Crowdfunding *([redacted] and Julien Dos Reis)*

I) Description

Financial Instrument: We have chosen to work on crowdfunding

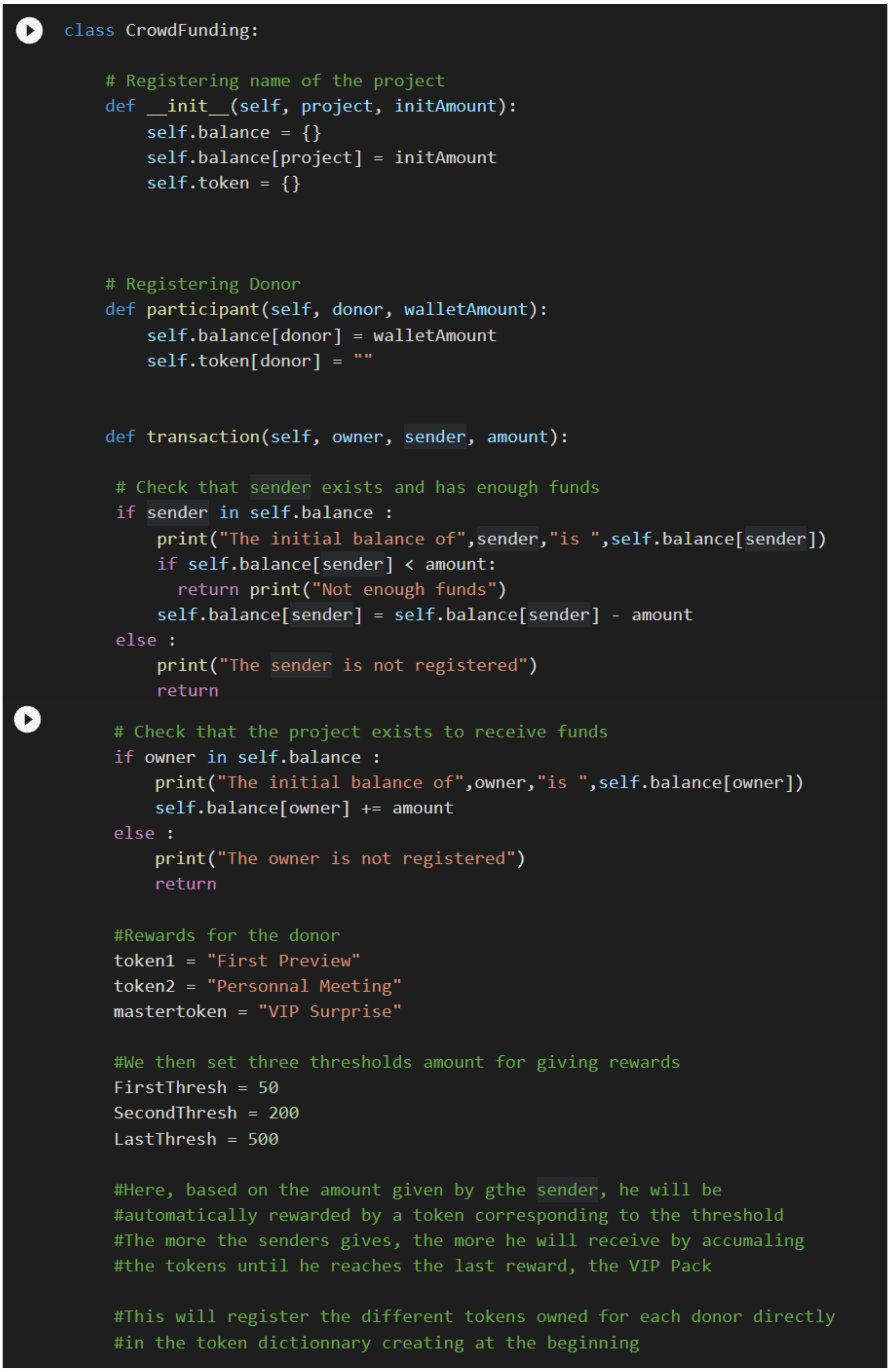
Crowdfunding is a way of raising money to finance projects and businesses. It enables fundraisers to collect money from a large number of people via online platforms. Crowdfunding is most often used by startup companies or growing businesses as a way of accessing alternative funds. There exist different types of crowdfunding, from equity-based crowdfunding to donation-based crowdfunding. The financial crisis of 2008-09 seems to have been a game-changer for the crowdfunding sector. Amidst the perceived collapse of the financial markets, many people turned to the Internet, and to each other, to seek funding. This period also marks the launch of two modern crowdfunding giants – IndieGoGo (2008) and Kickstarter (2009) as well as the proliferation of many other, niche crowdfunding sites. In just five years, crowdfunding has grown 1,000% and the number of platforms globally has surpassed 450.

