

Projet de programmation en Python

Introduction à l'étude en TI au cégep

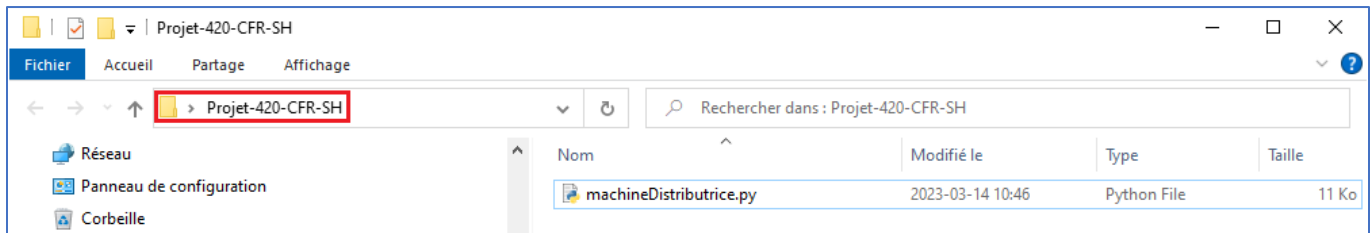
Partie 1. Mise en place du projet

Objectif

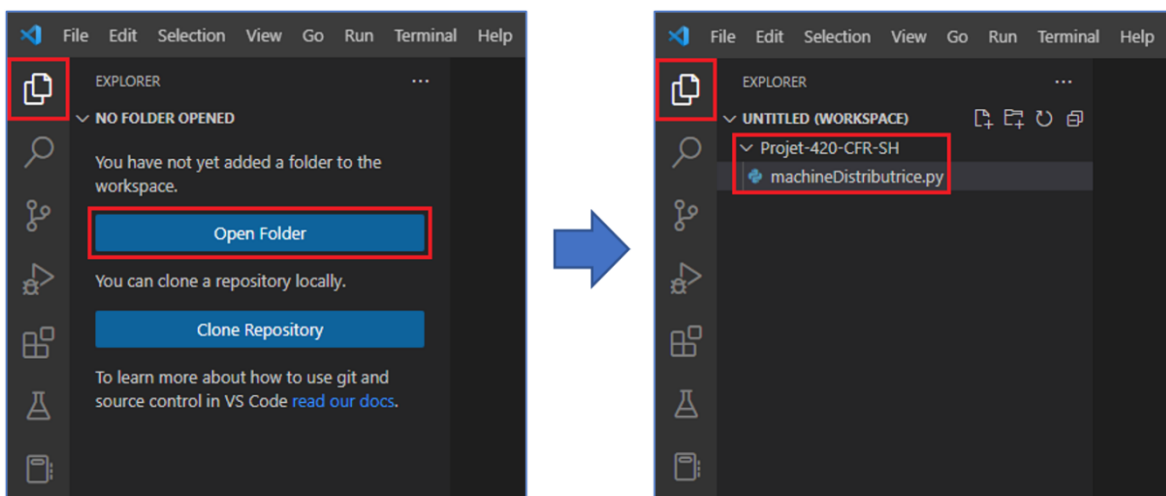
- Préparer l'environnement de développement en vue de programmer l'application en Python.

Préparation de l'environnement de programmation

Récupérez le fichier « machineDistributrice.py » qui a été fourni avec cet énoncé et placez-le dans un dossier « Projet-420-CFR-SH » que vous devez créer sur votre ordinateur. **Assurez-vous qu'aucun caractère accentué ou espace n'est présent dans votre nom de dossier.**



Ouvrez *VS Code* et ouvrez l'explorateur (touches [CTRL] + [SHIFT] + [E] ou icône représentant deux feuilles superposées) dans le menu de gauche (« activity bar »). Appuyez ensuite sur le bouton « Open Folder », puis sélectionnez le dossier créé à l'étape précédente et qui contient le fichier « machineDistributrice.py ». Ce dossier ainsi que ce fichier devraient alors s'afficher dans l'espace de travail de *VS Code* comme illustré ci-dessous. Si vous avez déjà un dossier d'ouvert vous pouvez le changer à partir du menu : « **File > Open Folder...** » ou avec les touches « **Ctrl+O** » simultanément.



Vous n'aurez pas à créer d'autres fichiers pour ce projet, toute la programmation se fera dans le fichier « machineDistributrice.py ».

