Scientific Computing in Python 2 – Deterministic for loop based modelling.

In the last practical we worked on creating functions that we can put values into and it will mathematically calculate an output for us.

These functions are the basis of much of the modelling we will be doing.

At the start of a model we specify how things change over time using a function with the inputs that will be our starting parameters or the constants in the model.

We can perform simple models by iterating parameters through a for loop over the length of time you want the model to run.

Or we could even use a while loop and loop it until some certain criteria is met by the model e.g. keep running it until our species of animal is extinct or the model has reached stability.

First of all, we should refresh our memories on how for loops work in python with a few examples.

```
for i in range(0,10):
        print("The current iteration is", i)
```

or how about a while loop?

```
Import random
X = 1
while x < 100:
        x = randint(0,110)
        print(x)
```

This just runs the loop until the random number is above 100.

These simple loops are the same principals of doing a basic model.

Let's throw our minds back to our bacteria that is doubling every time step. This is a fairly simple model

Ok let's run a simple model that tells us how many bacteria there would be after 100 times steps.

```
x = 1
for i in range(0,100):
        x = x * 2
        print(x)
```

but we could do this more efficiently using a function:

```
def bacteria(starting, timesteps ):
        x = starting
        for i in range(timesteps):
                x = x*2
                print(x)
bacteria(1,100)
```

We can now specify any starting number of bacteria and the number of timesteps we want to run the model for.

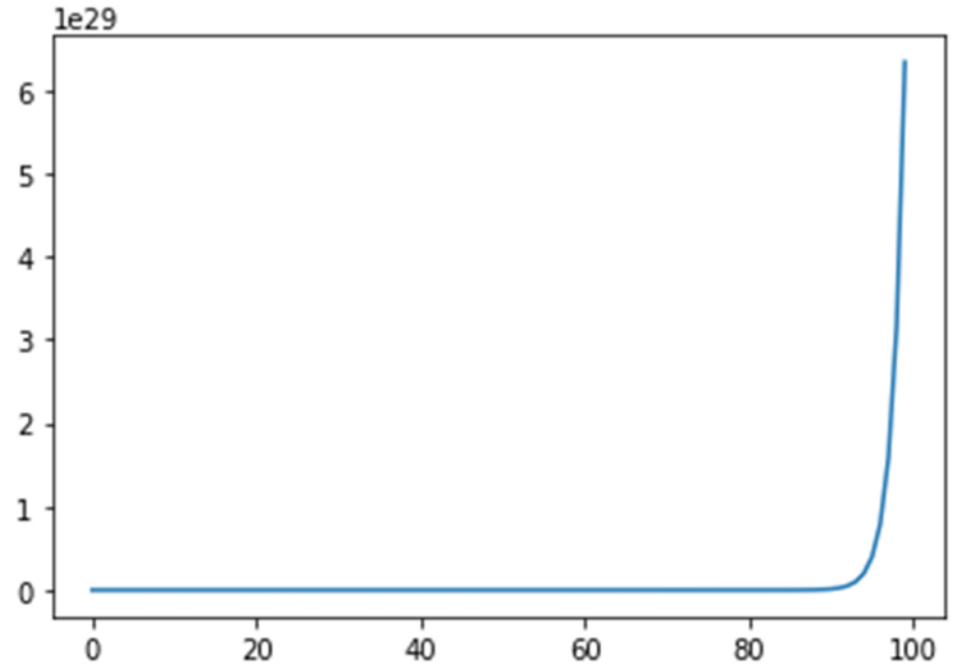
But just getting it to print out the number is not the most useful thing for when we are doing analysis.

Try for yourself to save the timestep in one column and the bacteria number in another of a data frame.

Hint here you want to use pandas to create a dataframe or list at the start and then fill it in row by row during the for loop.

Now use matplotlib to make a plot of with the number of bacteria on the y axis and the timestep on the x.

```
Import matplotlib as plt
plt.pyplot.plot(x, y)
```



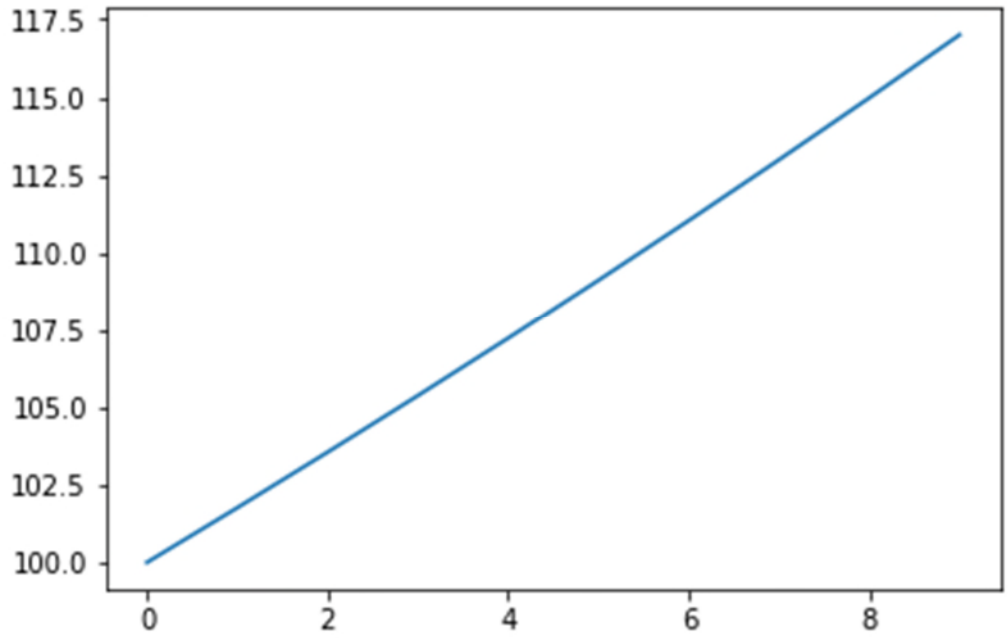
This is the default plot you will get because it is treating everything in terms of x10^29 which is the final size of the population.

