In [1]:
```
# Styling notebook
from IPython.core.display import HTML
def css_styling():
    styles = open("./styles/custom.css", "r").read()
    return HTML(styles)
css_styling()
```

Out[1]:

Sequences

a_n for an *arithmetic* sequence starting with "aStart" and difference d

We need to specify the index of aStart to make it work

In [1]:
```
def arith(aStart,d,n,start):
    assert start <= n, "Can't do that backwards"
    # for start = 0 it's just aStart + n*d
    # So, adjust for other values of "start"
    # 0: aStart + n*d
    # 1: aStart + (n-1)*d
    # --> General:
    return aStart + (n - start)*d
```

In [21]:
```
print(arith(7,1,1000,5)) # a_5 = 7, d = 1   - what is a_1000?
```

```
1002
```

Same just for a *geometric* sequence with r

In [9]:
```
def geom(aStart,r,n,start):
    assert start <= n, "Can't do that backwards"
    # for start = 0 it's just aStart*(r^n)
    # So, adjust for other values of "start"
    # 0: aStart*(r^n)
    # 1: aStart*(r^(n-1))
    # --> General:
    return aStart*(r**(n-start)) # ** is power in Python
```

In [20]:
```
print(geom(1,2,4,0))   #a_0 = 1, r = 2   - what is a_4?
print(geom(1,2,4,1))   #a_1 = 1, r = 2   - what is a_4?
print(geom(1,0.5,4,0)) #a_0 = 1, r = 0.5 - what is a_4?
```

```
16
8
0.0625
```

Based on the idea of successive (shorter) sequences of differences of the original sequence

See Mathologer: *Why don't they teach Newton's calculus of 'What comes next?'*
https://www.youtube.com/watch?v=4AuV93LOPcE (https://www.youtube.com/watch?v=4AuV93LOPcE)

Stopping on a constant sequence which can be tested, of course (-> allEqual)

A quote from Mathologer is important (video at 20:07)

The whole 'What's Next' game is fundamentally very silly. If anything can be an answer then of course nothing is an answer

Oh, and stop watching at 22:06 This is higher-end math, enormously pretty, but not really relevant for this course...

In [18]:
```
from functools import reduce

def extend(l) :
    assert len(l) > 0, "Can't continue empty list"
    if allEqual(l) :
        result = l.copy()
        result.append(l[0])
        return result
    lenL = len(l)

    # len(l) > 1 from here

    # a. build list of successive differences
    diffList = []
    for index in range(1,lenL) :
        diffList.append(l[index] - l[index-1])
    # b. extend this by 1 (RECURSION!)
    diffList = extend(diffList)

    # c. use the last element of the extended
    #    difference list to extend the original
    result = l.copy()
    result.append(l[lenL - 1] + diffList[lenL - 1])
    return result

# If you grew up with C, C++ or Java: Good luck figuring this one out ;-)
def allEqual(l) :
    return reduce(lambda a,b: a and (b == l[0]),l,True)
```

In [19]:
```
print(extend([0,1,2,3,4,5]))
print(extend([0,1,2,3,4,5,-300])) # That's cute!
print(extend([0,1,4,9,16,25]))
print(extend([0,1,8,27,64,125]))

# video at 21:11
print(extend([1,2,4,8,16]))
print(extend([1,2,4,8,16,31]))
```

```
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, -300, -2135]
[0, 1, 4, 9, 16, 25, 36]
[0, 1, 8, 27, 64, 125, 216]
[1, 2, 4, 8, 16, 31]
[1, 2, 4, 8, 16, 31, 57]
```