Objectifs

- Planifier un projet d'équipe avec *Trello*.
- Collaborer avec un outil de gestion des versions comme *Git* et *GitHub*.

Planification avec *Trello*

En équipe de deux ou trois, vous devez planifier sur *Trello* la réalisation de votre projet. **Les équipes de trois doivent être préalablement validées par l'enseignant(e)**. Chaque tâche de la liste suivante devra être inscrite dans un tableau *Trello* et assignée à un membre de votre équipe. Dans votre tableau *Trello*, il faudra bien voir le nom de la tâche et le nom du membre de l'équipe qui réalise cette tâche.

Une répartition équitable des tâches entre tous les membres de l'équipe est attendue. Pour une équipe de deux, chaque membre doit avoir deux tâches faciles, une tâche moyenne et une tâche difficile. **Pour une équipe de trois**, deux membres doivent avoir chacun une tâche facile et une tâche difficile, alors que le troisième membre doit avoir deux tâches moyennes. La tâche restante (tâche facile) doit être faite ensemble par tous les membres.

- A. Saisir un code de produit valide – difficile
- B. Modifier la quantité d'un produit – facile
- C. Afficher les détails d'un produit – facile
- D. Insérer des pièces de monnaie – facile
- E. Payer un produit – moyen
- F. Distribuer un produit payé – facile
- G. Calculer la monnaie à rendre – difficile
- H. Remettre l'argent au client – moyen

De plus, votre tableau *Trello* devra contenir le flux suivant :

1. Tâches à réaliser
2. Tâches complétées

À chaque cours de projet, vous devrez choisir des tâches dans la liste « Tâches à réaliser » et les déplacer dans la liste « Tâches complétées ». **Autrement dit, vous devrez utiliser au moins deux périodes de cours pour la réalisation du projet et votre tableau *Trello* devra évoluer comme suit :**

| Cours de projet | Moment de la période de cours | Disposition dans le tableau <i>Trello</i> |
|--|-------------------------------|--|
| #1 Groupe 4418 : 29 mars Groupe 4419 : 31 mars | Début de la période | Toutes les tâches se retrouvent dans la liste « Tâches à réaliser ». |
| | Fin de la période | Une partie des tâches se retrouve dans la liste « Tâches à réaliser » et une autre dans la liste « Tâches complétées ». Or, plus les périodes de projet passent, plus le nombre de tâches dans la liste « Tâches à réaliser » diminue au profit du nombre de tâches dans la liste « Tâches complétées ». |
| #2 Groupe 4418 : 12 avril Groupe 4419 : 14 avril | Début de la période | Toutes les tâches se retrouvent dans la liste « Tâches à réaliser ». |
| | Fin de la période | Toutes les tâches se retrouvent dans la liste « Tâches complétées ». |

Dans un document *Word* à la fin du projet, vous devrez donc remettre trois captures d'écrans de votre tableau *Trello* ayant évolué :

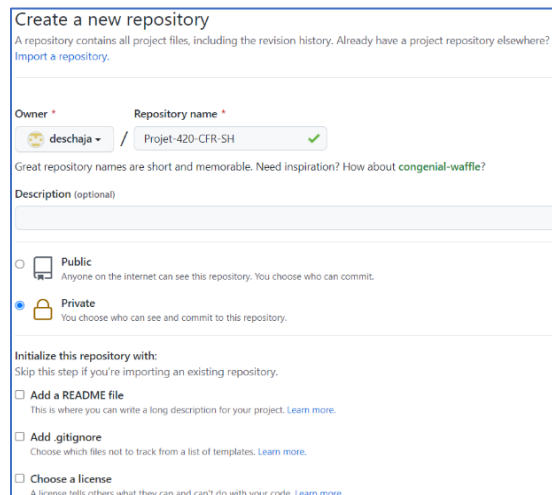
- Une capture d'écran au début du premier cours.
- Une capture d'écran à la fin du premier cours.
- Une capture d'écran au début du second cours.
- Une capture d'écran à la fin du second cours.

Dans ces captures d'écran, il faudra clairement voir :

- Les tâches qui restent encore à réaliser
- Les tâches qui ont été déplacées dans la liste « Tâches complétées »
- Les noms de chaque membre de l'équipe assignés aux différentes tâches
- La date de la période de cours concernée (cette date peut être ajoutée dans le document *Word* plutôt que dans le tableau *Trello*).

