CS 315 - Day 05, Sequences and Series

What are the odds that two people in this room have the same birthday?

https://www.scientificamerican.com/article/math-explains-likely-long-shots-miracles-and-winning-the-lottery/

Someone applied this to English Premier League teams!

To derive a formula, let's first consider the same question, but with birth ***months*** instead of birth***days***.

Let's start with four students. How many different ways can they have birth months?

Jan-Jan-Jan-Jan
Jan-Jan-Jan-Feb

…

Nov-Dec-Dec-Dec
Dec-Dec-Dec-Dec

We have 4 "slots" and 12 choices for each slot, so $12^4 = 20{,}736$

Of these 20,736 possibilities, how many have repeated birth months? In other words, how many have two or more students having the same birth months?

This question might be easier to answer if we consider the number of ways no students have the same birth month, and subtract it from the total.

There are 12 choices of birth month for the first student, 11 for the second, and so on…

$$12 \cdot 11 \cdot 10 \cdot 9 = P(12{,}4) = \frac{12!}{(12-4)!} = 11{,}880$$

This is an r-Permutation.

So, the number of ways two or more students can have the same birth month is $20{,}736 - 11{,}880 = 8{,}856$.

To calculate the ***probability*** that at least two of four students have the same birth month, we simply divide the ***number*** of ways they can have the same birth month by the total, $\frac{8{,}856}{20{,}736} = 0.421$, so it's about a 42% chance.

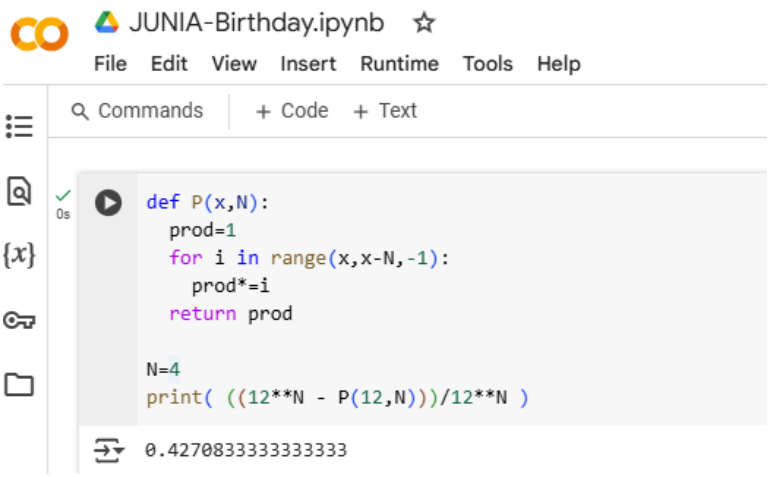
Let's try this out.

To generalize, what would the equation be to calculate the probability that given N students, at least two have the same birth month?

$$\frac{12^N - P(12, N)}{12^N} = \frac{12^N - [12 \cdot 11 \cdot 10 \cdot \cdots \cdot (12 - N)]}{12^N}$$

Implement this in a Jupyter Notebook (we've used Google Colab for this previously, and we can save our work to GitHub).

A straightforward solution for one value of N is:



While we don't have an analytical solution, we could simply loop over N. Modify your Jupyter Notebook to do this.

```
[15] def P(x,N):
       prod=1
       for i in range(x,x-N,-1):
         prod*=i
       return prod

     for N in range(2,12):
       print( N , ((12**N - P(12,N)))/12**N )
```

```
2 0.08333333333333333
3 0.2361111111111111
4 0.4270833333333333
5 0.6180555555555556
6 0.7771990740740741
7 0.8885995370370371
8 0.9535831404320988
9 0.9845277134773662
10 0.9961319283693416
11 0.9993553213948903
```

So far, we've done a lot of work, but we haven't answered the question of the probability of two or more people in the room having the same birth ***day***.

Well, it's an easy modification, 12 months→365 days

$$\frac{365^N - P(365, N)}{365^N}$$

And the update to the Jupyter Notebook should be fairly straightforward. I suggest copying your existing code into a new Execution group and editing, to save your previously working code.

