COM4013 – Introduction to Software Development
Week 13 – Classes and Logic Python

*Always select "**Enable Editing**" if prompted by Microsoft Word*

Lab Exercises

Always refer to the lecture notes for examples and guidance.
Also look at your previous work, many exercises are similar.

There are two "stages" marked in the lab sheet. Stage 1 is the *absolute minimum* point you should reach. Hopefully, you can reach it in the lab session. Reaching stage 2 suggests you are on target for a top-class mark for the module. Set your goals sensibly - do not just aim for the minimum or you may struggle to pass the module.

I use the term integer, string and float, and Boolean to infer what the input/output looks like. Unlike many other languages Python does not expect a data type before the variable names.

So if I ask that you declare an integer num1 to 1, then do as follows:

```
num1 = 1
```

For declaring a string and assigning the value hello

```
greeting = "hello"
```

For floats we can declare it like this

```
Money = 2.50
```

For Booleans (bool) we can declare it like this

```
isHeavy = True
```

- Create a new Jupyter Notebook project called **Lab 13 Exercises**. Refer to Week 1's lecture/worksheet/video if you have forgotten how to create a new project/file/program and using the shell code.

Fahrenheit to Celsius (From semester 1 – Week 2)

- Write a function called CelciusToFahrenheit (returns a float) that convert temperature in Celsius to Fahrenheit using the formula:

  Fahrenheit = Celsius * (9.0f/5.0f) + 32.0f

  Watch the brackets. Display the result as such (25C = 77.00F)
- Test the function in main, by asking the user for the temperature in Celsius
- Make sure the answer is to tow decimal places

A Simple 'while' Loop (From semester 1 – Week 3)

- Write this code into a cell – use the shell code

```
counter = 1
while (counter <= 10):
    print( counter )
    counter+=1
```

- Run the code and see that it works.
- Place commas between each number… i.e., 1, 2, … 9, 10, using the end= print option override

A more complex 'while' Loop (From semester 1 – Week 3)

- Now add a completely new while loop that displays **the even numbers only** from 0 to 100. This will be a very similar piece of code.
  - Be careful about declaring the same variable twice.
  - Assign a new value to the `counter` variable
- Display it on a single line, separated by commas like this…

```
0, 2, 4, 6, 8, 10, 12, 14...
```

- Write a **for loop to reverse the count** from, (inclusive) 100, 98, 96... 58 (inclusive)

Increment, Decrement and Assignment Operators (From semester 1 - Week 3)

- Write the following code into the main function (in a new cell):
- DO NOT COPY AND PASTE IT INTO THE IDE - TYPE IT IN!

```
userNumber = int(input("Enter a number, any number: "))
startNumber = userNumber

print("Now add 1...")
userNumber = userNumber + 1
print(f"You now have { userNumber }... ")

print("Double it...")
userNumber = userNumber * 2
print(f"You now have { userNumber }... ")

print("Now add 4...")
userNumber = userNumber + 4
print(f"You now have { userNumber }... ")

print("Now divide by 2...")
userNumber = userNumber / 2
print(f"You now have { userNumber }... ")

print("Now take away the number you first thought of...")
userNumber = userNumber - startNumber
print(f"You now have three : { userNumber }")
```

- Run the code, it's a simple maths trick - the result is always three.
- Update the code to use operators introduced in the last semester:

```
+=   -=    *=    /=
```

  Make sure that the trick still works after your changes

Create a Class

- Use the code below to create a CDetails class object called myDetails. Change the details to you details using the dot operator.

```
class CDetails:
    mName = None
    mAge  = 0

    def __init__(self, name, age):
        self.mName = name
        self.mAge  = age

   def main():
       # Create an instance of CDetails called myDetails
       myDetails = CDetails(“name”, 18)

   if __name__ == "__main__": main()
```

- Create a second object using the CDetails class. Call it “person2”. This time **initialise** it to the values “Susan” and 17.
- Create a third object using the CDetails class. Call it “person3”. This time **initialise** it to the values “Fred” and 21.
    - Output the values of person 3 in order to demonstrate that it works.