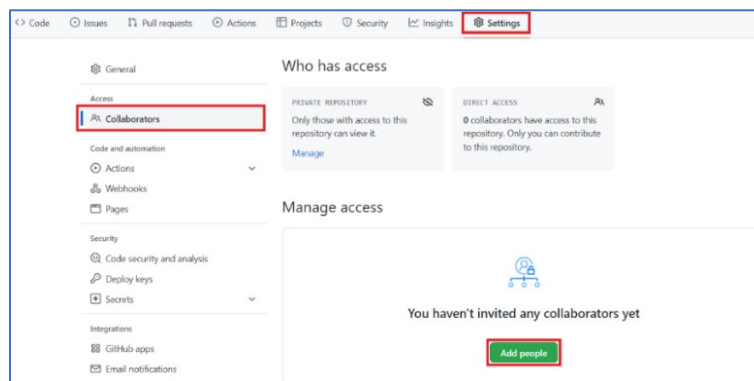
Soumission du code avec *Git* et *GitHub*

Sur le compte *GitHub* d'un des membres de votre équipe, créez un dépôt distant privé nommé « Projet-420-CFR-SH » (sans les guillemets).

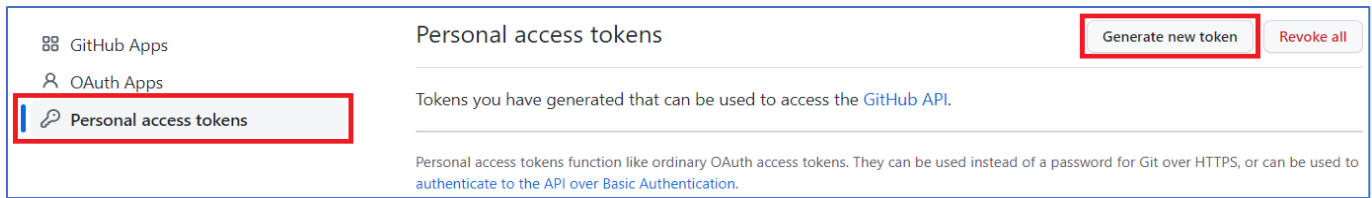


Une fois votre dépôt distant créé, cliquez sur l'onglet « Settings », puis choisissez l'option « Collaborators ». Confirmer votre accès en mode « sudo » puis, cliquez sur « Add people ». Ajoutez les courriels des membres de votre équipe **de même que le nom d'utilisateur *GitHub* de l'enseignant(e)** :