But a doubling populations isn't very realistic.

Lets build our own for loop model for a population that has a growth rate and a mortality.

Let's think about a population of 100 red deer that has a breeding season in the summer where there is a population growth of 6% but in the winter there is a culling of the population that reduces it by 4%.

```
def population(starting, growth, mortality, years):
    n = []
    n.append(starting)
    x = []
    x.append(0)
    grwth = 1 + 1 * (growth/100)
    mort = 1 - 1*(mortality/100)
    for i in range(1,years):
        x.append(i)
        n.append((n[i-1]*grwth)*mort)
    plt.plot(x,n)


population(100,6,4,10)
```



The code for this model defines a function where you input the starting population, the summer growth rate, the winter culling rate and the number of years you want the model to run for. It then plots this.

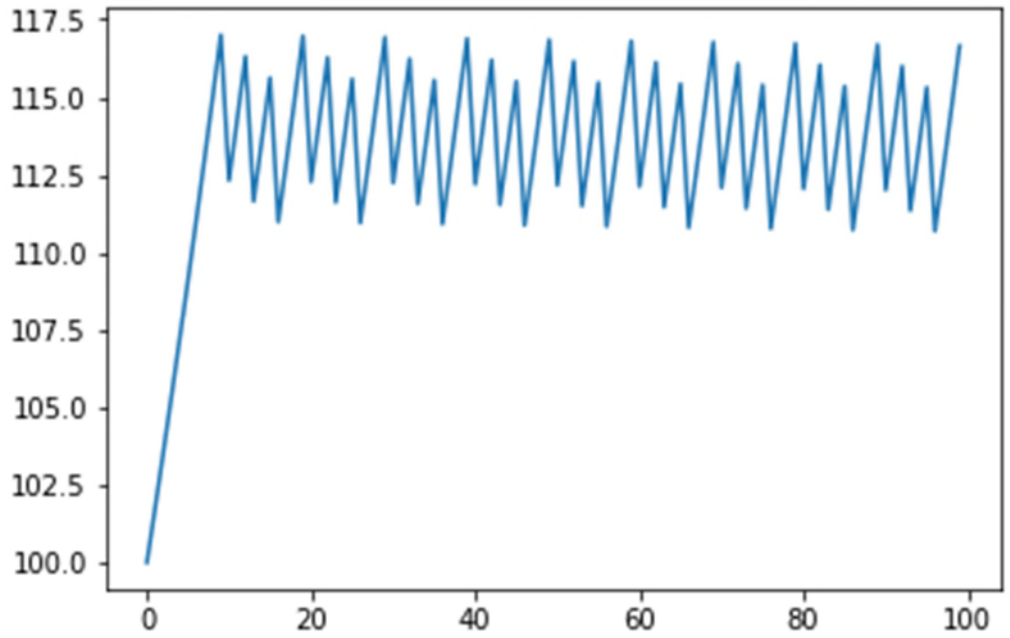
If we wanted we could use pandas to save the values as a dataframe and then we can export that data frame to a csv.

This model is extremely simple and is straight linear growth population.

How about if we add a carrying capacity of 115 individuals in the population.

```
n = []
n.append(starting)
x = []
x.append(0)
grwth = 1 + 1 * (growth/100)
mort = 1 - 1*(mortality/100)


for i in range(1,years):
    if n[i-1] >= K:
        x.append(i)
        n.append(n[i-1]*mort)


    else:
```

```
        x.append(i)
        n.append((n[i-1]*grwth)*mort)


  plt.plot(x,n)


population(100,6,4,100,115)
```



In this case we added a fairly simple if else statement essentially saying if the population goes above 115 individuals only register mortality the next year. This creates a fluctuating population that hits carrying capacity and then experiences mortality until it is bellow it and then goes up to it again.

This modelling is entirely deterministic the output of the model will not change no matter how many times we run it. It has no stocasticity in it. This form of modelling is probably a bit too simple for most real populations but we can add in further complexity and it can be used as a form of validation of data exploration.

Problems:

Model 1: Model a population that has variable mortality in the summer and in the winter. Where in the summer they produce 8% increase with 2% mortality, while in the winter they only experience a 5% mortality.

Hint this is a cyclical pattern that can still occur inside of a for loop. Use an If statement which does different things depending on the time (bigger hint you can test if it is odd or even).

Challenge Problem

Model 2: Model a population including its own internal age structure with variable parameters between seasons. There are adults with a mortality of 4% and a growth rate of 8% in the summer, and a mortality of 4% in the winter and no growth, and young which have a mortality of 4% but no growth rate in the summer as they are too young to reproduce and then a mortality of 15% in the winter. Young becomes adults after 2 time steps.

Hint, this is extremely tricky. You will need to use multiple functions that can create global variables that the other functions can use. This is approaching a hand build differential equation.