Après l'élaboration d'une panoplie d'idées loufoques ou inutiles, mais amusantes (créer un robot qui fait des sudokus, ou un robot twitter qui compile les gardiens partant de la Ligue nationale de hockey), je suis tombé sur une idée parfaite pour la situation. Il s'agit d'un script python qui me permet de réaliser un jeu-questionnaire sur le hockey. Ça réunit un élément de grand défi (coder dans Python, ce qui n'est pas encore une habileté complètement maitrisée) et le plaisir de parler d'un sujet qui me tient à cœur; le hockey.

J'ai choisi Python parce que c'est une technologie qui, malgré le fait qu'elle est très difficile à utiliser, permet une polyvalence et permet de faire un grand amalgame de choses. Étant donné que c'est une grande partie du cours que nous venons de suivre cette session, j'ai décidé de m'y plonger davantage. C'est des capacités qui ne seront jamais perdues et j'aimerais dans le futur pouvoir compléter plus de projets liés au codage.

Pour débuter, j'ai utilisé Excel pour créer une dizaine de questions, qui m'auront servi de base pour élaborer mon script. Une fois le script fonctionnel avec ces questions, il suffit ensuite de modifier quelques chiffres pour l'adapter à mon projet final, qui comportera une centaine de questions.

Une fois cette première étape complétée, on s'aventure dans l'élaboration du script! On lie le fichier Excel, enregistré en .CSV, au document de travail, puis on le fait rouler avec python. On établit ensuite les éléments de base. Par exemple de fonction comme *Input,* qui permettra à mon script de poser des questions à son utilisateur.

En faisait rouler le script du CSV une ligne à la fois, j'ai ensuite tenté d'insérer la fonction *Shuffle*, qui est très importante à mon travail, puisque c'est ce qui fait en sorte que les questions posées ne reviennent pas sans cesse. Première grosse difficulté donc atteinte : placer la fonction pour qu'elle roule correctement. Après maints essais ratés et frustrations multipliées, j'ai décidé de séparer mon document CSV en deux (un pour les questions, et un pour les réponses). Ceci, car je n'arrivais pas à mélanger à la fois les questions et les choix de réponse, tout en gardant ces derniers liés aux bonnes questions.

Réussir à lier les choix de réponses aux bonnes questions a été plus simple par la suite. Il a suffi de créer des fonctions *Define* pour déterminer ce qu'est une question et ce qu'est un choix de réponse. Ces fonctions ont par la suite été simplifiées grâce à l'aide de Jean-Hugues Roy.

```
41
42  for question in r1:
43          questions.append(question[0])
44
45  for reponse in r2:
46          reponses.append(reponse[0])
47          reponses.append(reponse[1])
48          reponses.append(reponse[2])
49          reponses.append(reponse[3])
50          reponses.append(reponse[4])
51
```

Pour être en mesure de mélanger, j'ai créé une bibliothèque contenant les possibilités de choix de réponses pour ensuite indiquer au script le bon moment pour les mélanger, mais seulement entre elles, pour ne pas avoir de choix en dehors de ce qu'on souhaite avoir. Après avoir mélangé ces choix, on crée une sous-liste pour déterminer seulement les choix qu'on veut imposer à la question.

```
def choixdereponses(i):
    global points
    rep = list(range((i-1)*5,(i-1)*5+5))
    random.shuffle(rep)
    print(reponses[rep[0]])
    print(reponses[rep[1]])
    print(reponses[rep[2]])
    print(reponses[rep[3]])
    print(reponses[rep[4]])
```

Au début, ces agencements ont dû être répétés une vingtaine de fois, dans le but de créer vingt questions sur cent réponses possibles. Après corrections, je me suis rendu compte qu'il aurait fallu que j'en fasse 100, à l'aide de *Copy/Paste.* Jean-Hugues Roy m'a vite convaincu qu'on pouvait simplifier la tâche, avec une simple formule, sortie de son cru :


« Voici ma formule : rep = list(range((i-1)*5,(i-1)*5+5)) » - JHR 2017

Ensuite, il y a eu des problèmes avec la fonction qui donne des points et qui dit quel résultat a été obtenu à la fin du questionnaire. Car, avec la formule implantée grâce à JHR, les bonnes réponses ne se compilaient plus. Ce qui s'écrivait au départ comme suit : « if choix == reponses[0]: » a été transformé en la formule suivante : « if choix == reponses[(i-1)*5]: ». Une logique qui a du sens, après s'être gratté la tête pendant une heure à essayer de comprendre « c'est quoi le *&?%$ de problème?! ».

Vers la fin, il reste les classiques *sapristis* de petits problèmes qu'on n'a pas envie de régler. Dans ce cas-ci, c'était que c9.io refuse de lire les caractères québécois, comme des phrases contenant des « é » ou des « à ». Ce problème, à ce jour, n'a toujours pas été réglé. Après quelques tentatives d'enregistrement de mes CSV en utf-8 ou encore de changer mon code en utf-16, j'ai du me faire à l'idée que mes questions et mes réponses ne peuvent pas contenir d'accents.

Malgré ce court *Making of*, c'est à noter que la réalisation de ce projet a pris au moins une vingtaine d'heures. Seulement l'écriture des cent questions a pris environ six ou sept heures. Beaucoup de ces heures peuvent être résumées par cette image :