Lab 2 – Data Structure

Recall that during the first half (Lab 1) you focus on learning the basics of Python programming. The problems are not algorithmically challenging, instead just ways to flex your Python muscles.

Today, in the second half (Lab 2), we will focus more on data structures (i.e. list, tuple, set and dictionary) to solve some interesting problems.

Data Structure (mutable vs immutable)

1. List
   A list in Python is resizable and can contain elements of different types.
   In your Python interactive interpreter, type the follow lines. Try to understand how to access items in a list.

   - ```
     s = [5, 6, 7]   # Create a list
     print(s, s[2])
     print(s[-2])
     s[1] = 'foo'    # List can contain elements of different types (Recall: type())
     print(s)
     s.append('bar') # What happen to the list when the following lines are executed?
     xs = s.pop()
     ```

   - ```
     s = list(range(5))
     print(s)
     print(s[:])
     print(s[1:3])
     print(s[:3])
     print(s[3:])
     ```

   - ```
     s = 5* [10]
     s[0] += 5
     print(s)
     ```

   - ```
     s = [''] * 3
     s[2] += 'a'
     print(s)
     ```

   - ```
     s = [[]] * 4
     s[1] += [1]
     print(s)
     ```

   Try replacing the second-to-last line with

   ```
   s[0] = [4, 1]
   ```

   Why is this happening? Hint: Consider using the 'id' function to investigate.

2. Tuples
   Tuples are almost identical to list. Then why do we want to use them? The main difference is that tuples cannot be changed, i.e. you cannot add, delete or change elements from tuples. Keyword: Immutable.
   ```

Let's see a tuple in action! Create a tuple holding three pieces of information of yourself: `myTuple = (Name, Age, Gender)`.

```
>>> myTuple = ('Jane','23','Female')
>>> print(myTuple)
```

Now 'unpack' the tuple, assigning each item of the tuple into a separate variable. A quick way is by doing it this way

```
>>> (Name,Age, Gender) = myTuple
```

What is the alternative to this approach? Think…

Now you feel like keeping your age a secret. Remove the Age item from the tuple.

Try:

```
>>> myTuple.pop(1)
```

Can you use item assignment to modify/remove elements of the tuple? Why?
By now, you know that with list, we can use .pop to remove an element in a list. But when you try to pop on the tuple, Python give a nasty error. So what if you wanted to remove a specific element in a tuple?

Fortunately for us there are two ways to do that:

Creating a new tuple

```
>>> myNewTuple = myTuple[:1] + myTuple[2:]
```

Casting the tuple to a list

```
>>> myList = list(myTuple)
>>> myList.pop(1)
```

So you have changed a tuple to list for modification, can you change it back to a tuple?

Spend some time understanding the methods shown. Do note that distinction between parentheses and brackets!

3. Dictionaries
   Unlike list or tuple, a dictionary is an unordered set of {key:value} pairs, where the keys are unique (within one dictionary). Partnered with these keys are the actual values of the dictionary.

   Example:

```
>>> myDict = {'someItem': 2, 'otherItem': 20}
>>> print(myDict['otherItem'])
```

   You might be thinking can I use index, just like list or tuple

```
>>> myDict[1]
```

You will notice that dictionaries aren't exactly based on index. Python will give you a nasty error for doing that.

What if you would like to add a new {key:values} to your existing dictionary? Adding a {key:value} is really easy. Simply put the key in the brackets and set it equal to the value.

```
>>> myDict['newItem'] = 200
```

To display the variable and value of a dictionary, you must use the key in brackets following the dictionary's name. Try it yourself.

4. Set

Recall set in TR1333 Discrete Mathematic

A set is a collection of objects. The elements of a set can be anything, e.g. integer, string, and even other sets. A set contains an unordered collection of unique and immutable objects.

To create a set, use the set() function

```
>>> mySet1 = {"Postcard","Radio","Telegram"}
>>> mySet2 = set("A Python Tutorial")
```

The second example shows that a string is singularized into its characters to build the resulting mySet2.

You can also pass a list to the set function

```
>>> mySet3 = set(["Perl", "Python", "Java"])
```

How about if you pass a tuple?

```
mySet4 = set(("Apple","Orange","Pear","Pineapple","Apple", "Lemon"))
```

Check the elements in the mySet4, notice any difference with the elements you have defined when creating it?

Now try including lists as elements:

```
>>> mySet5 = set((["Apple","Orange","Pear","Pineapple"],["Perl","Python"]))
```

Why do you get that nasty error? Do find out.

Do explore the Set method, e.g. add(), clear(), remove(), issubset(), .intersection(), on your own.

We don't expect you to finish all of the material here in one class period. If you do - great! But if not, you are encouraged to work through the extra material at your own pace.

Homework

Using what you have learned,

i) create a list of 3-tuples consisting of your name, age and favourite beverage
ii) convert the list of tuple**s** into dictionary
iii) then, try to get the key by the value in the dictionary that you have created.

What to submit

Submit only **ONE** Python file (i.e. just the .py file). The filename should follow the following standard: <StudentMatrixNumber>.py