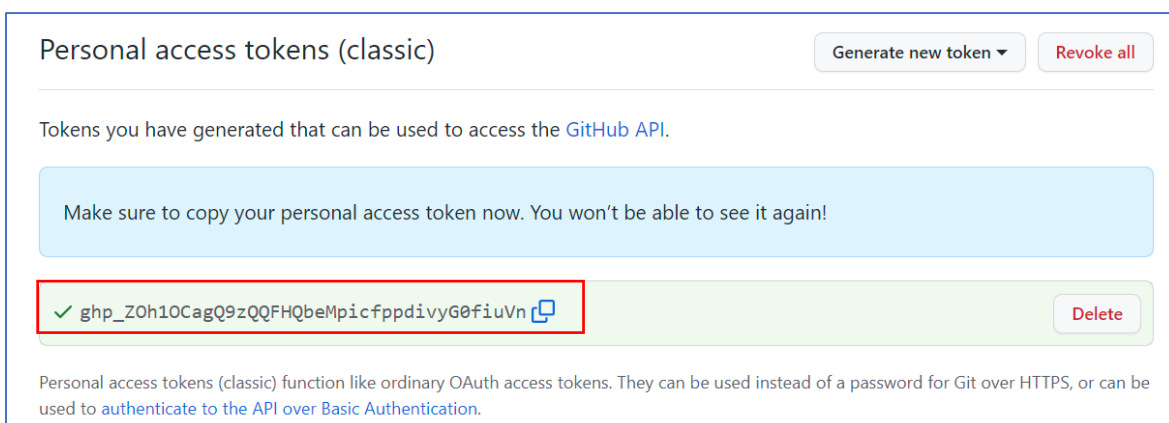
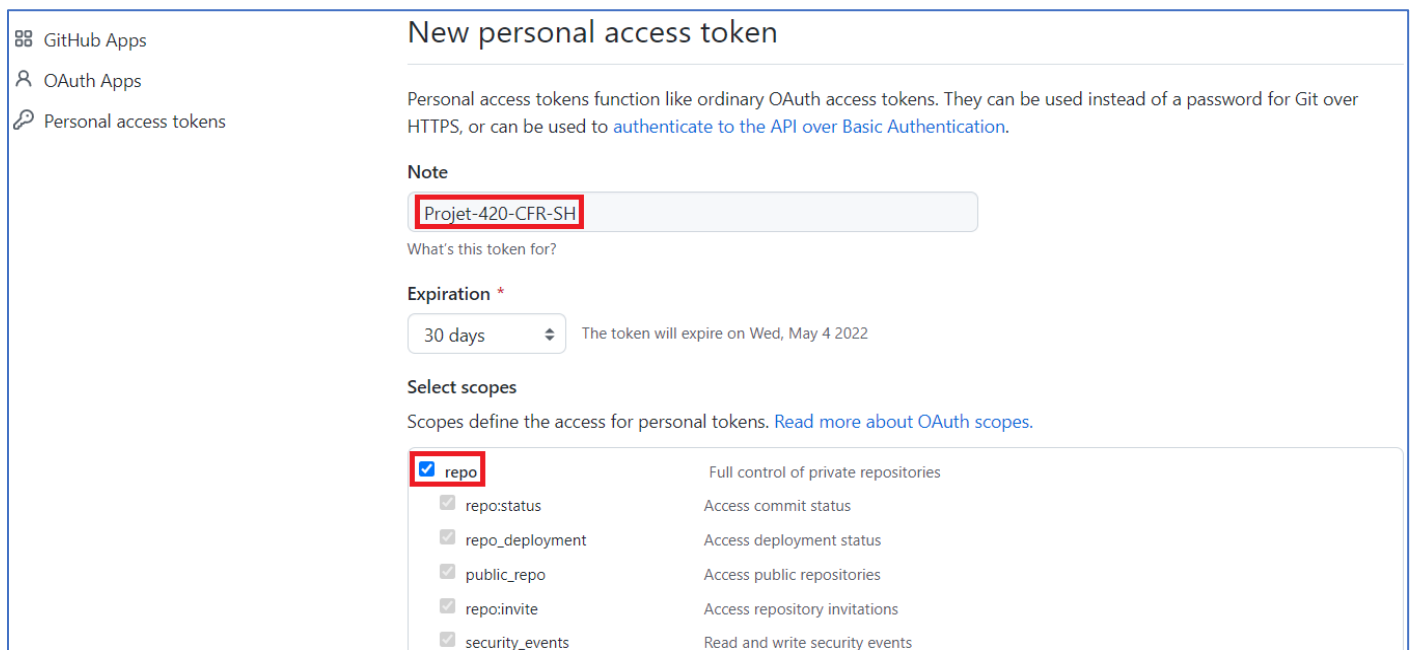
- **deschaja** (pour le groupe 4418)
- **mich-tr** (pour le groupe 4419)



Cliquez sur l'icône de votre compte *GitHub* dans le coin supérieur droit de l'écran et sélectionnez l'option « Settings » dans le menu ayant apparue. Cliquez ensuite sur l'option « Developer settings » complètement en bas du menu à gauche. Dans la fenêtre qui s'affiche, cliquez sur « Personal access tokens », puis sur « Tokens (classic) ». Cliquez ensuite sur le bouton « Generate new token ».



Nommez votre nouveau *token* « Projet-420-CFR-SH » (sans les guillemets), laissez l'expiration de 30 jours et cochez la case « repo ». Au bas de la page, cliquez ensuite sur le bouton « Generate token » et **prenez en note le *token* qui apparaît**.

