplication of the inputs.
2. Create two variable.
3. Call the function $f1$ with the created variables.
4. Print the results.
5. Test the function with the test cases given in Table 1

Table 1: Test Case for Example 1

Test Cases				
Test Case #	Input 1	Input 2	Output 1	Output 2
1	2	3	5	6
2	-4	10	6	-40

1.1 Solution of Exercise 1

The solution is given in Code Snippet 3;

Code Snippet 1: Solution of Exercise 1

```
1  # step 1
2  def f1(a, b):
3    return a+b, a*b
4
5  def f1_alt(a:int, b:int):
6    return a+b, a*b
7
8  def f1_alt_2(a:int, b:int=0):
9    return a+b, a*b
10
11 def f1_alt_3(a:int=0, b:int=0):
12   return a+b, a*b
13
14 # step 2
15 var1 = 2
16 var2 = 3
17
18 # step 3
19 a, b = f1(var1, var2)
20
21 # step 4
22 print(a, b)
23
24 # alternatives
25 print(f1(a=var1, b=var2))
26 print(f1_alt(a=var1, b=var2))
27 print(f1_alt_2(a=var1, b=var2))
28 print(f1_alt_2(a=var1))
29 print(f1_alt_3())
30
31 # test_case 1
32 print(f1(2,3))
33 # test_case 2
34 print(f1(4,-10))
35
```

2 Exercise 2

In this part, you will complete the following tasks;

1. Define a function named as *average_above_threshold* that takes a list and a threshold. It calculates the average of the list elements whose value bigger than the threshold. It returns the average.**Note:** What if the list has no item whose value is not bigger than threshold ?
2. Get an input from user and make it float list named as $lst1$.
3. Get a float from user *threshold* it will be the threshold.
4. Call the function *average_above_threshold* with taken parameters.
5. Print the result with 5 precision.
6. Test the function with the test cases given in Table 2

Table 2: Test Case for Example 2

Test Cases		
Test Case #	Input	Output
1	input_list = 10 20 30 40 50 threshold = 14	35.00000
2	input_list = 23.5 47 98 -100 threshold = -12	56.16667
3	input_list = 0 1 2 3 threshold = 12	0.00000

2.1 Solution of Exercise 2

The solution is given in Code Snippet 2;

Code Snippet 2: Solution of Exercise 2

```python
# step 1
def average_above_threshold(lst:list=[], threshold:float=0.0):
  sum_list = 0
  num_above = 0
  for item in lst:
    if item > threshold:
      sum_list += item
      num_above += 1

  if num_above > 0:
    return sum_list / num_above
  else:
    return 0

# step 2
input_list = input().split(" ")
input_list = [float(item) for item in input_list]

# step 4
threshold = float(input())   # taking input and casting to float

# step 5
result = average_above_threshold(input_list, threshold)
print("The average is: %.5f" % result)

```

3 Exercise 3

In this part, we will complete the following tasks;

1. Define a function named $recursive\_sum\_of\_list$ that takes a list as input. The function $recursive\_sum\_of\_list$ is a recursive function that calculates the sum of the list in recursive way.
2. Get an input from user and make it float list named as $lst1$.
3. Call the function $recursive\_sum\_of\_list$ with the $lst1$.
4. Print the result with 5 precision.
5. Test the function with the test cases given in Table 3

Table 3: Test Case for Example 3

Test Cases		
Test Case #	Input	Output
1	10 55 33 78 1.2	177.20000
2	-5 -4 -3 -2 -1 0 1 2	-12.00000
3	70	70.00000

3.1 Solution of Exercise 3

The solutions are given in Code Snippets 3 and 4;

Code Snippet 3: Solution of Exercise 3

```
# step 1
def recursive_sum_of_list(lst):
    if len(lst) == 1:
        return lst[0]
    # recursive case
    else:
        return lst[0] + recursive_sum_of_list(lst[1:])

# step 2
input_list = input().split(" ")
input_list = [float(item) for item in input_list]

# step 3
result = recursive_sum_of_list(input_list)

# step 4
print("The recursive sum of a list is: %.5f" % result)

```

Code Snippet 4: Solution of Exercise 3

```
# step 1
def recursive_sum_of_list(lst:list):
  # base case
  if not lst:   # len(lst) == 0
    return 0
  # recursion case
  else:
    val = lst.pop()
    return val + recursive_sum_of_list(lst)

# step 2
lst1 = input().split(" ")
lst1 = [float(item) for item in lst1]
# step 3
result = recursive_sum_of_list(lst1)
# step 4
```

```
print(f"{result:.5f}")

```

4 Exercise 4

In this part, you will complete the following tasks;

