Computing Methods for Physics 1

Lecturer: Francesco Pannarale

Hands-on lab session 3A, A.Y. 2021-22

The goal of these sessions is to practice the basic Python notions covered in class. There are three exercises which you should solve in 3 separate Jupyter notebooks.

1. **Evaluate π numerically with a Monte-Carlo** — The area of a circle with radius equal to 1 is π and this circle can be enclosed in a square with sides equal to 2. Randomly place points in this square and count the fraction of points that ends within the circle: this will be a Monte-Carlo estimate of π.

   Instructions:

   - Generate N points on a 2D plane, representing them with Cartesian coordinates with x and y uniformly drawn between -1 and 1 [$\mathcal{S} = \{(x, y) : x \in U[-1, 1] \text{ and } y \in U[-1, 1]\}$]
   - Separate your N points in ones that fall within a circle of radius 1 and centre $(0, 0)$, and ones that fall outside such circle [$\mathcal{C} = \{(x, y) \in \mathcal{S} : ||(x, y)|| \leq 1\} \subset \mathcal{S}$ and $\bar{\mathcal{C}} \subset \mathcal{S}$, with $\mathcal{C} \cup \bar{\mathcal{C}} = \mathcal{S}$ and $\mathcal{C} \cap \bar{\mathcal{C}} = \emptyset$].
   - Estimate π by dividing the number of points in the first group by N [$n(\mathcal{C})/n(\mathcal{S})$]: see how your precision varies with N.
   - Make a square plot of the two groups of points, using two different colours to distinguish them.

2. **Kniffel** — This exercise is inspired by a game played with 5 dice with 6 sides numbered 1 through 6 all rolled at once. Your code must determine the distribution of the number of rolls it takes to achieve six special combinations in total: the dice show $1, 2, 3, 4, 5$, they show $2, 3, 4, 5, 6$, they all show the same number (this is called a kniffel), only 3 of them show the same number, only 4 of them show the same number, 3 of them show the same number and the remaining 2 show another number [this is a full house, e.g., $(2, 2, 2, 5, 5)$].

   Instructions:

   - Simulate a fair roll of the five dice.
   - Check whether a special combination was produced.
   - Repeat the process (automatically!) until all special combinations are exhausted, keeping track of how many rolls it took.
   - Repeat all that N times, gathering information on the number of rolls each time.
   - Plot the distribution of number of dice rolls obtained as a histogram.
   - Also plot the cumulative distribution as a histogram.
   - What kind of distribution is it? Can you eyeball a fit with your intuition and knowledge (i.e., without a fitting algorithm)?

3. **CLR** — This exercise is also inspired by a dice game. In the setup you will simulate, there are three players with 20 coins each. The players roll the dice in turns and the outcome of the dice roll determines the fate of their coins. A player that has 3 or more coins rolls 3 dice, a player with 2 coins left rolls 2 dice, a player with 1 coin left rolls 1 die, and a player without coins cannot roll. Each dice has 6 faces: 1 showing a C, 1 showing an L, 1 showing and R, and 3 showing a dot. If a player rolls and L/R, one of their coins go the player on their left/right; if a C is rolled, a coin goes to the centre; rolling a dot has no consequences. The last player left with coins wins.

   Instructions:

   - Create a dictionary with the number of coins for each player and in the center.
   - Simulate a round of fair dice rolls, ensure that the number of dice rolled is established correclty.
   - Update the dictionary after each player rolls.
   - Keep track of how many full rounds are needed to have a winner.
   - Simulate N games of this kind, gathering data about the rounds needed for the game to end.
   - Plot the distribution of number of dice rolls obtained as a histogram.
   - Also plot the cumulative distribution as a histogram.
   - What kind of distribution is it? Can you eyeball a fit with your intuition and knowledge (i.e., without a fitting algorithm)?

Recommendations

- Comment your notebook.
- Start all exercises with a small N.
- Use functions.
- Use comprehension.
- Think thoroughly about when sets are needed: manipulating sets is faster than manipulating lists.
- Label your plots and their axes.
- Runtime guidelines: if you are way above these, you should seriously optimize your code:
  - exercise 1 with $N = 10^6$: $T_{\text{Run}} \sim 4\,\text{s}$;
  - exercise 2 with $N = 10^4$: $T_{\text{Run}} \sim 30\,\text{s}$;
  - exercise 3 with $N = 10^5$: $T_{\text{Run}} \sim 10\,\text{s}$.