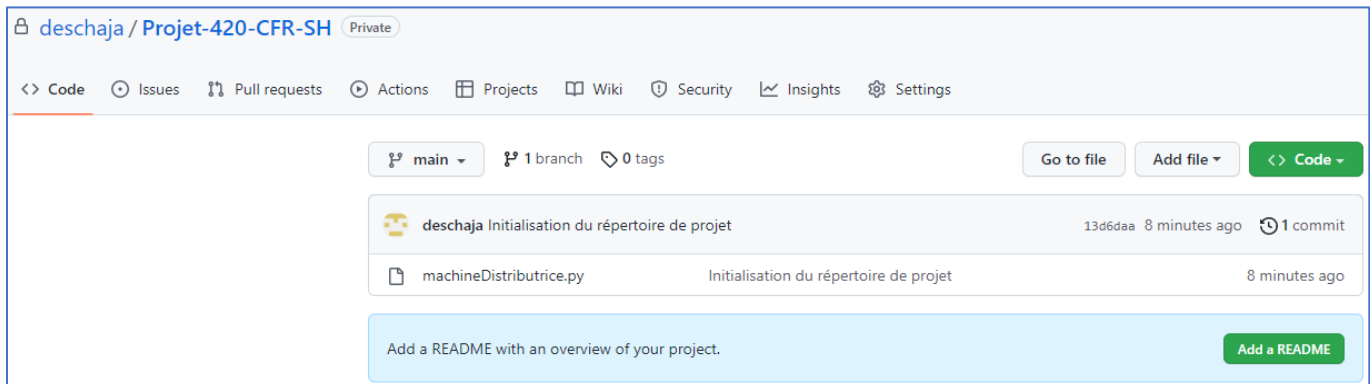


Dans le dossier « Projet-420-CFR-SH » **sur l'ordinateur du membre de l'équipe qui a créé le dépôt distant sur *GitHub***, entrez les commandes ci-après dans un terminal pour initialiser votre répertoire de travail avec *Git* et le lier à votre dépôt distant.

*** Attention !** Entrez les commandes ci-dessous une à la fois et remplacez les termes [NOM_UTILISATEUR_GITHUB] et [COURRIEL_UTILISATEUR_GITHUB] par votre propre nom d'utilisateur et votre propre courriel que vous avez utilisés pour vous inscrire sur GitHub. Pour exécuter la dernière commande (git push -u origin main), vous aurez besoin d'inscrire votre nom d'utilisateur de même que le token que vous avez précédemment généré et ce, à la place du mot de passe lorsqu'il vous sera demandé.

```
git init
git config --global user.name [NOM_UTILISATEUR_GITHUB]
git config --global user.email [COURRIEL_UTILISATEUR_GITHUB]
git add --all
git commit -m "Initialisation du répertoire de projet"
git branch -M main
git remote add origin https://github.com/[NOM_UTILISATEUR_GITHUB]/Projet-420-CFR-SH.git
git push -u origin main
```

Vous devriez maintenant être en mesure de voir le fichier « machineDistributrice.py » du côté de votre dépôt distant *GitHub* comme le montre d'ailleurs la capture d'écran ci-après.

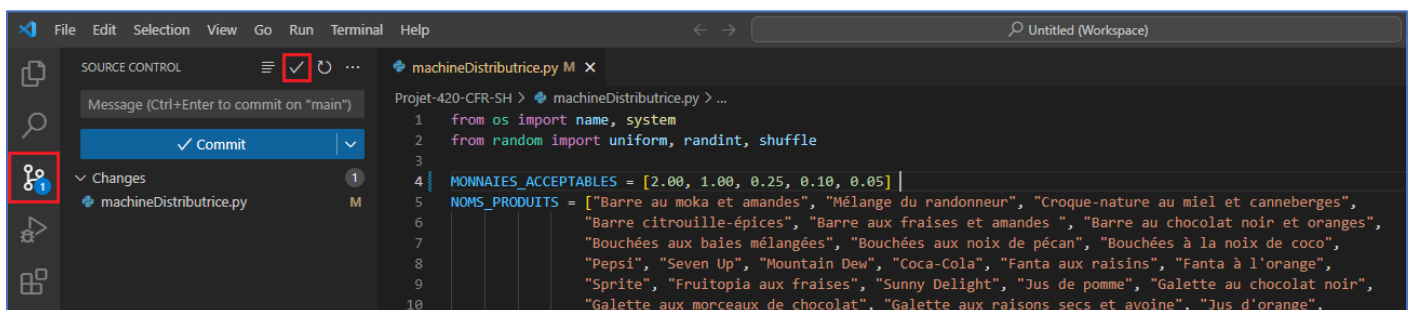


Une fois que les autres membres de l'équipe auront accepté les invitations à collaborer générées plus tôt dans *GitHub*, ils pourront récupérer le projet sur leur propre ordinateur en exécutant les commandes suivantes de leur côté.