Here we want to focus on ***Reward-based Crowdfunding*.**
This kind of Crowdfunding will offer a reward to the ones who decide to help financing the project. There are different kinds of rewards depending on the project. It can be having a percentage of profits generated by the project or more materialistic rewards like in our case for example.

Project chosen: We have chosen here to base our Crowdfunding smart-contract on a fictional movie project. The reward for donors will be different levels of tokens. These tokens will give access to different gifts. These rewards are fictional and only aim to give an example on how this smart contract could work.

Functionalities: The Crowdfunding smart contract objective is to replace the Crowdfunding platform. The project owner can initiate the smart contract by registering in a dictionary the project name and the amount of funding it needs. The function "**participant**" helps in registering the **lender/donor**. It also registers the amount the donor has in his wallet and initializes his reward balance. The **transaction** function imitates the process of a Crowdfunding: if one of the users is not registered, the function will stop and explain why. In parallel it will also check the existence of the project to which the donor wants to donate and will run the transaction if it exists. During the **second step**, it will check whether the donor has enough funds (compare the amount he wants to give to the funds at his disposal) and will stop if not. If the donor has enough funds, the reward system starts. It will compare the amount to donate to a certain threshold and send the appropriate reward.

II) <u>Code</u>



```
class CrowdFunding:

    # Registering name of the project
    def __init__(self, project, initAmount):
        self.balance = {}
        self.balance[project] = initAmount
        self.token = {}



    # Registering Donor
    def participant(self, donor, walletAmount):
        self.balance[donor] = walletAmount
        self.token[donor] = ""


    def transaction(self, owner, sender, amount):

     # Check that sender exists and has enough funds
     if sender in self.balance :
         print("The initial balance of",sender,"is ",self.balance[sender])
         if self.balance[sender] < amount:
           return print("Not enough funds")
         self.balance[sender] = self.balance[sender] - amount
     else :
         print("The sender is not registered")
         return
```