- Assign the contents of “person2” to “person3” using the dot operator, i.e. assign individual data members.
- Output the values of the “person3” variable in order to demonstrate that it works. “person3” should now have the values “Susan” and 17.
- Assign the contents of “myDetails” to “person3”. Do not use the dot operator, instead do this just using the method of assigning one class variable to the other.
- Output the values of the “person3” variable in order to demonstrate that it works. “person3” should now have your name and age.

Employee Records

- Imagine a company with employees. Define a class called CEmployee to store the name, national insurance number, and salary of the employees.
- Write a constructor for the class
- Create one variable to store the data for an employee.
- Enter some data into the class object using the constructor.
- Output the data in the class variable.

- Write a **function** outside of the class to **output the data for a single employee** to the screen. You will need to pass the employee over in the parameter list to the function.
- Demonstrate the function being used.
- Format this output well

- Write a **function** to **read in the data for a single employee**. You will need to pass the employee over in the parameter list to the function.
- Demonstrate the function being used.

Write Data to File

- Write a function that **writes** the contents of a single CEmployee class variable to a file "store.txt".
- Check that the function has worked by examining the contents of the file.
- Write a function that **reads** the contents of a single class objects from the file "store.txt" into a single class variable.

- It is possible to create a List using classes
- Create 4 instances of your class, and add them to the list.
- Create a function to read in the details of 4 people. You should use a *for* loop to implement this. (If you are familiar with the concept of "magic numbers" then ensure that your program does not use magic numbers, i.e. replace the '4' with a suitable constant and use this constant throughout your program.
- Create a function to output the details of all of the people in the List.
- Create a function to write the details of all of the people in the List to file.
- Create a function to read the details of all of the people in the List to file. Note: this is comparatively easy if you assume that there are the details of exactly 4 people in the file. Increase the size of the List to 100. Can you rewrite the function so that it will deal with an unknown number of people in the file.
- See if you can include appropriate validation and error checking as well.

Advanced Tasks – (PYTHON PORTFOLIO PIECE 4)

Logical Calculator

- I want you to implement a console-based Truth Table Generator and/or Logic Calculator.
- Use this week's PowerPoint to help you with figuring out the steps needed to create this tool.
- In logic we have the five basic operators: conjunction, disjunction, implication, biconditional, and negation
- Remember that each option is binary 1 meaning true, and 0 meaning false.
- Let the users know if the complex proposition is a tautology.
- Thing about how you want users to interact with the text-based tool, how many terms you want to allow (begin with one (maybe p) and make your way up.

- Create a menu maybe?

Advanced Tasks – (PYTHON PORTFOLIO PIECE 5)

Task Manager
Create a console-based task manager that allows users to manage their tasks efficiently. The program should provide functionality for adding tasks, marking tasks as complete, listing tasks, and removing tasks.

- Add whatever features you feel are needed.
- A menu as well as responses.
- Users should be able to save their list of tasks to file and reload them on resumption of the tool.
- Ensure that the tool is user centric in its design.

Advanced Tasks – (PYTHON PORTFOLIO PIECE 6)

Wordsearch
Create a wordsearch. Read in two text files and search them for words using your very own (algorithms).

- Copy the text below into two text files and name them accordingly.
- Use a two-dimensional array to store the wordsearch, line by line.
- Store the words in another array.
- Write an algorithm to search for the words in the forwards, backwards, orthogonal directions.
    - I do not expect a binary search algorithm for this or any other searching algorithm – you can make your own by learning to traverse the arrays.
- Print out the words found and their index to screen.

`wordsearch.txt`

```
WSUNSHINESOWANEYF
AEUCEALTPCONEOTNW
TUINYTOLEONTHIEON
EHSTNURALREGOLALI
ROVNCYNRRAINBOWOI
FTUENREEARTHQUAKE
ALTSEROFCLHOPENEL
LEORIRVHEAHOEYRAW
LMIWANIATNUOMDEGF
GRNOEOATETEMAEVLL
NHNANREIEOAYLIIEG
```

BUTTERFLYTSNUHRRE

searchterms.txt

SUNNY
RAINBOW
WATERFALL
BUTTERFLY
EAGLE
MOUNTAIN
FOREST
OCEAN
DAYS
FLOWER
RIVER
LION
EARTHQUAKE
SUNSHINE