Week03.3 In-Class Team Activity

Saturday April 22nd, 2023 - upload PDF with photograph (either rendered or a separate file) Due by 11:59pm

If you missed class – no credit for this in-class activity, instead, I drop two unexcused absences from the Monday/Wednesday activities before deductions begin.

- Please identify your team & the names of your teammates.
- Please insert a team photo somewhere in this notebook. If you cannot get the to render correctly to PDF please upload a separate file of the photo

Names: Blair Lee, Aryan Sunkersett, Warren Wu

In [ ]:
```
from IPython.display import display
from PIL import Image

display(Image.open("/Users/blairlee/Desktop/Stats 21/IMG_4735.jpeg"))
```

Please try to write your own factorial function without the use of recursion

So you have seen a factorial function programmed to illustrate the use of recursion, but can your team program a version without recursion using the cell below?

Start by discussing as a team the features of the recursive function (e.g., a base case that returns a special value). Then try programming a solution. If your team can't do it, it is OK, just note how far you were able to get before running into problems.

In [ ]:
```
# please try to write a factorial function without recursion
def factorial(n):
    if n <= 0:
        return 1
    else:
        product = n
        next = n - 1
        while (next > 0):
            product = product * next
            next = next - 1
        return product

print(factorial(5))
print(factorial(0))
```

```
120
1
```

Please try the sum of numbers 1 to n with recursion

Suppose your function is named `we_tried_it`. Can you make it work so something such as:

`we_tried_it(10)` or `we_tried_it(n = 10)`

prints out:

`The result is: 55`

Yes, I know sum(range(1, 11)) will result in 55 but this is for the exercise of writing a recursive function in Python.

In [ ]:
```
# please choose any number that fits within your resources
def we_tried_it(n):
    if n == 1:
        return 1
    else:
        return n + we_tried_it(n - 1)

we_tried_it(10)
```

Out[ ]:
```
55
```

(optional) How about using recursion and generating a Fibonacci sequence?

The Fibonacci sequence is a sequence in which each number is the sum of the two preceding numbers usually starting with zero and one.

$F_0 = 0, \quad F_1 = 1,$

and

$F_n = F_{n-1} + F_{n-2}$ for n > 1.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34

So maybe that is the result of Fibonacci(10)

In [ ]:
```python
# please choose any number that fits within your resources
def Fibonacci(n):
    f0 = 0
    f1 = 1
    fn = 1
    seq = []
    seq.append(f0)
    seq.append(f1)
    while (n > 2):
        fn = f0 + f1
        seq.append(fn)
        f0 = f1
        f1 = fn
        n = n - 1
    return seq

# Not sure if this counts as recursion as the function
# is not called within the function, but gives a sequence of Fib numbers.

Fibonacci(10)
```

Out[ ]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```