Winter Quarter 2019 – UCSB Physics 129L
Homework 3
Due Friday, February 8, 5 pm

Read carefully the instructions on the website on how to prepare your homework for turning it in to the TAs.
If your last name starts with A through L, send the homework to Francesco. Otherwise send it to Jenny.
The emails of the TAs are on the website.
Put the instructor in cc to the email.

Exercise 1
Pick N pairs of variables (x, y) from a uniform pdf between 0 and 1. Interpreted as coordinates of points in the (X, Y) plane, these pairs will populate uniformly a square of area $= 1$. The number n of points that lie within the semicircle of radius $r = 1$ will vary from trial to trial, and will follow a binomial distribution, $p(n) = \binom{n}{N} f^n (1-f)^{N-n}$, where f is the ratio of the area of the semicircle to that of the square, $f = \frac{1}{4}\pi$.
Estimate f as $\tilde{f} = \frac{n}{N}$, and therefore *compute* π as $\tilde{\pi} = 4\tilde{f}$. It can be shown that, for N large enough, and $\tilde{f}$ not too close to 0 or 1, the uncertainty on the estimate of f is $\sigma_f = \sqrt{\frac{\tilde{f}(1-\tilde{f})}{N}}$. Increasing N by a factor of 10, improves the accuracy by a factor of $\sqrt{10} \approx 3$.
Repeat the procedure 5 times with N=1e2, 1e3, 1e4, 1e5, and 1e6. Make sure that your 5 *toy experiments* are statistically independent, i.e., do not recycle the same random numbers from one toy to the next. For each toy output to the screen the number of pairs N, how close (in percent) is $\tilde{\pi}$ to π, and the *pull* defined as $\rho = \frac{\tilde{\pi}-\pi}{\sigma_\pi}$ where σ_π is the uncertainty calculated on your estimate of π obtained from σ_f.
Calculate a χ^2 for your measurement as $\chi^2 = \sum \rho^2$.
Calculate the probability of finding a χ^2 larger than what you have found, given that you have 5 *degrees of freedom.* This is a useful test: probabilities too close to 0 or 1 mean that something is wrong (or that you were unlucky).
Hints:
The exact value of π can be obtained by

```
import math
....
```

```
exactPi = math.pi
```
The chisquare probability can be obtained as
```
from scipy import stats
....
prob = 1 - stats.chi2.cdf(chisquared, 5)
```

Exercise 2
Write a python function $f(x) = x^2(1-x)$. Use this function to calculate numerically $f'(x)$ as either

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$
$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}$$

where h is small.
Calculate $f'(1)$ in the cases for $h = 10^{-2}, 10^{-3}, 10^{-4}, ...$ until something *really bad* happens (see below). Show on the screen both h and $f'(1)$. Format the output of $f'(1)$ to show 20 decimal places. Do the calculation using the built-in python float data type (nothing fancy please).
As h gets smaller and smaller, you should be able to note a couple of things:

- The second approximation is better. This is expected, but perhaps it is surprising that it is so much better. If you think of a Taylor expansion

  $$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{1}{2!} h f''(x) + \frac{1}{3!} h^2 f'''(x) + ...$$

  in the 1st case we approximate to order h and in the second case to order h^2, as the first order terms cancel. So it is a big deal!

- As h becomes too small, cancellations due to finite machine precision becomes important and the accuracy deteriorates. In most cases[1] this happens when $h \approx x\sqrt{\epsilon}$ (first case) or $h \approx x\epsilon^{2/3}$ (second case), as long as x is not too close to zero, where ϵ is the machine precision. For 64 bit floats, $\epsilon \approx 2 \cdot 10^{-16}$.

- At some point either the numerator or the denominator will be within machine precision of zero, and therefore you will get either zero or divide by zero. This is now *really bad*.

[1]See Chapter 5.7 of *Numerical Recipes in C*. If you google it you will find a pdf online.

Exercise 3
Benford's law says that in many numerical datasets the frequency of the first digit of each number is not constant. There tend to be many more numbers with most significant (decimal) digit 1 than 9. This applies best to dataset where the numbers span several orders of magnitude.
The file
`/home/pi/physrpi/campagnari/python/CensusTownAndCityPopulation.csv`
is a comma-separated (csv) file downloaded from the United States Census Bureau and it lists the number of inhabitants for every town in the USA[2]
There are three columns in the csv file: State, Town, Inhabitants, and the first row is a header row. Write a program to read this file, extract the most significant digit, and plot it in a histogram. Hints:

- The csv file is a text file, so you can just read it normally. There is a python module called `csv` that may make it a bit easier.
- The file came from excel and ended up being saved as utf-8 instead of ascii (the curse of text files). Do
  ```
  linux> file CensusTownAndCityPopulation.csv
  ```
  to see what I mean. So watch out for that.
- Have a look inside the file with `more`; the numbers are formatted a bit weirdly.
- There is at least one town with a comma in its name which might fool you, do for example
  ```
  linux> grep -i islamorada CensusTownAndCityPopulation.csv
  ```
  to see what I mean. So pay attention to that as well.

To simplify grading, include the csv file in the tgz file that you send to the TAs.
PS Benford law is used in forensic accounting to detect fraud,
https://www.nigrini.com/

Exercise 4
The quantum mechanical wavefunction for a particle in a 2D square box of sides L is $\Psi(x, y) = A \sin(n_x k_x x) \sin(n_y k_y y)$ wher n_x and n_y are integers, $k_x = k_y = \frac{\pi}{L}$, A is a normalization constant, and the sides of the box are

[2]This file is loaded onto your rpi when you run the update script.

at $x = 0, y = 0, x = L$, and $y = L$. Show the pdf for the position of the particle $p(x, y)dxdy = |\Psi(x, y)|^2 dxdy$ as a color map (or temperature plot if you prefer to call it that) for $n_x = 2$ and $n_y = 5$. Make sure that the x (y) coordinate shows up as horizontal (vertical). Do not worry about the normalization constant.

Exercise 5
Generate 10,000 random numbers between 0 and 1 with pdf $f(x)dx = \frac{1+4x}{3}$. Then plot them in a histogram with 10 bins of size $\Delta x = 0.1$ between 0 and 1. Superimpose on the histogram a curve $g(x) = Af(x)$ with your expected distribution. Think carefully about what A needs to be. Note: the `numpy.random` package does not (as far as I know) have a prepackaged generator of linearly distributed random numbers. There is something pretty close that could be adapted (`np.random.triangular()`), but do not use that. Be more creative....

Exercise 6
Generate a series of 1000 pairs of integers to represent a 2D random walk, ie:

- start at $x = y = 0$
- in the next step move randomly by one unit, either in x or in y (but not in both), either forward ($x \to x + 1$ or $y \to y + 1$), or backwards ($x \to x - 1$ or $y \to y - 1$)

What you have generated is an example of a Markov chain, *i.e.*, a sequence where the probability of (x_i, y_i) only depends on (x_{i-1}, y_{i-1}).
Visualize the random walk through an animation. Something that looks like this:
http://hep.ucsb.edu/people/claudio/ph129-w19/randomWalk.mp4.
Hint: get some inspiration from
`/home/pi/physrpi/campagnari/python/animationExample.py`