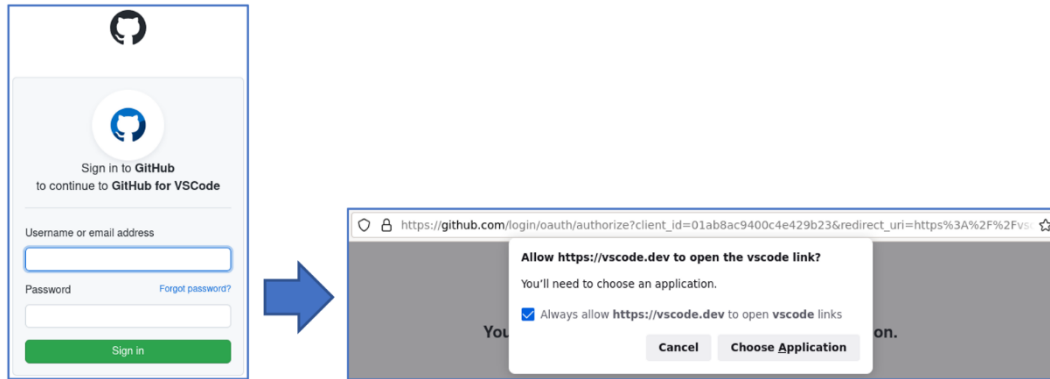
*** Ici encore, remplacez les termes [NOM_UTILISATEUR_GITHUB] et [COURRIEL_UTILISATEUR_GITHUB] par votre propre nom d'utilisateur et votre propre courriel que vous avez utilisés pour vous inscrire sur GitHub.**

```
git init
git config --global user.name [NOM_UTILISATEUR_GITHUB]
git config --global user.email [COURRIEL_UTILISATEUR_GITHUB]
git clone https://github.com/[NOM_UTILISATEUR_GITHUB]/Projet-420-CFR-SH.git
```

Ainsi, tous les membres de votre équipe peuvent contribuer au développement du code de votre projet et soumettre les changements vers le dépôt distant *GitHub*. Ils pourront même le faire directement dans *VS Code* en sélectionnant l'onglet à gauche encadré en rouge dans la figure ci-après et en cliquant ensuite sur le crochet dans le haut de l'interface.



Lors de votre toute première tentative de soumission *Git* dans *VS Code*, vous devrez cependant vous connectez à votre compte *GitHub* (*VS Code* vous le demandera automatiquement) et accepter que *VS Code* puisse gérer votre dépôt distant.



À la fin de chaque période de cours passée sur le projet, le code réalisé devra être soumis sur le dépôt distant *GitHub*. L'enseignant validera que les différentes soumissions ont bien été effectuées via l'historique du dépôt distant *GitHub*. Chaque membre de l'équipe devra donc s'assurer, durant le projet, de faire usage de *Git* et *GitHub*.

| Cours de projet | Moment de la période de cours | Disposition dans le dépôt <i>GitHub</i> |
|---|-------------------------------|--|
| #1 Groupe 4418 : 29 mars Groupe 4419 : 31 mars | Fin de la période | Chaque membre de l'équipe doit soumettre sur <i>GitHub</i> les fonctionnalités qu'il a développées dans le code Python durant ce cours. Pour une équipe de trois membres, trois soumissions (chacune identifiée au nom d'un membre différent de l'équipe) devraient, au minimum, avoir lieu. |
| #2 Groupe 4418 : 12 avril Groupe 4419 : 14 avril | Fin de la période | Ici encore, chaque membre de l'équipe devra soumettre sur <i>GitHub</i> les fonctionnalités qu'il a développées dans le code Python durant ce cours. À la fin de ce cours, le code devrait être final ou en voie de l'être. Si le code n'est pas encore terminé, une troisième série de soumissions (une par membre de l'équipe) devrait avoir lieu plus tard afin que le code final se retrouve du côté de <i>GitHub</i> . |

** Le code qui sera récupéré par l'enseignant(e) pour la correction sera celui présent sur GitHub. Ce faisant, assurez-vous que la version finale de ce code a bien été soumise sur GitHub pour la remise du projet.*

Partie 3. Programmation des fonctionnalités en Python

Objectif

- Programmer, en langage Python, les fonctionnalités relatives à une machine distributrice.

