**CS5003. Lab Work 04_B – *meanwhile* – While Loops and Functions on Strings**
**For Lab Session 05 on 2022_0216 (due at next Lab session)**

Many Lab Assignments include both Coding Practices (submitted on Gradescope) and additional Lab Work (submitted on Canvas). This second part of recitation is intended to feel a *bit* more challenging than Coding Practice, but *not* as challenging as a full-blown Homework Assignment. Like Coding Practices, and unlike Homework Assignments, working with your Lab Partner for these is strongly encouraged.

For this assignment, please create a Python program in a single file, called **meanwhile.py**, starting from our standard **template.py**. Be sure to include a top of file documentation string and the end of file "magic incantation" to invoke your main() function only when appropriate. Your file should contain three functions:

- **print_lower**(string)
  - *prints* the string *in all lowercase* (***without*** *invoking the built-in string.lower() method)*
  - Characters that are not in the range **a-z** or **A-Z** should be printed without modification
  - Remember to include "top of function" documentation
- **print_upper**(string)
  - *prints* the string, *in all uppercase*, *(**without** invoking the built-in string.upper() method)*
  - Characters that are not in the range **a-z** or **A-Z** should be printed without modification
  - Remember to include "top of function" documentation
- **main()**
  - Tests the above functions, on three examples each, including the string 'Hello, world!'

    >>> print_lower('Hello, world!')
    hello, world!
    >>> print_upper('Hello, world!')
    HELLO, WORLD!

**Hints**:

- You can access each individual character of a string like this:

```
>>> string1 = 'Hello, world!'
>>> string1[0]
    'H'
>>> string1[1]
    'e'
>>> string1[12]
    '!'
```

- To check the length of a string, use:

```
>>> len(string1)
13
```

- Each function should use a **while** loop to operate on each "character" of the string. *(In Python, there isn't actually a separate datatype called **character**; but a string of length 1 is equivalent to our usual notion of a character.)*

- To print one character at a time, on the same line, with no spaces or newlines:

```
>>> print(string1[0], sep="", end="")
```

- Since computers ultimately treat everything as a number, you need to know how to convert a 1-letter string to its corresponding numeric code, and vice versa. Python uses an extended version of the ASCII code (https://www.ascii-code.com/), in which 'A' is represented by decimal 65, 'Z' is represented by 90, 'a' is represented by 97, and 'z' is represented by 122. The function **ord**(char) returns the decimal number for a character, and the function **chr**(int) returns the character for that number, as seen below:

```
>>> ord('a')
97
>>> chr(97)
'a'
>>> ord('z')
122
>>> chr(122)
'z'
>>> ord('A')
65
>>> chr(65)
'A'
>>> chr(65 + 32)
'a'
```

- Uppercase letters are in the range 65 – 90, inclusive

- Lowercase letters are in the range 97 – 122, inclusive

- Some of you may be aware of more advanced ways to accomplish this task. Please use only what has been covered so far in class, plus the Tips provided here.