Life of a Particle : Assignment 1

Sam Meehan & Claire David

Due Date : 26 January 2019

How to Submit
This assignment should be submitted by replying to the email sent out requesting its submission. In the future, we will use GitHub, but not for this assignment. You should include a single PDF file that has any verbal description of the answers to the questions, along with description of what computer code files go with which question. Please bundle this all into a single tarball and submit this one tarball file.

1.1 - The Stern Gerlach Simulator
Obtain the Stern Gerlach simulation code from the course website - AIMS - Life of a Particle. You may have to search around on the website a bit to find it (Hint : Look for something having to do with "Assignments") but nothing in real research is clear cut so finding it is part of the assignment. You should find four files :

- `SternGerlach_v0_Classical.py`
- `SternGerlach_v1_FixedProduction.py`
- `SternGerlach_v2_UnknownState.py`
- `SternGerlach_v3_FullQuantum.py`

With this code, do the following

1. Execute the code and include as part of your assignment the output `.eps` files that are produced.

2. Determine which of the pieces of code produces the results observed in the actual Stern Gerlach experiment. Describe how you know this and how the other pieces of code fail. (It will probably be easier to complete this task *after* commenting the code)

3. Comment all four pieces of code and include them in the submission. Be sure to include a header at the top with your name and email address so that someone can contact you with a question. Beyond this, commenting code itself is difficult and can be overdone. Follow this guide (Link to Python Commenting Guide) to get a sense of how to comment your code well.

1.2 - The Coach and the Player

As a first example of a Monte Carlo model, we examined a scenario which could plausibly happen in real life with the goal of constructing a model to answer the question. There may be ways to approach this problem using statistical theory (please, let me know if you solve it this way, I have not!) but we can also solve it by building a model to match the data and writing a program to implement this model. The scenario is :

Imagine that you are a consultant who has been hired by an aspiring football coach. The football coach has arrived at AIMS Ghana and every morning he looks out his window and sees two people on the beach, another coach and a player. The coach has a bag of candies and a six-sided die. He is clearly going to be training the player. You notice that he makes the player run sprints using the following procedure :

- He rolls the dice and gets a number. He gives the player that number of candies to eat.
- He rolls the dice another time and get a number. He then tells the player to run at a speed of 5 m/s for that many seconds.
- The player eats the candies and then runs down the beach.
- The coach then walks behind him to catch up while the player rests and they repeat this exercise.

This continues on until the coach rolls the dice and there are not as many candies in the bag as the dice indicates he should give the player. At this point, the player takes the rest of the candies, the coach rolls the dice one more time to obtain the speed of the player, and the player runs. You have seen this for three mornings thus far and have observed that the players distance for those three days is 25m, 19m, and 40m. This training procedure seems effective so the football coach has hired you to figure out how many Candies are in the bag. He wants to purchase the same amount of candies and execute the same training regimen. He also doesn't want to buy a huge amount of candies and overtrain his players, so providing him with an accurate estimate of the number of candies is important.

Your goal is to provide the coach with this estimate.

To help, he has provided you with a set of observations for the distance the player runs each morning over the course of the next 1000 days (... ok, fine fine, so he had to wait almost three years to accumulate this data, let's pretend ...). These observations can be found on the course website (again, you'll have to search for it).

1.3 - Coding Like a Hipster

You don't need to do any work here, these are just directions. After reading and understanding them (ask if they are not clear!) choose one of the two questions to complete. If you do both, bonus points will be awarded.

For many common tasks, there are built in functions already existing in python, or which can be imported from the `math` or `numpy` modules. In particular, the ones you may have used before are `min(LIST)`, `max(LIST)`, `sum(LIST)`, `len(LIST)`. However, these are not necessary for "good" programmers. Nearly all programs can be written using the following small set of operations

- + (add)
- $*$ (multiply
- / (divide)
- = (assignment)
- == (equals comparison)
- $<$ (less than comparison)
- $>$ (greater than comparison)

In addition to these operators, it is taken for granted in programming that you can use the following features as well

- variables and lists - for storing the initial dataset
- for loops
- if statements
- print statements - for viewing your code

In this quiz, these are the only things that may appear. If you determine that you absolutely need some other function or operator, then include a comment clearly describing why this is the case.

Finally, you are not allowed to "hard code" in your program, meaning that there cannot be code that you must manually change each time you run it. An example of this is the length of a list. If I have a list [1,4,2,5,3,6], and you want to use the number of items in the list in your code, then the number "6" may not appear in your code.

Option 1 - Standard Deviation Like a Hipster
For this question, follow the directions to code like a "hipster". Note that you can use the ROOT package to make histograms or graphs if you like.

Given a set of numbers Q (example Q =[2,3,4,2]), write a program which calculates the standard deviation of this set. For the purposes here, the definition of the standard deviation of a set of numbers is

$$\sigma(Q) = \sqrt{\frac{\sum_{i=0}^{N-1} (x_i - x_{avg})^2}{N}}$$

$$x_{avg} = \frac{\sum_{i=0}^{N-1} x_i}{N}$$

$$N = N(Q)$$

Option 2 - Sorting Like a Hipster
For this question, follow the directions to code like a "hipster".

Given this set of data (you can copy and paste it into an array if you like)

[71, 51, 32, 62, 84, 109, 43, 92, 72, 41, 102, 80, 72, 69, 46, 94, 52, 95, 90, 72, 63, 70, 34, 80, 78, 34, 31, 37, 26, 41, 42, 107, 33, 108, 108, 75, 66, 23, 90, 53, 24, 70, 26, 41, 93, 24, 71, 39, 48, 66, 97, 107, 77, 71, 67, 39, 38, 107, 96, 92, 84, 46, 60, 95, 87, 90, 92, 63, 78, 78, 84, 107, 70, 108, 32, 36, 93, 108, 49, 72, 56, 43, 30, 56, 51, 97, 45, 92, 40, 43, 49, 83, 98, 28, 99, 97, 102, 89, 58, 87]

write a program in python which sorts the data into an ascending array. By this, I mean that the 0^{th} element in the list is the smallest, and the last element in the list is the largest. If there is more than one element which has the same value, then the order of these two elements does not matter.