Fonctionnement attendu pour le programme final

Il est préférable de lancer l'application finale dans une console Windows plutôt que dans le terminal de *VS Code* de façon à pouvoir mettre cette console en plein écran (puisque l'affichage du contenu de la machine distributrice nécessite une

certaine largeur). Pour ce faire, appuyez simultanément sur les touches [WIN] et [R], puis inscrivez « cmd » (sans les guillemets). Dans le terminal, tapez les commandes suivantes :

```
cd [CHEMIN_VERS_VOTRE_DOSSIER_DE_PROJET]
python machineDistributrice.py
```

* Remplacez le terme [CHEMIN_VERS_VOTRE_DOSSIER_DE_PROJET] par le chemin complet vers votre propre dossier dans lequel se retrouve le fichier « machineDistributrice.py ». Vous pouvez le copier à partir de l'explorateur Windows en cliquant sur la barre d'adresse et en utilisant les touches « Ctrl + C ». Pour le coller dans le terminal, il faut cliquer sur le bouton droit de la souris (les touches ctrl+v, ne fonctionnent pas dans un terminal).

```
Administrateur: C:\Windows\system32\cmd.exe
Microsoft Windows [version 10.0.19044.2604]
(c) Microsoft Corporation. Tous droits réservés.

C:\Windows\system32>cd C:\Users\deschaja\Desktop\Projet-420-CFR-SH
C:\Users\deschaja\Desktop\Projet-420-CFR-SH>python machineDistributrice.py
```

1. En lançant le programme, le contenu de la machine distributrice devrait apparaître sous la forme d'une grille où les rangées sont identifiées de A à J et où les colonnes sont numérotées de 1 à 4. Il est à noter que le contenu de la machine distributrice est réorganisé aléatoirement à chaque redémarrage du programme.

| # | 1 | 2 | 3 | 4 |
|---|-----------------------------------|--------------------------------------|-----------------------------------|--------------------------------------|
| A | Croustilles au ketchup | Fanta aux raisins | Croustilles à saveur BBQ | Jujubes aux pêches |
| B | Mélange du randonneur | Seven Up | Galette au chocolat noir | Barre aux fraises et amandes |
| C | Croustilles aux piments jalapenos | Croustilles au sel et vinaigre | Jujubes aux melons d'eau | Barre aux pistaches et pommes |
| D | Jus d'orange | Barre tendre aux arachides | Bouchées aux noix de pécan | Bretzels sucrés salés |
| E | Galette aux noix de macadame | Galette aux morceaux de chocolat | Coca-Cola | Croustilles de riz |
| F | Croustilles au cheddar | Bouchées à la noix de coco | Jus de pomme | Croustilles nature |
| G | Pepsi | Croque-nature au miel et canneberges | Barre au chocolat noir et oranges | Fanta à l'orange |
| H | Barre au moka et amandes | Fruitopia aux fraises | Sunny Delight | Mountain Dew |
| I | Barre citrouille-épices | Galette aux raisins secs et avoine | Croustilles au bacon à l'érable | Croustilles aux cornichons à l'aneth |
| J | Jujubes à la framboise bleue | Jujubes aux cerises | Bouchées aux baies mélangées | Sprite |

Veuillez entrer le code du produit voulu (ex: A1) :

2. L'utilisateur peut alors entrer le code du produit qu'il souhaite acheter en spécifiant la lettre de sa rangée suivie du chiffre de sa colonne. Si l'utilisateur entre un code invalide (rangée et/ou colonne inexistante(s)), un message d'erreur apparaît et il lui demande de recommencer.

| # | 1 | 2 | 3 | 4 |
|---|-----------------------------------|--------------------------------------|-----------------------------------|--------------------------------------|
| A | Croustilles au ketchup | Fanta aux raisins | Croustilles à saveur BBQ | Jujubes aux pêches |
| B | Mélange du randonneur | Seven Up | Galette au chocolat noir | Barre aux fraises et amandes |
| C | Croustilles aux piments jalapenos | Croustilles au sel et vinaigre | Jujubes aux melons d'eau | Barre aux pistaches et pommes |
| D | Jus d'orange | Barre tendre aux arachides | Bouchées aux noix de pécan | Bretzels sucrés salés |
| E | Galette aux noix de macadame | Galette aux morceaux de chocolat | Coca-Cola | Croustilles de riz |
| F | Croustilles au cheddar | Bouchées à la noix de coco | Jus de pomme | Croustilles nature |
| G | Pepsi | Croque-nature au miel et canneberges | Barre au chocolat noir et oranges | Fanta à l'orange |
| H | Barre au moka et amandes | Fruitopia aux fraises | Sunny Delight | Mountain Dew |
| I | Barre citrouille-épices | Galette aux raisins secs et avoine | Croustilles au bacon à l'érable | Croustilles aux cornichons à l'aneth |
| J | Jujubes à la framboise bleue | Jujubes aux cerises | Bouchées aux baies mélangées | Sprite |