1. Define a function named as *recursive_sum_of_digits_in_integer* that takes an integer. It calculates the sum of the digits in the integer in a recursive way. You can use arithmetic operators.**Note:** The input integer will not be negative.
2. Get an integer from user named as $num1$.
3. Call the function *recursive_sum_of_digits_in_integer* with $n1$.
4. Print the result.
5. Test the function with the test cases given in Table 4

Table 4: Test Case for Example 4

Test Cases		
Test Case #	Input	Output
1	98756	35
2	65789	35
3	0	0
4	123	6

5 Mockup Midterm Question

In the realm of programming, text encryption and decryption stand as crucial practices with widespread application. Throughout history, individuals have relied on computers and programming languages for these purposes. In this question, your task is to develop a text decryption algorithm using the recursive function approach. The decryption algorithm that you will utilize in this question is called the "Smaller End Trimming Algorithm".
The working process of the Smaller End Trimming Algorithm is as follows:

1. Checks the leftmost and the rightmost characters of the encrypted word.
2. Removes the character with the smaller value (according to lexical order) from the word. If both characters have the same value, the algorithm retains the leftmost occurrence and removes the rightmost one.
3. Moves one character inward from each side.
4. Repeats steps 1-3 until no further characters remain to compare.
5. Returns the decrypted word.

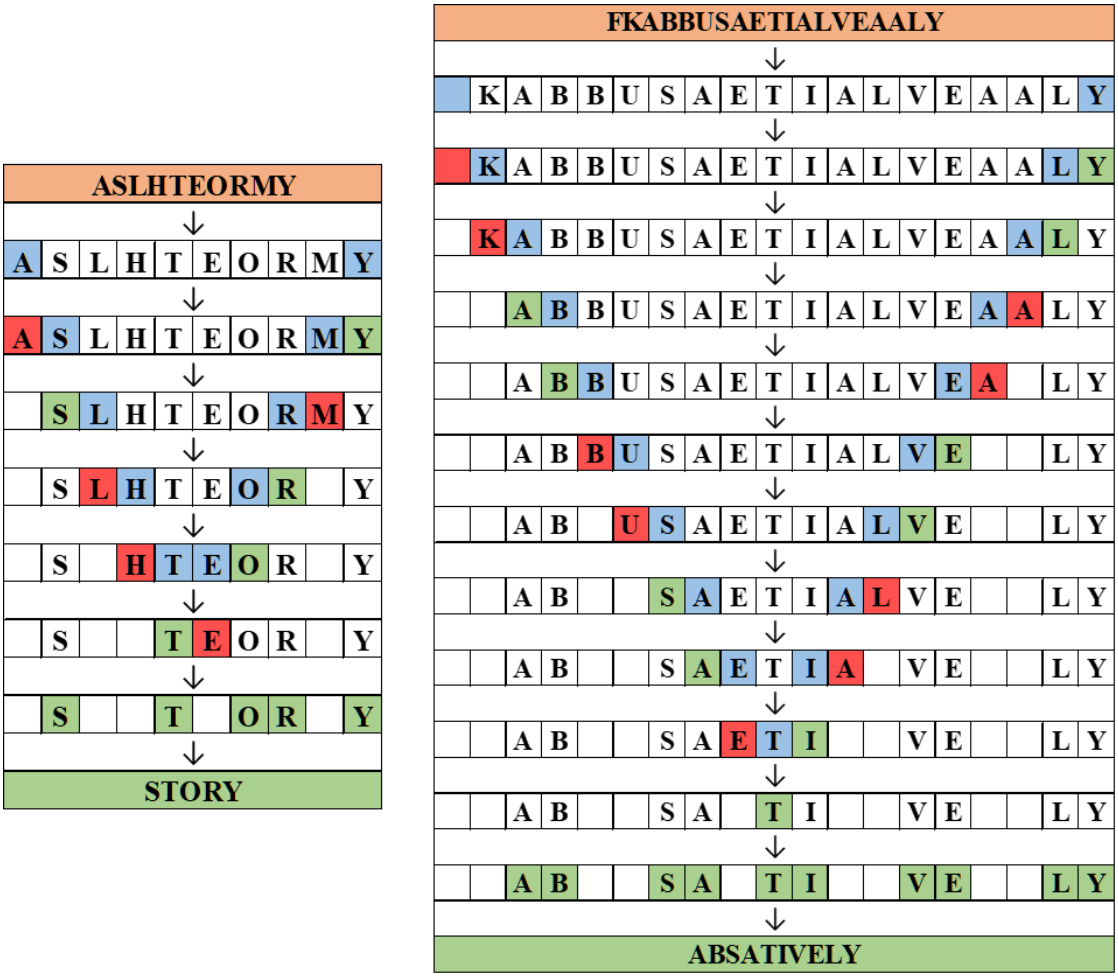

Figure 1: Example Cases