COM4013 – Introduction to Software Development
Week 7 –Exam Preview, I/O, and Functions

*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
```

Exam - Preview

After Christmas there will be an exam, which goes towards the final mark for the module. The exam will be mostly multiple choice, with some written answers. The exam will be done on paper, although other arrangements will be made for those who need it.

Below are 5 sample questions to give you a flavour of the exam content. Next week, you will be given another sample test. Attempt these questions now, they should take you no more than 5-10 minutes. Your tutor will give you the answers afterwards.

It is best to first try the exercises without looking at your lecture/lab notes. Then go back and check your answers against your notes and see where you need practice.

1. What is the value of this expression: `4 + 12 / 4`

(a) `7`
(b) `4`
(c) `8`
(d) `This is an invalid expression.`

2. What is the key difference between a `while` loop and a `do-while` loop?

(a) A `do-while` loop executes in reverse.

(b) A `while` loop has one condition, but a `do-while` loop may have several.

(c) A `do-while` loop is guaranteed to execute at least once.

(d) A `do-while` loop calls a function to do it's task.

3. Which condition tests if `lotteryBall` is in the range 1 to 49 inclusive?

[Tip: *inclusive* means including the end values, *exclusive* means not including them]

(a) `lotteryBall <= 1 && lotteryBall >= 49`

(b) `lotteryBall >= 1 && lotteryBall <= 49`

(c) `lotteryBall > 1 || lotteryBall < 49`

(d) `lotteryBall >= 1 || lotteryBall <= 49`

4. What will be displayed on the output from the following code:

```
numbers = [3, 7, 11, 2, 4, 17]
  result = numbers[2] + numbers[4]
  print(result)
```

(a) 9 (b) 15 (c) 19 (d) 23

5. Write a for loop that displays the odd numbers from 79 down to 3 in exactly this format (3 marks for this question):

```
79, 77, 75, ... 7, 5, 3,
```

Functions

- Create a new Jupyter Notebook project called **Lab 7 Exercises**. Refer to Week 1's lecture/worksheet/video if you have forgotten how to create a new project/file/program or use the shell code.

- Add a function called WaitForEnter as shown in the lecture notes on the slides.
    - Make sure you add the function in the correct location, as shown on slides. You should not put the function inside `main`, it should be declared outside, and before the `main` function call.

- Call the function *two times* in the `Main` function to check that your function works and to show how using functions saves repeated code.

- In the same project create a function called `DrawHorizontalLine`
    - Inside of the function there should be a for loop that prints the hyphen character "-" out exactly 80 times.
        - The "-" characters should all be on the same line...
    - Add an empty print statement after the for loop – outside of the for loops scope and looping area.

- Call the `DrawHorizontalLine` function in-between the two calls to `WaitForEnter`.

A Function Returning a Value

- In the same cell as the above task, add a function `ReadFloatFromUser`, as illustrated in the slides.
- However, your version of this function should allow the user to enter an `int` not a `float` and the function should return an `int`. Be careful to make all the appropriate changes for this.
- Rename the function to `ReadIntFromUser`

- Just before your call to `DrawHorizontalLine`, place the text `"How wide do you want the line? "` Into a variable called `message`, then call your new `ReadIntFromUser` function. Put the result of the function into an integer called `lineWidth`. Refer to the example on the slides to avoid making the mistake shown in the slides.
    - We will use this integer `lineWidth` in the next exercise...

A Method with a Parameter

- Now update your `DrawHorizontalLine` method to accept a single `int` parameter called `width`. Refer to lecture slides for examples of this. Now change the horizontal line code so the line is not 80 wide but uses the parameter `width` instead.

- Finally change the *call* to `DrawHorizontalLine`. You now need to pass an integer to this method to say how wide you want the line - pass the variable `lineWidth` that you input from the user.

If you have done everything correctly to here, you should end up with output as shown below. The length of the line should match the user input:

```
Press Enter to continue...
How wide do you want the line? 5
-----
Press Enter to continue...
```

Your main code should be simple and easy to read, and you have a collection of useful, reusable methods. The advantage of methods should be clear.

Calculator Functions

- Copy the calculator code from week 4, and edit the code to implement functions for addition, subtraction, multiplication, and division.
    - If you have not completed this task, then rest assured that you can take a copy of my solution from the Week 4 – Session Practical Solutions file on Moodle.
- Add a modulus operation function to find the remainder of dividing two numbers.
    - Edit the code to allow for the usage of this function

Squared and Cubed Functions

- Create a new cell in Jupyter Notebook (use the shell code).
- Write a function called Squared that return the square value of a number – it should take a parameter called number.
    - i.e., To find the square of a number we multiply it by itself.
    - 3 * 3 = 9, therefore 3 squared is 9
- Write a function called Cubed that returns the cubed value of a number – it should take a parameter called number.
    - i.e., To find the cube of a number we multiply it by itself twice.
    - 3 * 3 * 3 = 18, therefore 3 cubed is 18
- Test the functions above in your main function, you should get the following output.

