CSE 1321L: Programming and Problem Solving I Lab

Lab 11

Sequence Types (Tuples)

What students will learn:

* Using and manipulating tuples

An interesting feature in Python is the ability of seemingly returning multiple pieces of data from a single method, as per the syntax below:

def myMethod(input1, input2):  
 output1 = input1 + input2  
 output2 = input1 \* input2  
 **return output1, output2**

While the code above makes it seem as if multiple variables are being returned, inspecting the type being returned by the method reveals what is actually going on:

print(myMethod(2, 3)) *# prints (5, 6)*

print(type(myMethod(2, 3))) *# prints “<class 'tuple'>”*

As we can see, a method which attempts to return multiple variables packs them all into a single tuple, in the order that they are returned. We’ve seen something similar to this before, when using the enumerate() method:

name = “Alice”

for position, letter in enumerate(name):

print(“Letter in position “ + str(position) + “ is “ + letter)

The enumerate() method returns two values packed into a tuple: the position of the element being examined and the element itself. Since we are giving the FOR loop two variables to work with (position and letter), it automatically unpacks the outputs for us. However, since the output of enumerate is a tuple, we could have a single variable and still be able to use its output as below:

name = “Alice”

for pair in enumerate(name):  
 print(“Letter in position “ + str(pair[0]) + “ is “ + pair[1])

**Lab11A**: All math, all the time

Write a method **allMath()** which takes in two numbers as inputs and returns a tuple containing the result of each arithmetic operation between both numbers in the following order: addition, subtraction, multiplication, division, floor division, modulus, power. If one of the operations requires a division by 0, replace its result with “None”. For example:

* allMath(2, 3) would return the tuple (5, -1, 6, 0.6666666666666666, 0, 2, 8)
* allMath(1, 8) would return the tuple (9, -7, 8, 0.125, 0, 1, 1)
* allMath(6, 0) would return the tuple (6, 6, 0, None, None, None, 1)
* allMath(7, 8) would return the tuple (15, -1, 56, 0.875, 0, 7, 5764801)

Outside the method, prompt the user for two numbers, pass those two numbers to allMath, and then print out the result, as the sample outputs below show.

Note: You can assume both inputs are valid numbers (i.e. you don’t need to check if the inputs are numbers)

Hint: Remember that, while tuples are immutable, you can concatenate two tuples, much like you would two strings.

Sample outputs (user input in **bold**)**:**

Enter your first number: **5**

Enter your second number: **4**

Your resulting tuple is (9, 1, 20, 1.25, 1, 1, 625)

Enter your first number: **8**

Enter your second number: **0**

Your resulting tuple is (8, 8, 0, None, None, None, 1)

Enter your first number: **239**

Enter your second number: **19**

Your resulting tuple is (258, 220, 4541, 12.578947368421053, 12, 11, 1547248669875101348163600707196216422023050959)

**Lab11B:** Show me where it is

Write a method called **letterPositions()** which takes in two strings and returns a tuple. The first string can be a phrase of any length while the second one will always be a single character. The tuple that the method returns must contain the index of all occurrences of the second string inside the first string. Be sure to not discriminate between upper and lower case (i.e.: a and A should be the same character). If the resulting output is an empty tuple, simply return the empty tuple.

Outside the method, prompt the user for a phrase, a letter, and then print out the resulting tuple as per the sample outputs.

Note: You can assume both inputs are always strings, and that the second input is always a single character.

Sample outputs (user input in **bold**)**:**

Enter your phrase: **bAnAna**

Enter a letter: **a**

a appears inside your phrase in the following positions: (1, 3, 5)

Enter your phrase: **The only thing we have to fear is fear itself!**

Enter a letter: **f**

f appears inside your phrase in the following positions: (26, 34, 44)

Enter your phrase: **What? 9000? There's no way that can be right!**

Enter a letter: **Z**

z appears inside your phrase in the following positions: ()