Lab 6 Manual

Lab 6 Outline

Part 1: Experimenting with Python [35 mins]

- Play with tuples
- Unpack tuples and Nested for loops

Part 2: Quiz [15 mins, Policy Changed!!!]

From this lab on, you can take the quiz any time through the day before 11:59pm to make your life easier. This will still be a **timed (15 mins)**, multiple choice quiz on the lab material. If you completed the lab, you will be able to answer all of the questions on the quiz. We strongly recommend you finish the quiz during the lab section in case you forget to do it.

Play with tuples

1. Create a new file `create_tuple.py`. Copy the following code and paste it to the file:

```
list_example = ["Welcome ", "to ", "CSE", " 8A"]
```

- Define a tuple named `tuple_example` which contains the same elements as `list_example`.
- Print out the length of `tuple_example`, what built-in function will you use to get its length?

- Concatenate all the elements in `tuple_example` into one string `sentence` and print it out.
- We can change the elements in `list_example` by `list_example[3]=" 285"`. Can we do the same thing to `tuple_example` ? Try it out by yourself.

2. Create a new file `singleton_tuple.py`. Copy the following code and paste it to the file. Print out the type of `singleton_tuple` and `singleton`. What difference did you notice?

```
singleton_tuple = ("apple",)
singleton = ("apple")
```

3. Copy the following function definition and paste it to a new file `extract_tuple.py`:

```
def extract_tuple(input_tuple, start_index, end_index):
    return input_tuple[start_index : end_index+1]
```

In the terminal, run function call of `extract_tuple` with different arguments. For example, try `extract_tuple((100,200,300,400,500,600), 1, 4)`.

Unpack tuples and Nested for loops

1. Create a new file `unpack_tuple.py`. Copy the following code and paste it to the file:

```
fruit_tuple = ("apple", "orange")
fruit_x, fruit_y = fruit_tuple
```

What are fruit_x and fruit_y?

Appending the following line to your code. Run your code, what are fruit_x and fruit_y then?

```
(fruit_x, fruit_y) = (fruit_y, fruit_x)
```

2. Create a new file `number_pairs.py.` Write a function `pair_numbers` that takes in two lists of numbers with the same length and returns a list of tuples.

Example:

```
pair_numbers([1],[1]) = [(1,1)]
pair_numbers([1,2],[3,4]) = [(1,3),(1,4),(2,3),(2,4)]
pair_numbers([],[]) = []
```

3. In the same file, write a function `sum_of_product` that takes in a list of tuples and returns the sum of the product of the elements in each tuple. Each tuple always contains two elements.

Example:

```
sum_of_product([(1,2),(3,4)]) = 14
sum_of_product([]) = 0
```