Economic simulation

Prerequisites

- Python 3
- Module abc

Objective

Develop an application to simulate financial transactions between individuals and study the distribution of wealth among the population.

The goal is to design a "sufficient well designed application" that allows future evolutions to take place without breaking the existing code.

Explanations

Let's consider a group of individuals (a **population**), each one having a certain amount of **wealth**, initially defined by an **initialization function**. We call an **iteration** the process of exchanging wealth between two individuals of the population. Both individuals will be chosen according to an **interaction function** and they will exchange wealth following the rules of a **transaction function**. The simulation will run for multiple iterations. The aim is to track the distribution of wealth among the individuals, thanks to a **Gini parameter** defined as :

$$G = \frac{2\sum_{i=1}^{n} i y_i}{n\sum_{i=1}^{n} y_i} - \frac{n+1}{n}$$

where y_i is the wealth of individual i. **This array should be sorted in ascending order.** A value near 0 means that the wealth is equally spread among the population (each individual has roughly the same wealth than the others) whereas a value tending to 1 means that a strong inequality arises (a person or group of persons centralizes all the wealth)

Computing this parameter at the start and at the end of the simulation will show how the initialization, the interaction and the transaction function interact with the wealth distribution.

How to proceed

We will start by designing the application with a classical functional programming path.

1) Declare a population of 1000 individuals. What is the simplest way/type to represent population's wealth ?

2) Declare 3 functions (initialization, interaction, transaction) and define their implementation.

- start with an uniform initialization (every individual has the same wealth).
- pick two individual randomly among the population
- when exchanging wealth, one individual receives one part **of the sum of both individual's wealth**, the other takes the rest.

Explain and justify the type of the arguments of these functions as well as their return type.

3) Setup the simulation by initializing population's wealth and repeating 10000 times an iteration (interaction then transaction).

4) Compute the Gini parameter.

5) Change the size of the population and the number of iteration and perform a small parameter study.

Now we will modify the design of our application.
Let's consider a customer interested in our application. He may buy it but we need to ensure that it is well designed for future developments. Indeed, the customer will probably need new functionalities and we don't want to modify our existing working code that could break our application. We then switch to a OOP pattern design.

6) Create 3 classes, responsible of initialization, interaction and transaction respectively. Move the previous defined functions into these classes (methods).

Now we will add abstraction in our code to be able to add new functionalities without breaking the application.

7) Create 3 abstracts classes that will define the concept of initialization, interaction and transaction respectively. Use module **abc** to define abstraction in Python

8) Create 3 concrete classes that inherit from the 3 abstract classes defined in step 7 and define theirs methods.

9) Now add a new, second initialization function. This function will initialize each individual's wealth randomly (use module **random**)

10) Add a new, second transaction law such as among the two individuals involved in the trasanction, one takes all the combined wealth, the other takes noting.

11) What can you conclude about this design ?