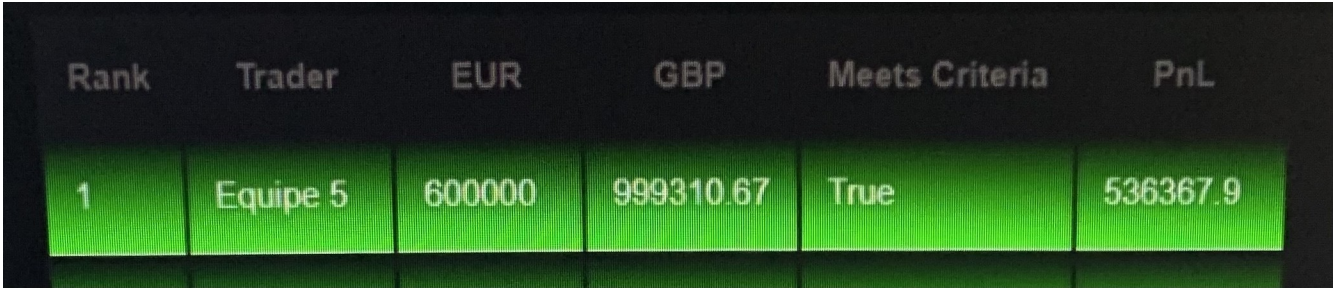
Sommaire :



Rank	Trader	EUR	GBP	Meets Criteria	PnL
1	Equipe 5	600000	999310.67	True	536367.9

I. Notre stratégie

Notre stratégie reposait sur les différents « crash » qu'a subi le marché EUR/GBP.

Une première phase de test se composant d'un cours à peu près stable de l'indice EUR/GBP, nous a permis de comprendre les outils mis à disposition et la manière de faire du trading. A ce moment là, nous testions différents achats et ventes et analysions la courbe du marché pour nos trades futurs. Nous observions de faibles gains/pertes sur chacune des équipes par rapport à notre capital de départ, cela étant dû à la petite variation du taux de change et le peu investi par chaque équipe. Nous avons alors décidé que chacun de nos trades s'effectuera à une valeur de 100k pour pouvoir considérablement augmenté nos gains et en vu du peu qu'investissent les autres équipes. Cela étant risqué car pouvant aussi considérablement augmenté notre perte.

Nous nous sommes alors penchés sur les deux annonces concernant la perte de la valeur du GBP puis de l'EUR à différentes échéances.

Première annonce
En regardant notre historique de transactions, nous avons accès à tous les taux pour lesquels nous avons investis. Nous avons alors calculé la moyenne de ces taux de change, à partir de cet historique. Ainsi, en se basant sur cette moyenne, nous avons calculé les futurs taux maximaux des crash. C'est à dire +40% et -20% de ce taux moyen.
Notons X la moyenne calculée, à 16h30, le nouveau taux serait de X(1+0,4) car il augmenterait de 40 %. De même pour la perte de valeur de ce taux à 16h45 avec un nouveau taux de X(1-0,2).
Cette approximation nous a donné une première bonne idée pour savoir à quel taux il fallait vendre/acheter des actifs.
Une fois cela effectué et le premier crash passé, nous nous rendons compte que nous avons vendu nos actifs un peu trop tôt par rapport à l'évolution de la courbe du taux. Cela n'empêche de bons gains et l'évolution considérable de notre capital.

Seconde annonce
Notre stratégie est alors différente pour le deuxième crash. Afin de maximiser notre profit, nous décidons de prédire jusqu'où le taux allait descendre selon le taux exact que nous observions à l'heure du crash 16h45.
Ainsi, nous avons investi la totalité de notre capital au taux le plus bas observée sur le marché, à l'aide d'un algorithme présenté plus bas.

Nous sortons donc de ce deuxième crash avec un gain très important. Puis nous avons revendu de façon à respecter les 30 % d'EUR de notre solde en GBP, après que le taux se soit stabilisé à 0,8970 ; très supérieur au taux auquel nous avions achetés.

II. Notre implémentation

Voilà différentes fonctions que nous avons crées dans le fichier fourni client.py, nous ayant aider à analyser et faire nos décisions.

```
def get_positions(trader_id):
    '''
        Cette fonction nous a servi à avoir notre position EUR/GBP.
    '''
    api_url = URL + "/positions/"+trader_id
    res = requests.get(api_url)
    if res.status_code == 200:
        return json.loads(res.content.decode('utf-8'))
    return None
```

Fonction qui nous a servi à accéder à notre position EUR/GBP lorsqu'il y avait divers coupure sur le site, nous empêchant de voir notre position actualisée.

```
def history():
    '''
        Cette fonction sert à  nous montrer nos historiques de trade
    '''
    api_url = URL + "/tradeHistory"
    res = requests.get(api_url)
    if res.status_code == 200:
        for i in json.loads(res.content.decode('utf-8')):
            if i.get('user_name') == 'Equipe 5':
                print(i)
```

