CS215 Chapter4C Lab – Tuples & Arbitrary Argument Lists Name: Jake

***All user-defined functions could be at the TOP of your python file and each function should have a docstring.

Recall that the following functions can take an arbitrary number of inputs.

```
min(88, 75, 92, 97)
max(3, 2, 4)
print("Hi")
print("Hey", "Hi", "Howdy", "Hola", "Bonjour")
```

How does this occur? The answer is via arbitrary argument lists which we explore in this lab. But first you need to learn what a tuple is.

1 TUPLES

1. A Tuple in Python (and even math) is a list of items in which order matters. Here is some examples.

   A 2-tuple of numbers: (0,0)

   A 3 tuple of strings: ("Jacob", "Daniel", "Gadaleta")

   A mixed 4 tuple: (1, "Jake", "Gadaleta", 7)

2. Create a file called Lab4C.py. Include a the top all the needed comments (name, filename, description of file). Then type the following. What is printed out?

```
x = 10
tuple1 = (2, "hi", x)
tuple2 = (3.14, 5**2, 10, 11, 12)

print(tuple1)
print(tuple2)
```

```
(2,"hi",10)
(3.14, 25, 10, 11, 12)
```

3. We use an asterisk to "unpack" the tuple, ie, to see them as a list of individual items. Add the following to your code. What is printed out?

```
print(*tuple1)
print(*tuple2)
```

```
2 hi 10
3.14 25 10 11 12
```

4. Brackets are used to denote lists, which are not the same as tuples. (More on lists next chapter.) We can asterisks to unpack lists too. Guess what is printed below, and then type it to check.

```
yList = [1, 5, 4, 1, 2, 6]
yTuple = (1, 5, 4, 1, 2, 6)
print (yList)
print (yTuple)
print (*yList)
print (*yTuple)
print (yList == yTuple)
```

```
[1, 5, 4, 1, 2, 6]
(1, 5, 4, 1, 2, 6)
1 5 4 1 2 6
1 5 4 1 2 6
False
```

5. One important fact about tuples is that they are immutable. This means you cannot add (append) or remove (delete) items from tuples. It also means you cannot change the values in the tuple. (FYI: There are some tricks that you can perform to make it seem like you can change the values, but you cannot.)

6. Let's Review:
   - [✓ True] or False: When a tuple is printed, the parentheses are printed too.
   - [✓ True] or False: When a list is printed, the brackets are printed too.
   - To unpack a tuple into a list of individual items, put a(n) * ______________ in front of it.
   - True or [✓ False]: A list and a tuple are the same thing.
   - True or [✓ False]: We can add items to an existing tuple.
   - True or [✓ False]: We can delete items from an existing tuple.
   - [✓ True] or False: We can change items in an existing tuple.

2 Arbitrary Argument Lists

- We are going to create a function called average which can take an arbitrary number of values. Type the function below. We will fill in the details.

```
def average(*args):
    "Finds the average of an arbitrary number of values."
    return ???
```

- The term "args" above is just a variable name. The name "args" is used by convention when a function can take in an arbitrary number of inputs. The * before args tells Python to pack the arbitrary number of terms into a **tuple**.

- The sum(…) and len(….) functions are built into Python and can compute the sum and length, respectively, of any iterable item like a tuple. (You can think of an iterable item is just one that you can iterate over in a for loop.) So for example, sum(args) will just sum up all the values in args. Use this information to complete the average function so that it returns the average of an arbitrary list of values.

- Let's test our new function by calling it. Type the values below. What is printed out?

```
print(average(5,8,14))
z = average(-1, 4, 5, -8)
print(z)
average(2, 10)
```

9.0
0.0
____
____

  - Why didn't the average of 2 and 10 print out?

    It was never assinged to anything

  - The average function expects an arbitrary listing of values to run. Let's try and pass it a tuple and see what happens. What is printed below? Recall that you've already defined yTuple in your code.

```
z = average(yTuple)
print(z)
```

TypeError unsupported operand type(s) for +: 'int' and 'tuple'

  - To get around this error, unpack the variable yTuple when you pass it to the average function. How do you this? Hint: What is the unpacking symbol?

- As with tuples, lists will cause an issue too. Add this to your code, run and 0see that you get an error and then fix the error.

```
z = average(yList)
print(z)
```

7. Let’s Review:
   - [✓ True] or False: We will get an error if we pass a list or a tuple as input to a function expecting arbitrary arguments.

8. Create a function called product which takes in an arbitrary number of arguments and returns their product. (HINT: You will need to iterate over the arbitrary args input with a loop to do this.)

9. Test your function by calling it with these values below. Do you get the right answer?

```
print(power(2, 4, 6))
z = power(-1, -1, -1, -3)
print(z)
```

3 RETURNING MULTIPLE VALUES

- Using tuples, we can return multiple values from a function. Here is a silly example.

```
def getNames():
     "Requests a first and last name and returns it."
     first = input("What is your first name? ")
     last = input("What is your last name? ")
     return (first, last)
```

- Test this function by calling with the code below. What is printed out?

```
fullName = getNames()
print(fullName)
print("First:", fullName[0])
print("Last:"fullName[1])
```

('Jake', 'Gadaleta')

First: Jake

Last: Gadaketa

- Update the getNames() function so that it gets the middle name too and outputs the full name and then separately prints out the first name, the middle name, and the last name.

4 TYING IT TOGETHER

- Create a function called getMyStats(…) that takes in an arbitrary number of arguments. This method should calculate the min, max, mean, and median of the list and return these values to the user. Test your function on this data set. Print out what is returned by your function as a whole and then separately (min by itself, max, by itself, etc.).

  98, 97, 55, 82, 73, 88

SUBMISSION INFO

TO GET CREDIT FOR THIS LAB, UPLOAD THESE 2 DOCUMENTS TO THE SUBMISSION AREA.

- Lab4C.py (all functions defined up top; all conventions followed.)
- LabCh4C.pdf (with all blanks filled in)