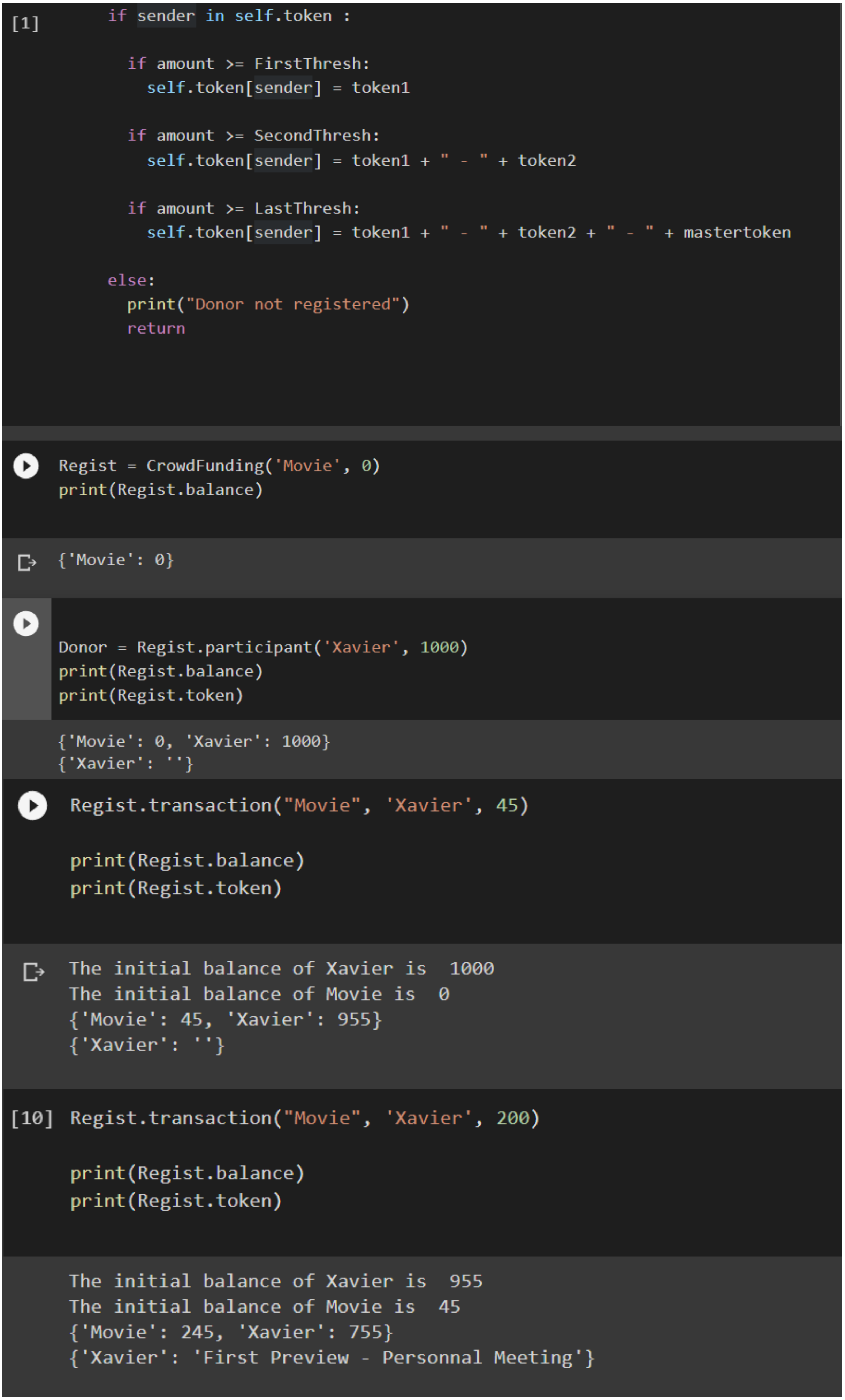
```
     # Check that the project exists to receive funds
     if owner in self.balance :
         print("The initial balance of",owner,"is ",self.balance[owner])
         self.balance[owner] += amount
     else :
         print("The owner is not registered")
         return

     #Rewards for the donor
     token1 = "First Preview"
     token2 = "Personnal Meeting"
     mastertoken = "VIP Surprise"

     #We then set three thresholds amount for giving rewards
     FirstThresh = 50
     SecondThresh = 200
     LastThresh = 500

     #Here, based on the amount given by gthe sender, he will be
     #automatically rewarded by a token corresponding to the threshold
     #The more the senders gives, the more he will receive by accumaling
     #the tokens until he reaches the last reward, the VIP Pack

     #This will register the different tokens owned for each donor directly
     #in the token dictionnary creating at the beginning
```



[1]

```
    if sender in self.token :

      if amount >= FirstThresh:
        self.token[sender] = token1

      if amount >= SecondThresh:
        self.token[sender] = token1 + " - " + token2

      if amount >= LastThresh:
        self.token[sender] = token1 + " - " + token2 + " - " + mastertoken

    else:
      print("Donor not registered")
      return
```

```
Regist = CrowdFunding('Movie', 0)
print(Regist.balance)
```

```
{'Movie': 0}
```

```
Donor = Regist.participant('Xavier', 1000)
print(Regist.balance)
print(Regist.token)
```

```
{'Movie': 0, 'Xavier': 1000}
{'Xavier': ''}
```

```
Regist.transaction("Movie", 'Xavier', 45)

print(Regist.balance)
print(Regist.token)
```