Fonction qui nous a servi à accéder à l'historique de nos investissements. Cela nous a servi à voir chaque taux avec lesquels nous avons pris position afin de calculer une moyenne du taux de change, et avoir une « fourchette » des prix probables.

```python
def benef(prixchoisi):
    '''
        On a utilisé cette fonction pour acheter lorsque le prix actuel est
        inférieur au prix choisi. Le but est de pouvoir acheter lorsque le prix
        est bas notamment lors du deuxieme crash en calculant 20% du prix actuel.
    '''
    prixactuel = int(get_price())
    if prixactuel < prixchoisi:
        print("Effectively traded at:" + trade(TRADER_ID, 100000, Side.BUY))
```

Fonction qui nous a servi à acheter des actions. Elle achète une action fixée à 100k lorsque nous lui avons donné en argument le taux qu'on avait envisagé selon le crash.
Exemple : Nous avions prédit que le taux allait descendre jusqu'à 0,65 au second crash. On a alors appelé la fonction *benef(0,7)* qui a achété 100k dès lors que le taux était en dessous de 0,7 (en laissant donc une marge d'erreur de 5 % sur notre calcul).
Cette fonction peut être utile notamment dans une boucle while, elle analyse ainsi le prix du marché chaque seconde et trade automatiquement lorsque la condition est vérifiée jusqu'à épuisement du capital en EUR.

```python
def buy_or_sell(seuil, qty): #à mettre en boucle infinie
    '''
        pour acheter ou vendre dans une zone par rapport à un seuil fixé
    '''
    if get_price()<0.9*seuil:
        trade(TRADER_ID, qty, Side.BUY)
    elif get_price()>=1.10*seuil:
        trade(TRADER_ID, qty, Side.SELL)
```

Fonction ayant pour but de faire du scalping durant les phases oscillatoires (stables). C'est à dire a tirer profit des phases oscillatoires autour d'une valeur arbitraire. En lui donnant une valeur seuil (valeur arbitraire), elle permettait d'acheter quand la courbe du taux était en « oscillation basse » et de revendre en « oscillation haute ». Lorsque le taux sortait de la phase, il fallait arrêter la boucle portant sur cette fonction et attendre une nouvelle phase oscillatoire, avec une nouvelle valeur de seuil.

```python
def anticipation_crash(evolution, qty, side): #à mettre en boucle infinie

#evolution represente la variation annoncée par les institutions financières,
#donc peut etre positif ou négatif.
    '''
        prise en compte des fortes variations annoncées par les institutions
        financières en prévoyant le prix de chute ou de hausse attendue et
        donc savoir à quel meilleur moment prendre position
    '''
    a=get_price()
    if get_price()<1.10*evolution*a or get-price()>0.9*a*evolution:
        #a*evolution represente le prix minimal ou maximal prévu;
        trade(TRADER_ID, qty, Side.side)
```

Fonction exclusivement implémentée pour gérer les annonces de la part des institutions financières. Elle prend en argument l'évolution attendue, la quantité avec laquelle on souhaite prendre position, et la position. On a utilisé cette fonction, mise dans une boucle infinie, exactement 20 secondes avant l'heure de la deuxième annonce. Ainsi, dès qu'on était dans une zone de plus ou moins 10 % du prix attendu après évolution, l'algorithme prenait positon « en masse ». Puis, une fois qu'on avait plus de quoi acheté (pour la deuxième annonce), on pouvait stopper la boucle et attendre que le taux remonte pour tout vendre. On a ainsi fait un énorme bénéfice.

Au final voici comment nous appelions nos fonctions en les mettant en commentaire quand elles nous servaient pas et en les mettant en fonctionnelles quand on en avait besoin :

```python
if __name__ == '__main__':
    print("Expected to trade at:" + str(get_price()))
    print("On a en tout : "+str(get_positions(TRADER_ID)))
    print("Notre historique : ")
    history()

    #Achetons 100k lorsque l'on descend sous le taux de 0.7
    while(True):
        benef(0.7)

    #Achetons 100k lorsque la variation est de -20%
    while(True):
        anticipation_crash(-0.2, 100000, BUY)
        #le while s'arrête quand la forte baisse/hausse s'estompe

    #Achetons/vendons 50k dans une zone de taux de 0.85
    while(True):
        buy_or_sell(0.85, 50000)
    #la valeur seuil est à mettre à jour pour chaque palier assez plat (petites oscillations)
```

III. Conclusion

Nous avons finalement réussi à maximiser notre gain en minimisant les risques pour un total d'environ +530 000 PnL. Notre stratégie de se concentrer sur les crash nous a permis d'atteindre cet objectif, à l'aide de notre implémentation et de nos calculs.

Nous avons appris sur l'utilisation de bibliothèques tels que json et requests pour faire le lien avec HTTP.
Nous avons aussi pu échanger sur divers stratégies de trading et mette en œuvre notre apprentissage des marchés financiers sur une situation concrète s'inscrivant pleinement dans le cadre de nos projets futurs.
L'ambiance compétitive du Hackathon nous a motivé et nous a donné un avant goût d'un environnement nous stimulant et exigeant.