Pyramid Scheme

In 1919, Charles Ponzi adapted a money-making scheme involving buying and selling postage stamps to the point where the primary mode of earning money was no longer profit from the stamps, but subscription fees paid by new recruits to the scheme.

A Ponzi scheme, or pyramid scheme is the term given to a venture in which:

- ***A high reward is promised for a low risk.***
- ***The details of how money is made is not made clear to investors.***
- ***The primary method of making money is through recruitment.***

Example:

I tell you I have a great scheme for making money (the details of which need not concern you). You can buy into it for a £100 subscription fee, but you'll soon make your money back, because for every new investor you find, you get to keep half of their subscription fee (with the other half going to me). So if you recruit three people, you get £150 from them, giving you an immediate £50 profit. But even better, for every person they recruit, you receive a quarter of their subscription (50% goes to the recruiter, 25% to their 'boss', and 25% to their boss's boss).

For instance:

You pay £100 to join the scheme. ($-£100$)

You recruit three people, who pay £100 each to join. You keep 50% of this ($+£150$).

At this point, you're up £50.

Each of your three recruits also recruits three people, and you get 25% of anything they bring in: ($3 \times 3 \times £25$) which gets you an additional ($+£225$).

Overall, you end up with a profit of £275 from a £100 investment – that's a 175% profit!

Am I my boss's boss?

The guy at the top of the pyramid is essentially his own boss (and his boss's boss), so he has the added advantage of getting the entire subscription fee from everyone he recruits, as well as both 25% shares from anyone they recruit.

Try it out:

You set up a pyramid scheme, and convince four people to subscribe, following the rules described above. Each of them recruit four people, and each of them four more.

How much do you end up with, and how much does everyone else make?

What could possibly go wrong?

The fundamental issue with the pyramid scheme is that it doesn't involve the generation of wealth in any way – it is essentially a method for encouraging people to give one another money in the hopes that they can recoup their investment by perpetuating the scheme. However, to make back the money they have spent, each person needs to recruit more than a single investor. In the example given above, if you find two new recruits, you break even, even if they find no recruits themselves. If they themselves recruit, you do even better.

Question 1:

By the time the pyramid reaches 10 layers deep, with 1 person at the top, two on the next layer, 4 on the next, 8 on the next, etc, there will be 512 people in the 10^{th} layer. Sooner or later, with this exponential growth, the available population (all the people, or at least anyone with sufficient money and gullibility) will very rapidly be exhausted, and the bottom layer lose their investment. *How much money is made by each strata within the pyramid?*

Question 2:

If I recruit two people during the first week, and they each recruit two people during the second week, and they each recruit two people in the following week, etc, *how long will it be before there are over a million subscribers to the scheme?*

Can Python simulate the effects of recruiting within a given population?

Making People

Step 0: Figure out what attributes and methods a person has that you care about.

We could make a **Person** class which is capable of keeping track of the following:

- How much money they currently have.
- How gullible they are (how likely they would be to fall for the pyramid scheme).
- Whether or not they are a member of the scheme, and if they are:
    - Who their immediate boss is (the person they give half of their proceeds to).
    - How many people they have succeeded in recruiting.
    - What level of the pyramid they are on.

The **Person** object would also need to be able to *do* the following:

- Attempt to recruit someone.
- Collect subscriptions from recruits.
- Pass on money to their boss.
- Provide summary information (their pyramid level, number of recruits, money).

Step 1: Write the class code.

```
1   import random
2
3   class Person:
4     def __init__(self):
5       self.cash = 100
6       self.gullibility = random.random()
7       self.boss = None
8       self.level = None
9
10    def try_to_recruit(self, other):
11      if other.boss == None:
12        if random.random() < other.gullibility:
13          other.boss = self
14          other.level = self.level + 1
15          other.cash -= 100
16          self.cash += 50
17          self.boss.cash += 25
18          self.boss.boss.cash += 25
```

I'm importing the **random** module, because I'll use it both to set the gullibility level of each person and to select people at random from the population.

The **Person** class has the key attributes **cash**, **gullibility**, **boss** and **level**. The **boss** is initially nobody, but will, if a person gets recruited, point to the person who recruited them. **level** refers to how far they are from the 'peak' of the pyramid (the original boss of the scheme).

There is only one other method: **try_to_recruit** takes an other person as the input, and only does anything if they are not currently in the scheme (ie, don't have a boss). In that case, they get the sales-pitch (a random number between 0 and 1 is compared to their gullibility level) and, if successful, they are recruited.

Step 2: Make a simulation.

```
20  n = 20
21  sims = 1000
22
23  population = [Person() for i in range(n)]
24  population[0].boss = population[0]
25  population[0].level = 0
26
27  max_level = 0
28  for i in range(sims):
29      a, b = random.sample(population, 2)
30      if a.boss != None:
31          a.try_to_recruit(b)
32          if b.level != None:
33              max_level = max(max_level, b.level)
34
35  results = {i:[] for i in range(max_level + 1)}
36  for p in population:
37      if p.level != None:
38          results[p.level].append(p.cash)
39
40  for level, amounts in results.items():
41      amounts.sort()
42      num = len(amounts)
43      print(f"Level {level}: {num} recruits, mean: {sum(amounts)//num}, Median: {amounts[num//2]}")
44      print(f"{amounts[::-1]}\n")
```

The code above makes use of our **Person** object, and hopefully encapsulating all of that in the class defined previously allows the main body of the code to be considerably more readable.

Try reading carefully through the code above, annotating the various lines to explain what they do, and why they are there. A good piece of code will be understandable even if it doesn't have comments throughout, but it's good practice to have both: clear terminology, but also additional clarification where necessary with comments. For instance, if you add something after writing your code, to fix a problem or add functionality later that wasn't vital to the basic running, maybe you could add explanation there:

Extensions / Adaptations:

Write down some things you wonder about this scenario that you think might be worth investigating. You might have some thoughts before writing the code, and you may have more once you've tried running it. For example, what happens if the boss's boss's boss gets some money? Or if a different proportion is given to each person? Can you run enough simulations to see how extensive a pyramid scheme can become depending on the size of the initial population?