# EABC Summer 2023
# Python 3: Loops and Functions in Python

**Recall the guidelines for team activities:**

1. You should work as a team; **all** team members will be held responsible for all material. You may work together and contribute to one program and submit similar codes as long as the contributors to the development of the solution are documented.
2. Each team is responsible for submitting their own team assignment to Gradescope.

## Task 1 (of 2):

**Learning Objective:** Predict the output of a complete `for` and `while` loop in Python.

**Background:** Another common mathematical series is a factorial, which is often used in figuring out combinations of events. For example, the number of ways to arrange the letters A, B, C without repeating the letters is determined using a factorial. The factorial of a positive integer, n, is defined as:

$$n! = n * (n-1) * (n-2) * (n-3) \ldots .3.2.1$$

(By convention, 0! = 1 and the factorial calculation cannot be performed with a negative number) Therefore, the three letters A, B, and C can be arranged in 3! = 3*2*1 = 6 ways; specifically, the arrangements are [ABC], [BAC], [BCA], [CBA], [CAB], and [ACB]. As n increases it becomes difficult to determine these permutations. By using a computer and loop structure, however, computing the factorial becomes a snap.

**Task Instructions**

Create a flow diagram of an algorithm programmed as a sub-process that will calculate the factorial of a number passed as an argument. The function should return an error flag value (such as -999) if the number passed is negative, since the factorial function can only be calculated for non-negative integers. Save the flow chart in a separate page in the PDF file.

Next, write a Python program that defines a function `MyFactorial` to compute the factorial of a number using your flow chart.

**Example 1 with 5 passed as the value for n:**

```
The Factorial of 5 is 120.
```

**Example 2 with -5 passed as a value of n:**

```
Error 999 [Negative input].
```

Save your main program as: *`Py3_Task1_teamnumber.py`*

**Deliverables:** *`Py3_Task1_teamnumber.py`*

## Task 2 (of 2):

**Background**
Convolution is an important operation in signal and image processing that is used to modify a given signal or image (e.g., smoothing, sharpening, enhancing, etc.). To compute a convolution, you need a signal and a filter each expressed as an array. Below is an example of 1D convolution.

Each element of the convolution array is calculated by the following summation:

$$h_i = (f * g)_i = \sum_{j=0}^{m-1} g_j \cdot f_{(i+j)}$$

Where *f* is the original signal array, $g$ is the filter array, m is the length of the filter array, i is an index for the elements in f, and j is an index for the elements in $g$. The first element in each array has an index of 0.

(*Note: One of the issues that arises with convolution is dealing with "Edge Handling," the cases where indices extend beyond the length of the signal array. One solution is to not do any calculations where this would happen, which makes the convolution array smaller than the signal array.)*

If you are given a signal as:

$$f = [10,20,60,34,23,14]$$

And a filter as:

$$g = [\frac{1}{2}, \frac{1}{2}]$$

The elements of the output after the convolution can be calculated as:

$$h_0 = (f * g)_0 = (g_0 \cdot f_0) + (g_1 \cdot f_1) = (\frac{1}{2} \cdot 10) + (\frac{1}{2} \cdot 20) = 15$$

$$h_1 = (f * g)_1 = (g_0 \cdot f_1) + (g_1 \cdot f_2) = (\frac{1}{2} \cdot 20) + (\frac{1}{2} \cdot 60) = 40$$

...

$$h_6 = (f * g)_6 = (g_0 \cdot f_6) + (g_1 \cdot f_7) = (\frac{1}{2} \cdot 14) + (\frac{1}{2} \cdot ?)$$

Then the entire output should be:

$$h = [15, 40, 47, 28.5, 18.5]$$

**Task Instructions:**

In this problem you will need to first generate a signal with 1000 elements using the following formula:

$$f[i] = \sin^2\left(\frac{i}{500} * \pi\right) \; i = 0,1,2 \ldots$$

Then, you will need to apply the following filter to this signal and output the final results.

$$g = [\frac{1}{16}, \frac{1}{4}, -\frac{3}{8}, \frac{1}{4}, \frac{1}{16}]$$

Please use NumPy to do the calculations.
Output: Print the first 10 elements of *h* to the console.

*Optional:*
*Try to plot the original signal and the signal after the convolution process in a single plot using matplotlib.*

**Deliverables:**
`Py3_Team_teamnumber.pdf`
`Py3_Task2_teamnumber.py`