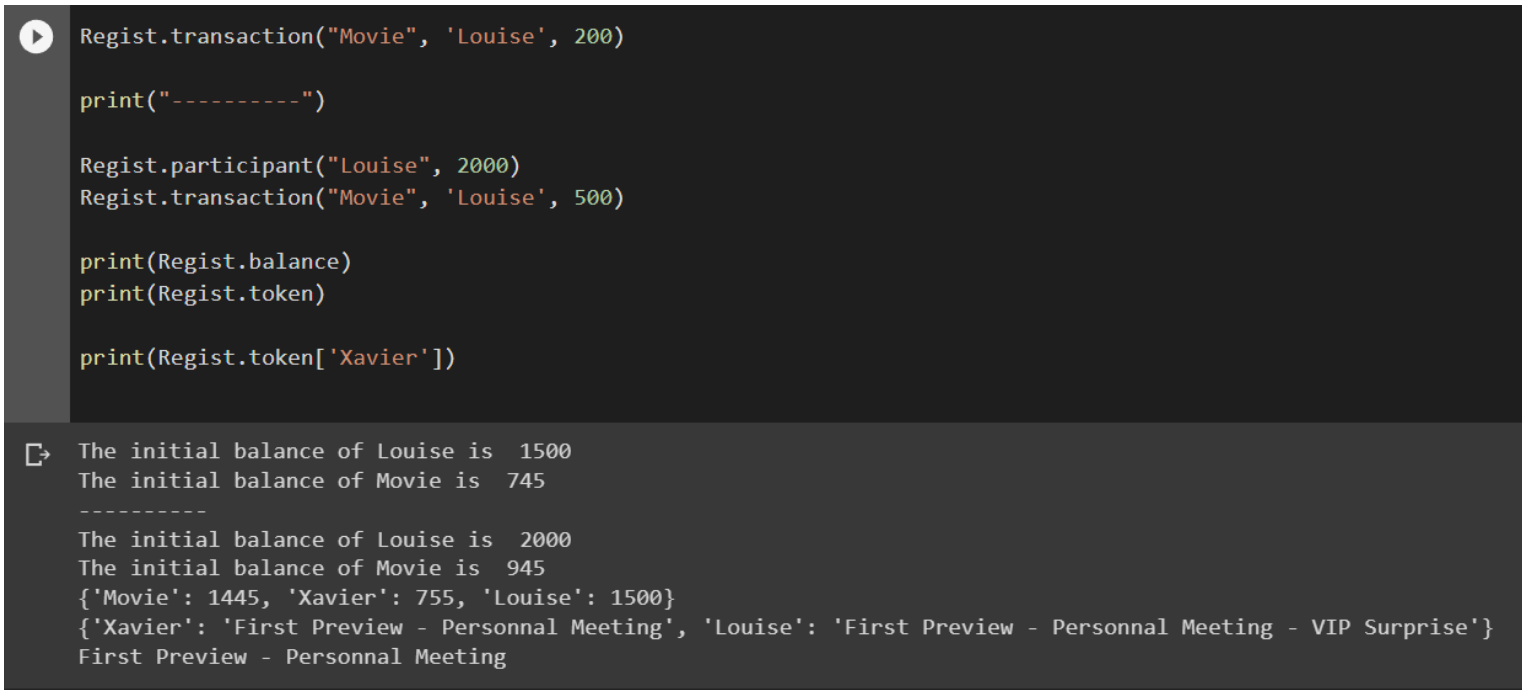
```
The initial balance of Xavier is  1000
The initial balance of Movie is  0
{'Movie': 45, 'Xavier': 955}
{'Xavier': ''}
```

[10]

```
Regist.transaction("Movie", 'Xavier', 200)

print(Regist.balance)
print(Regist.token)
```

```
The initial balance of Xavier is  955
The initial balance of Movie is  45
{'Movie': 245, 'Xavier': 755}
{'Xavier': 'First Preview - Personnal Meeting'}
```



```
Regist.transaction("Movie", 'Louise', 200)

print("----------")

Regist.participant("Louise", 2000)
Regist.transaction("Movie", 'Louise', 500)

print(Regist.balance)
print(Regist.token)

print(Regist.token['Xavier'])
```

```
The initial balance of Louise is  1500
The initial balance of Movie is  745
----------
The initial balance of Louise is  2000
The initial balance of Movie is  945
{'Movie': 1445, 'Xavier': 755, 'Louise': 1500}
{'Xavier': 'First Preview - Personnal Meeting', 'Louise': 'First Preview - Personnal Meeting - VIP Surprise'}
First Preview - Personnal Meeting
```