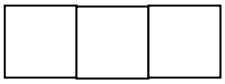
```
def P(x,N):
  prod=1
  for i in range(x,x-N,-1):
    prod*=i
  return prod

for N in range(23,32):
  print( N , ((365**N - P(365,N)))/365**N )
```

```
23 0.5072972343239854
24 0.5383442579145288
25 0.5686997039694639
26 0.5982408201359389
27 0.626859282263242
28 0.6544614723423994
29 0.680968537477777
30 0.7063162427192686
31 0.7304546337286438
```

Let's do a quick review of Sequences and then Series. First, consider rows of square boxes.

A row with one box has 4 sides,  so $a_1 = 4$

A row with two boxes has 7 sides,  so $a_2 = 7$

A row with three boxes has 10 sides,  so $a_3 = 10$

Find an <u>explicit</u> (not a recursive) formula to describe this, which is a function of n, where n is the number of boxes and a_n is the number of sides. Be sure to ***<u>quantify</u>*** n.

$a_n =$

There are infinitely many correct answers, two are: $a_n = 3n + 1\,, \forall\ int\ n \geq 1$ or $a_n = 3n + 4\,, \forall\ int\ n \geq 0$

The only difference in these two explicit formulas are where the subscripts on the variable a begin, 0 or 1.

Sometimes we are simply given a sequence in roster notation: $4, 7, 10, 13, 16, \ldots$
The "dot, dot, dot" is called an "ellipsis."

Find a recursive (***NOT*** explicit) formula to describe this, which is a function of n, where n is the number of boxes and a_n is the number of sides. Be sure to give an initial value for the sequence.

$a_1 =$

$a_n =$

==========

$a_1 = 4$ again, there are infinitely many correct answers: $a_0 = 4$

$a_n = a_{n-1} + 3\,, \forall\ int\ n\ \geq 2\,.$ $a_n = a_{n-1} + 3\,, \forall\ int\ n\ \geq 1\,.$

This is recursive because the definition for a_n depends on previous value in the sequence, a_{n-1}.

Why does the quantification of n start at 2? ***Because in the subscript, n-1, if n was only 1, the subscript would be 0***

In many ways, the recursive definitions are easier, or more natural, to find.

We will also consider sums of numbers, which could defined by explicit/direct formulas, or given in roster format.

First, we will just focus on shortcuts to writing these sums, ***NOT*** yet calculating the value of these sums.

Consider the sum $1 + 2 + \cdots + 10$. Without considering what this sum is (it's 55, btw, but we'll get to that next class), we can write it in ***sigma*** or summation notation, which is like a nickname ("Gerald" ←→"Jerry").

$$1 + 2 + \cdots + 10 \quad = \quad \sum_{k=1}^{10} k$$

We have a ***loop control variable*** (or "dummy" variable) which is as if we were writing a loop in a Python program, here it is k.

We start at 1

and

end at 10.

If we were writing a formula for the ***sequence*** $1, 2, 3 \cdots, 10, a_k = k, for\ int\ 1 \leq k \leq 10$, and we use this explicit/direct formula in the sigma notation.

Again, $\sum_{k=1}^{10} k$ does not tell us anything about the what the sum of the numbers is, it's just a nickname for

$1 + 2 + \cdots + 10$

By the way, a loop in Python, using these variables, and introducing a variable to keep the running sum, would be:

```
sum = 0
for k in range (1,11):
   sum += k
```

Sometime we parameterize the final term in the summation, it goes to n rather than a particular number like 10 above.

This is actually just a simple change:

$$1 + 2 + \cdots + \mathbf{n} \quad = \quad \sum_{k=1}^{\mathbf{n}} k$$

This can be confusing, because now we have an expression with two variables, n and k. Sometimes it helps to refer back to this original example, and swap 10 for n. Note that the roster type notation is called ***expanded form***.

Sequences and series are the main mathematical tools we will use in class to determine the efficiency of algorithms.