```
3 squared is: 9
3 squared is: 27
```

- Write a function called CalculatePower. First prompts the user to enter an integer called number, and a second integer called power in the main function. Calculate the result of raising the power to the users input (number) – use number and power as parameters for the function.
    - Raising to the power of two (squared) and raising to the power of three (cubed) have been displayed above – use the logic to help with understanding what is being asked here.
    - In Python the ** operator is used for exponentiation (or raising a number to a specific power). Use this in your function and update your square and cubed functions to use this operator.
    - Figure out the value of 12 raised to the power of 4 (the answer will be released in our next session).

```
12 raised to the power of 4 is: WHAT IS IT
```

Factorial

- Create a new cell in Jupyter Notebook (use the shell code).
- Write a function to figure out the Factorial of a number – You must have a parameter called number to do this.
- The factorial (!) is the product (multiplication) of all numbers up to the number specified.
- 5! (factorial of 5) is 1*2*3*4*5

- Use a range for loop to do this task...
- Test the program on Factorial 5.

```
The Factorial of 5 is: 120
```

- Add Print statements in the for loop to see what is happening in the loop.

File Reading and Writing

- Create a new cell in Jupyter Notebook (use the shell code).
- Create a text file called "numbers.txt" using the (with-open-write("w")-as-file) statement and add the following numbers using the code shown in the slides.
  - 60, 70, 80, 90, and 100
  - Make sure each of these numbers appear on their own line.
  - There is a shorthand method to doing this – think newlines

- Append the following numbers (10, 20, 30, 40, 50) onto the end of the "numbers.txt" file by using the (with-open-append("a")-as-file) statement. Ensure that these new numbers are also placed onto their own line too.

- Read the "numbers.txt" file using the (with-open-read("r")-as-file) statement and read the contents of the entire file into a string variable called `contents`.
- Print the contents variable and make sure that you get the following output

```
60
70
80
90
100
10
20
30
40
50
```

- Add an empty `print` statement after the above code.
- Declare an empty Python List called `numbers`
- Now that you are sure that the contents of your file matches mine, write a (with-open-readtext("rt")-as-file) statement that reads the contents of "numbers.txt" into the `numbers` List.

- Outside of the scope of the above statement, sort the `numbers` List.
- Using the new type of `for` loop shown in the slides `print` the contents of the List onto the screen on a single comma seperated line:
    - To print on a single line, you must `strip()` the newline character from each element in the list.
        - Use the stirp function to do this i.e., `string = string.strip()`
    - You should have the following output on your screen: `10, 100, 20, 30, 40, 50, 60, 70, 80, 90,`
- Place an empty `print` statement after the for loop.

- You may have noticed that the printed-out values are not actually in order. This is because the values within the `numbers` List are not in fact numbers, but strings.

- Place an empty `print` statement after the for loop.

- Use a container iterator for loop to convert each element within the `numbers` List into integer values.
    - You must first declare an empty temporary List container.
    - Iterate through the numbers List as you have previously. Cast and strip all the elements of numbers and append onto the temporary List.
    - Sort the numbers and print the List contents on a single line
        - Note that the strip function will not work here if you copy-paste the code from before – as this is now a List of integers.
    - You should see the following values on your console:
        - `10, 20, 30, 40, 50, 60, 70, 80, 90, 100,`

- **`FOR YOUR ASSIGNMENT YOU MAY WISH TO STORE YOUR VARIABLES INTO SEPERATE TEXT FILES GROUPED BY VARIABLE TYPE.`**

- You may want to turn the code you've written into functions and use them in your assignment as there is a file reading and a file writing task involved (I will not be releasing a solution for this task).

- For all times sake, print the `numbers` List in reverse order onto the console using a for loop with range method.

THAT IS ALL FOR NOW

Submission

1. Upload your work to the GitHub classroom – Here
    a. Classroom Link for Week 7