Scientific computing in python 3 – Random distribution sampling for models.

So we have used for loops to create simple deterministic models based on parameters that were provided. However in the real world, parameters like that would come from real data which itself would have a distribution of error around it.

For this reason, we can also use these distributions for models and in reality we can create a far more representative system when we put it into a model.

Let's start with a fairly simple one, a basic climate model. Most people have probably heard of the best and worst case scenario and the temperature limits we have set. Best case scenario is generally considered to be a 2 degrees warming maximum, with worst case being a 6-degree rise by 2100. These have associated rises in sea level.

Let's model the rise in sea level associated across the distribution of temperatures from best case to worst case scenario. We will sample within this distribution of possible scenarios and come up with a distribution of associated sea level rises which we can the summarise.

So lets say that each 0.1 degree increase is associated with a 8 cm increase in sea level rise. So we want to generate a random number of temperature rise and then simulate the rise of sea level associated with it. However, we specify that the temperature rise is within the 2 – 6 degree range and randomly sample within that range.

Lets start by defining a sealevel rise in respect to the temperature put into it.

```
def sealevelrise(temp):

   sealevel = temp * 10 * 8

   print('Sealevel Rise will be', sealevel, 'cm at', temp, 'degrees warming')
```

This code is a simple function that calculates the sea level rise when we put a temperature in it. But then this is just a single temperature rise. How about writing code to sample it many, times.

```
level = []

for i in range(0,99):

   t = random.uniform(2,6)

   level.append(sealevelrise(t))
```

So we then want to workout the mean and the confidence intervals around the mean and we can understand the distribution of sea level rise. Numpy has a mean function built in and a standard error function. We can then use this to work out the confidence intervals.

```
import numpy

meanlevel = numpy.mean(level)

sdlevel = numpy.std(level)


confint1 = meanlevel - (1.96*(sdlevel/100))

confint2 = meanlevel + (1.96*(sdlevel/100))
```

So we have worked out a distribution of sea level rises based on a distribution of temperatures, by sampling within it. However this is a very simple model. Lets build upon it to make it more complex.

If we assume that the estimate time started in 2000 and goes to 2100 we will build a model that will predict sea level. First of all we have to create an actual rate of temperature rise per year. Lets assume this is a linear growth so if we are going to see 2 degree rise in 100 years it will be 2/100 to get the rate per year.

```
def sealevelrise2(temp):

  year = []

  sealevel = []

  sealevel.append(0)

  year.append(2000)

  trate = temp/100

  for i in range(1,101):

    year.append(2000 + i)

    sealevel.append(sealevel[i-1] + trate * (80 / 100))

  return(sealevel, year)
```

This function generates a sea level in each year when given a specified temperature. We could plot this to get the output of a single model. But what we want to do is get the mean of all the models to have a model average.

We have an array of arrays and we can call out to specific coordinates within it so we can work out our means. (or use a pandas dataframe!). To do it the no pandas way we will use a nested forloop.

```
modellevels = []

for i in range(0,99):

  t = random.uniform(2,6)

  modellevels.append(sealevelrise2(t))
```

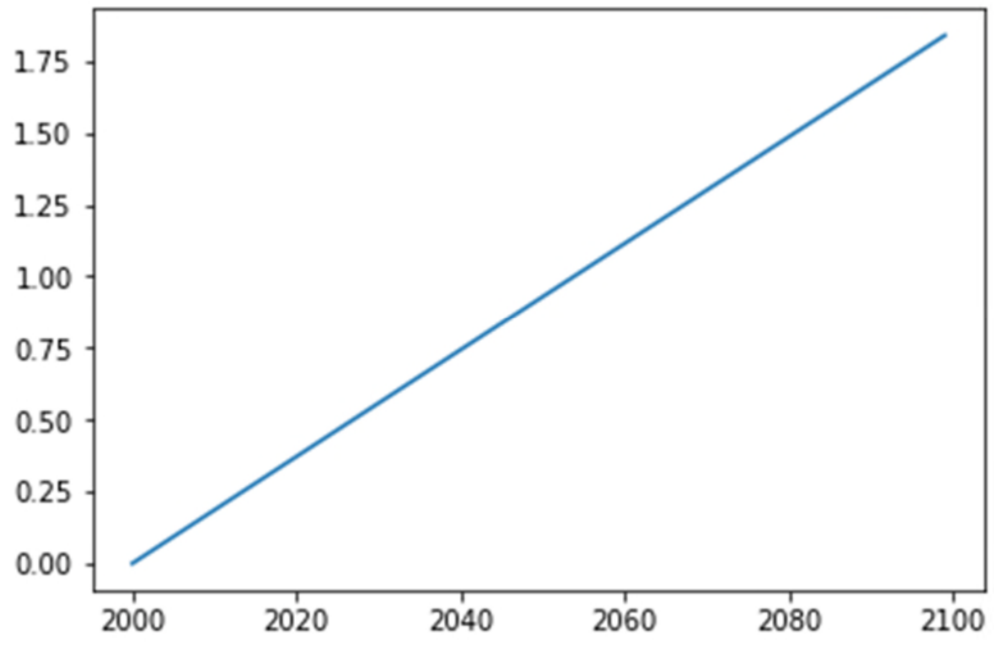
```
modellevels[80][1]


numpy.mean(modellevels[][1])
year = []
year.append(2000)
meanrow = []
for i in range(0,100):
    row = []
    for j in range(0,99):
        pull = modellevels[j][i]
        row.append(pull)


    meanrow.append(numpy.mean(row[j]))


for t in range(1,100):
    year.append(2000 + t)


matplotlib.pyplot.plot(year,meanrow)
```



Well that is not the most complex of plots.

Problem 1: Use the same techniques to produce standard errors around the mean sea level we plotted.

Challenge Problem:

Problem 2: Use the same techniques we just did to model our population from yesterday a population that has variable mortality in the summer and in the winter. Where in the summer they produce 8 ± 3.1% increase with 2 ± 0.8% mortality, while in the winter they only experience a 5 ± 2.3% mortality. Include standard errors.