Module 4 Case introduces Python lists, tuples, and strings. Your assignment is to create mini-Python projects or programs and to run them in the Python-IDLE environment.

Review the videos and read Chapter 11-13 in the online book of "Python 2: For Beginners Only.”

Use Python to run the following .py Files with the original values and statements. Next, change some of the values to the following. Please keep in mind that statements may change accordingly once you modify the programs. The modified programs must execute correctly.

Code is adapted from MITOpenCourseWare, “A gentle introduction to programming using Python” (under the Open Education Consortium Global Network for Open Education and authorized by the creative commons license).

**For Example 1**, change “Hi Class!” to “Good Morning All”

**For Example 2**, change [3, 4, 5, 6] to [1.50, 7, 2.75, 9]

**For Example 3**, change (5, 6, 7, 8) to (100, 101, 102, 103)

Shaw, Z. (2012). Learn Python the hard way. Retrieved from [*http://learnpythonthehardway.org/book/ex32.html*](http://learnpythonthehardway.org/book/ex32.html)

Shaw, Z. (2012). Learn Python the hard way. Retrieved from [*http://learnpythonthehardway.org/book/ex34.html*](http://learnpythonthehardway.org/book/ex34.html)

Shaw, Z. (2012). Learn Python the hard way. Retrieved from [*http://learnpythonthehardway.org/book/ex40.html*](http://learnpythonthehardway.org/book/ex40.html)

Kindy, M. (2008). Chapter 11: Lists. Python 2: For Beginners Only. Edition1.0. Retrieved from [*http://cs118.kindy.net/p2fbo\_20131230.pdf*](http://cs118.kindy.net/p2fbo_20131230.pdf)

Kindy, M. (2008). Chapter 12: Tuples. Python 2: For Beginners Only. Edition1.0. Retrieved from [*http://cs118.kindy.net/p2fbo\_20131230.pdf*](http://cs118.kindy.net/p2fbo_20131230.pdf)

Kindy, M. (2008). Chapter 13: Strings. Python 2: For Beginners Only. Edition1.0. Retrieved from [*http://cs118.kindy.net/p2fbo\_20131230.pdf*](http://cs118.kindy.net/p2fbo_20131230.pdf)

**1 - String Examples.py File**

# Lecture 4

# string\_examples.py

# Strings

# Define a string

new\_string = "Hi Class!"

# Remember we can iterate through it

for letter in new\_string:

    print letter

# We can concatenate two strings together

s1 = "Hi"

s2 = "Class"

print s1 + s2

# but remember, gluing together with a comma adds an extra space

print s1, s2

# and with a comma you can glue together different data types

print s1, 6.189, s2

# We can index the string

print "new\_string[0] is", new\_string[0]

# And slice it

print "new\_string[0:3] is", new\_string[0:3]

# We can get the length of our string using the len function

print "len(new\_string) is:", len(new\_string)

# And use various string methods on it

print "new\_string.upper()", new\_string.upper()

print "new\_string.lower()", new\_string.lower()

**2 - List Examples.py File**

# Lecture 4

# list\_examples.py

# Lists are defined by brackets

new\_list = [3, 4, 5, 6]

print "new\_list is:", new\_list

# Just like strings, we can index & slice them

print "new\_list[2] is:", new\_list[2]

print "new\_list[0:2] is:", new\_list[0:2]

# And iterate through them:

for item in new\_list:

    print item

# Lists, however, are mutable! So, if we want we can change the

# value of one element

new\_list[2] = 100

print "new\_list is:", new\_list

# Or, add on a new element with append:

new\_list.append(87)

print "new\_list is:", new\_list

# Or insert

new\_list.insert(0, 200) # insert at position 0 the element 200

print "new\_list is:", new\_list

# Or even delete an element using remove

new\_list.remove(100) # Write in the item that you want to remove from the list

print "new\_list is:", new\_list

# Lists are possibly the most useful data structure in Python!

# We'll see more about them in lab; check out the documentation on

# list methods for more cool things to do

**3 –** **Tuple Examples.py File**

# Lecture 4

# tuple\_examples.py

# Tuples are immutable and defined by parentheses

new\_tuple = (5, 6, 7, 8)

print "new\_tuple is:", new\_tuple

# We can index them, just like strings

print "new\_tuple[2] is:", new\_tuple[2]

# And iterate through them:

for item in new\_tuple:

    print item

# Even show how long they are

print "Tuple length is:", len(new\_tuple)

# and iterate through indicies

for index in range(len(new\_tuple)):

    print "Index is:", index

    print "Value at that index is:", new\_tuple[index]

# But because they are immutable, we cannot redefine

#  a single element (remember this does work with lists, though)

#new\_tuple[1] = 77 # Returns an error

# We can also do something called \_tuple unpacking\_

(a, b, c, d) = new\_tuple

print "a is:", a

print "b is:", b

print "c is:", c

print "d is:", d

# Make sure that you always have the same number of

# variables when you unpack a tuple!

# Tuples are immutable. To change a tuple, we would need

# to first unpack it, change the values, then repack it:

# Redefine b

b = 77

# Repack the tuple

new\_tuple = (a, b, c, d)

print "new\_tuple is now:", new\_tuple

Once you have executed these programs (**1 - String Example; 2 - List Examples; and 3- Tuple  Examples)**, modify the .py files according to the instructions given to revise the code by program examples. You need to run the IDLE to execute the program changes and review the program results.

You can use the Snipping tools or screen print (ctrl + Print Screen) to show the Pythons editor’s (IDLE) code and results and demonstrate that your program executed correctly.

Create a submission file named as “ITM205-Case 4-Exercises-YourFirstNameLastName “containing executed programs (.py files) for **1 - String Example; 2 - List Examples; and 3- Tuple  Examples.**

Write a summary document in Microsoft Word format named “ITM205-Case 4-Summary-YourFirstNameLastName” to show what you have accomplished.