CSE 1321L: Programming and Problem Solving I Lab

Lab 9

Sequence Types (Part 2)

What students will learn:

- Using and manipulating tuples
- Using and manipulating dictionaries

Content

- Overview
- Lab9A: All math, all the time
- Lab9B: User Authentication

Overview

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 being returned:

```
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 order that they are returned. We have 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 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])
```

Another useful feature in Python are dictionaries. Dictionaries are another type of data structure that we can use to store data, retrieve, update, and delete. Dictionaries work by storing and organizing data as key-value pairs.

Key-value pairs are a way to store and organize data where each data value is represented with a unique identifier or key. This makes it easy to look up and retrieve the data value by just using its identifier.

Think of it as a regular language dictionary, the words are the key, and the definition of a word is the data value.

Dictionaries are defined using curly-braces {} and you can either create a dictionary empty or with some initial values:

```
person = {}

person = {
      “name”: “John”,
      “age”: 25,
      “city”: “Atlanta”
}
```

As you have seen, the definition of a key-value pair defines the key on the left-side, and the value on the right with a colon in between.

You retrieve from a dictionary like a list, call the dictionary and pass the key inside the square-brackets:

```
print(person[“name”]) # prints “John”
```

You can also add a new key-value entry into the dictionary by calling the dictionary and passing the new key inside the square-brackets:

```
# This adds the lastname “doe” into the list
person[“lastname”] = “Doe”
```

To update a value in a dictionary you can use the `update()` built-in dictionary function:

```
# This changes the name value from “John” to “James”
person.update({“name” : “James”})
```

Lastly, to remove an element from a dictionary, you can use the del statement like lists:

```
# This removes the lastname key and value from the list.
del person[“lastname”]
```

As with previous weeks, all labs should have the appropriate file names:

- Lab9A.py
- Lab9B.py

Lastly, make sure you review the sample output and make sure the output of your program follows the exact same format including the input statements, print statement, etc. As always, user input is shown in **red** and **bold**.

Lab9A : 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, and 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)

On the main program, prompt the user for two numbers, pass those numbers to the `allMath()` function, and then print out the result.

Note

- You can assume both inputs are valid numbers (i.e., you do not need to check if the inputs are numbers)
- Remember that while tuples are immutable, you can concatenate two tuples, much like you would with strings.

Sample Output #1:

```
Enter your first number: 5
Enter your second number: 4
Your resulting tuple is (9, 1, 20, 1.25, 1, 1, 625)
```

Sample Output #2:

```
Enter your first number: 8
Enter your second number: 0
Your resulting tuple is (8, 8, 0, None, None, None, 1)
```

Sample Output #3:

```
Enter your first number: 239
Enter your second number: 19
Your resulting tuple is (258, 220, 4541, 12.578947368421053, 12, 11,
154724866987510134816360070719621642202305095 9)
```

Lab9B: User Authentication

Build a program that authenticates the user login by asking the user for a username and a password. For this program, you are going to use a dictionary to store the user's data. This is not ideal but just for this lab we are going to use the user's username as the key and the password as the value for the username key.

Requirements

- The program must use a dictionary that stores the user password using the username as the key.
- The program should be able to hold multiple user login information,
- The program should also feature a "registration" option that asks the user for a username and password and adds it to the dictionary.
- If the login fails by either incorrect username, incorrect password, or if the username does not exist in the dictionary, the program should output "Incorrect Username/Password"
- If a login is successful, the program should show a different set of options: log out, change password, and exit
- If the user chooses to log out, the program should go back to asking for login, register, or exit.
- If the user chooses to change password, input the new password and update it in the dictionary.
- The program stops if the user chooses to terminate it.
- Do not worry with input validation or option validation, assume the user will always follow the correct program flow and input the correct options.

Sample output #1:

```
Choose an option
1 - Login
2 - Register
E - Exit
1

[Login]
Username: jdoe01
Password: 4321
Incorrect username/password!

Choose an option
1 - Login
2 - Register
E - Exit
2

[Register]
Username: jdoe01
Password: 4321
User successfully added!

Choose an option
1 - Login
2 - Register
E - Exit
```

```
1

[Login]
Username: jdoe00
Password: 4321
Incorrect username/password!

Choose an option
1 - Login
2 - Register
E - Exit
1

[Login]
Username: jdoe01
Password: 1234
Incorrect username/password!

Choose an option
1 - Login
2 - Register
E - Exit
1

[Login]
Username: jdoe01
Password: 4321
Success!

Choose an option
3 - Change Password
4 - Logout
E - Exit
4

Logging Out...

Choose an option
1 - Login
2 - Register
E - Exit
E

Terminating...
```

Sample Output #2:

```
Choose an option
1 - Login
2 - Register
E - Exit
```

```
2

[Register]
Username: jdoe01
Password: 4321
User successfully added!

Choose an option
1 - Login
2 - Register
E - Exit
2

[Register]
Username: scrappyOwl
Password: ksuksuksu
User successfully added!

Choose an option
1 - Login
2 - Register
E - Exit
1

[Login]
Username: jdoe01
Password: 4321
Success!

Choose an option
3 - Change Password
4 - Logout
E - Exit
3

[Changin password]
Password: 1234

Choose an option
3 - Change Password
4 - Logout
E - Exit
4

Logging Out...

Choose an option
1 - Login
2 - Register
E - Exit
```

```
1

[Login]
Username: jdoe01
Password: 4321
Incorrect username/password!

Choose an option
1 - Login
2 - Register
E - Exit
1

[Login]
Username: jdoe01
Password: 1234
Success!

Choose an option
3 - Change Password
4 - Logout
E - Exit
4

Logging Out...

Choose an option
1 - Login
2 - Register
E - Exit
1

[Login]
Username: scrappyOwl
Password: ksuksuksu
Success!

Choose an option
3 - Change Password
4 - Logout
E - Exit
E

Terminating...
```

Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.

- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
    - Lab9A.py
    - Lab9B.py
- Programs (source code files) must be uploaded to Gradescope by the due date.