Histograms and Boxplots

We'll use Anaconda, Python 3, and Jupyter Notebook for the examples. To get started, we need to import a math package, a package for random number generation to create our sample data, and a plotting package.

We'll start by bringing in some packages.

```
import math
from matplotlib import pyplot
import random
```

We may not need the math package, but it may depend on the kind of simulation you choose and how you implement it.

We'll start by creating interest calculations. In order to generate some data, we'll use a simulation with some random interest rates. We'll consider two different cases: 1) normally distributed interest, and 2) uniformly distributed interest.

```
def Interest_normal(principal):
    rate=random.normalvariate(0.03,0.015)
    return principal*rate

def Interest_uniform(principal):
    rate=random.uniform(0.0,0.06)
    return principal*rate
```

The first function calculates normally distributed rates with a mean of 3% and a standard deviation of 1.5% (thus, only 2.5% of time will the interest earned be negative, and 2.5% of the time it will be over 6%). The second function does the same but with a uniformly distributed interest rate between 0 and 6%. You can also use other distributions. Those included in the random package are:

'betavariate', 'expovariate', 'gammavariate', 'lognormvariate', 'normalvariate', 'paretovariate', 'triangular', 'uniform', 'vonmisesvariate', 'weibullvariate'

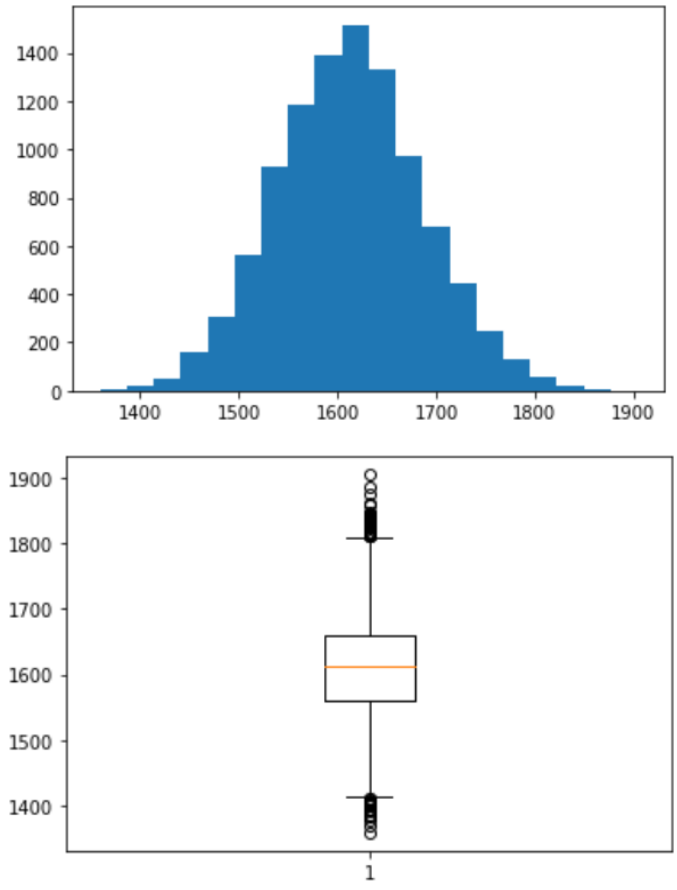
Each type will need its own set of parameters specific to that distribution. However, the two we have is enough to illustrate how to generate the histogram and boxplot.

If you have data, you can enter it into a vector surrounded by square brackets. The simulation will generate some data so we don't have to enter it by hand.

```
years=10
sims=10000
end_balances=[]
for i in range(sims):
    time=0
    balance=1200
    while(time<years):
        balance+=Interest_normal(balance)
        time+=1
    end_balances.append(balance)

pyplot.hist(end_balances, bins=20)
pyplot.show()
```

To get a reasonably smooth graph for the histogram, you need a fairly large set of values. However, when testing it out, don't feel like you need to start with 10,000 simulations.



The code above generates the graph shown. The boxplot can be generated from the same data.

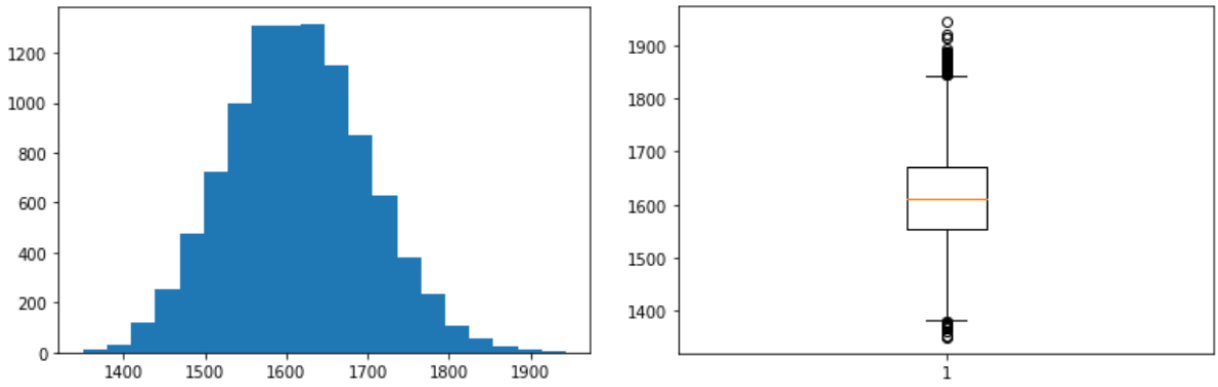
```
pyplot.boxplot(end_balances)
pyplot.show()
```

And the graph produced is shown below.

Obviously, if you use a smaller simulation, you'll get fewer outliers on the boxplot also.

If you run a similar simulation with the uniform distribution, you can plot the graphs in a similar way. The first time I tried the uniform distribution, though, my histogram was not this nice.

The code is exactly the same, except to change from *Interest_normal* in the calculation to *Interest_uniform*.



In this case, both look pretty similar. Depending on the size of your simulation, you may need to tweak the number of bins to get a decent looking graph.

Note: Data generation example adapted from *How to Program: Computer Science Concepts and Python Exercises*, Professor John Keyser, in the Course Guidebook (The Great Courses).

I'll add to this document over time with histogram and boxplot data in other data formats, such as dataframes, an so forth.