159.171 - Tutorial 5 – Nested loops

Part 1: Problems for you to try

Exercise A: Using nested for loops to print number pairs

Use nested loops to print the following pairs of numbers:

```
0, 0
0, 1
0, 2
1, 0
1, 1
1, 2
2, 0
2, 1
2, 2
```

Use the outer loop to generate the first number in the pair. Use the inner loop to generate the second number in the pair and print each pair.

Exercise B: Using nested for loops with lists

Use nested loops and these lists to print the following pairs of strings as output:

```
nuts = ["Peanut", "Walnut", "Almond", "Pecan"]

products = ["Butter", "Cake", "Praline", "Pie"]
```

```
Peanut Butter
Peanut Cake
Peanut Praline
Peanut Pie
Walnut Butter
Walnut Cake
Walnut Praline
Walnut Pie
Almond Butter
Almond Cake
Almond Praline
Almond Pie
Pecan Butter
Pecan Cake
Pecan Praline
Pecan Pie
```

Part 2 – Problems for you to submit for marking

Create a new python file (.py extension) for each of the following problems, name each Python file accordingly. Remember to save your work often!

Extramural students: *Submit your .py files for Part 2, one file for each of the following Exercises {A,B} on Stream (follow the 'Tutorial 5 submission' link).*

Internal students: *Your work will be marked during the Tutorial.*

Exercise A: Using nested for loops to print numbers

a) Write a program that prompts for two numbers (*Limit* and *Copies*) and then writes all the numbers from 1 up to and including *Limit, Copies* number of times. Use nested loops.

e.g. if you input 5 as the Limit and 3 for the number of Copies you'll get:

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

b) If you haven't already, break this program up into functions. Write a function

`printRow(startValue, limit)`

that prints out a row of numbers separated by tabs, from startValue to and including limit.

e.g. `printRow(3, 12)` would display:

```
3   4   5   6   7   8   9   10  11  12
```

c) Create another function

`printNumberBlock(startVal, limitVal, numCopies)`

that calls `printRow()` the specified number of times (numCopies).

e.g. When you call `printNumberBlock(3,12,4)` it will produce:

```
3   4   5   6   7   8   9   10  11  12
3   4   5   6   7   8   9   10  11  12
3   4   5   6   7   8   9   10  11  12
3   4   5   6   7   8   9   10  11  12
```

Exercise B: Using nested loops, lists and an if statement

You have two class lists, one with Maths students and the other with Computer Science students.

```
mathStudents = ['Audrey', 'Ben', 'Julia', 'Paul', 'Gerry',
                'Sue', 'Helena', 'Harry', 'Marco', 'Rachel',
                'Tina', 'Mark', 'Jackson']

csStudents = ['William', 'Melissa', 'Sue', 'Ben', 'Audrey',
              'Susan', 'Mark', 'Hemi', 'Brendan',
              'Paul', 'Barry', 'Julia']
```

Use these lists and nested loops to find out

- which students are enrolled in both classes and
- how many there are (count the number in both classes).

Your output should look like the following:

```
There are 13 students in the Maths class.
There are 12 students in the CompSci class.

Student:  Audrey is enrolled in both classes
Student:  Ben is enrolled in both classes
Student:  Julia is enrolled in both classes
Student:  Paul is enrolled in both classes
Student:  Sue is enrolled in both classes
Student:  Mark is enrolled in both classes

6 students are enrolled in Computer Science and Maths
```

For the first two lines, DO NOT just print out the fixed strings:

```
      print("There are 13 students in the Maths class.")
      print("There are 12 students in the CompSci class.")
```

Instead, use Python to find the length of each list and then print its value. This will ensure your code still works if you add or delete names from the lists.

Part 3 – Extra problems for you to try

a) Write a Python program that will print the following:

```
10
11 12
13 14 15
16 17 18 19
20 21 22 23 24
25 26 27 28 29 30
31 32 33 34 35 36 37
38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54
```

There are (at least) two ways to do this:

1. using one for loop that produces values from 10 to 54, AND an integer variable that controls when to skip to a new line.
2. Using two loops, one inside the other. The outer one produces values 10, 11, 13, 16 etc. and the inner one controls an increment value.

To build each row, you can either

1. **use a separate string variable to build the row, one number at a time**
   e.g.
   ```
   row = ""              # before looping
   row = row + str(x)    # inside for, where x is the new number
   print(row)            # when loop has finished
   ```
2. **use a print() statement, but stop it outputting a new line**
   by adding the parameter **end = " "**
   e.g.
   ```
   print(12, end=" ")    # inside the loop
   print(13, end=" ")
   print()               # when loop has finished
   ```

   would result in 12 and 13 being displayed on the same line, with a space between them.

b) Create a big box out of *n* rows of little o's for any desired size n. Use an input statement to allow the user to enter the value for n and then print the properly sized box.

```
E.g. n = 3
oooooo
o    o
oooooo
```

```
E.g. n = 8
oooooooooooooooo
o              o
o              o
o              o
o              o
o              o
o              o
oooooooooooooooo
```