Veuillez entrer le code du produit voulu (ex: A1) : B8
 Le code de produit est invalide. Veuillez réessayer.
 Veuillez entrer le code du produit voulu (ex: A1) :

Si l'utilisateur entre un code de produit valide, mais que le produit est déjà écoulé (la quantité du produit vaut 0), un message d'erreur apparaît et il lui est demandé de recommencer.

| # | 1 | 2 | 3 | 4 |
|---|------------------------------------|-----------------------------------|-----------------------------------|--------------------------------------|
| A | Coca-Cola | Fruitopia aux fraises | Jujubes aux pêches | Barre aux pistaches et pommes |
| B | Croustilles nature | Croustilles au sel et vinaigre | Jujubes à la framboise bleue | Fanta à l'orange |
| C | Jus d'orange | Barre tendre aux arachides | Jus de pomme | Croque-nature au miel et canneberges |
| D | Galette au chocolat noir | Bouchées aux noix de pécan | Pepsi | Croustilles aux cornichons à l'aneth |
| E | Croustilles à saveur BBQ | Mountain Dew | Barre aux fraises et amandes | Jujubes aux cerises |
| F | Jujubes aux melons d'eau | Bouchées aux baies mélangées | Sunny Delight | Galette aux noix de macadame |
| G | Croustilles au ketchup | Barre au chocolat noir et oranges | Bouchées à la noix de coco | Fanta aux raisins |
| H | Bretzels sucrés salés | Croustilles de riz | Croustilles au cheddar | Barre citrouille-épices |
| I | Croustilles au bacon à l'érable | Barre au moka et amandes | Sprite | Mélange du randonneur |
| J | Galette aux raisins secs et avoine | Seven Up | Croustilles aux piments jalapenos | Galette aux morceaux de chocolat |

Veuillez entrer le code du produit voulu (ex: A1) : G3
 Ce produit est épuisé. Veuillez en choisir un autre.
 Veuillez entrer le code du produit voulu (ex: A1) :

Si l'utilisateur entre un code de produit valide et que le produit n'est pas encore écoulé (la quantité du produit est supérieure à 0), les détails du produit correspondant à ce code (nom, prix et quantité disponible) sont affichés. Le montant que l'utilisateur pourra ensuite insérer est affiché.

```
Produit choisi : Bouchées aux baies mélangées
Prix unitaire : 2.10 $
Quantité restante : 11

Montant inséré : 0.00 $
```

3. L'utilisateur peut insérer des pièces de 2.00 \$, 1.00 \$, 0.25 \$, 0.10 \$ et 0.05 \$ en appuyant sur des touches spécifiques du clavier. Tant que l'argent inséré n'atteint pas ou ne dépasse pas le prix du produit, l'utilisateur peut continuer à ajouter de la monnaie. Lorsque l'argent atteint ou dépasse le prix du produit, la quantité du produit diminue de 1 et la distribution de ce produit peut avoir lieu (le message « Distribution du produit en cours... » apparaît alors).

```
Produit choisi : Bouchées aux baies mélangées
Prix unitaire : 2.10 $
Quantité restante : 10

Montant inséré : 2.25 $

Distribution du produit en cours...
(Appuyez sur [ENTER] pour continuer)
```

4. En appuyant sur la touche [ENTER], l'argent inséré et le prix du produit sont à nouveau affichés. L'argent à rendre (différence entre l'argent inséré et le prix du produit) apparaît également de même que sa répartition en pièces de monnaies (combien de pièces de monnaie de chaque type entre 2.00 \$, 1.00 \$, 0.25 \$, 0.10 \$ et 0.05 \$). La remise de l'argent à l'utilisateur se fait et le message « Remise des pièces de monnaie en cours... » s'affiche.

```
Montant inséré : 2.25 $
Prix du produit : 2.10 $
Argent à rendre : 0.15 $

1 x 0.10 $
1 x 0.05 $

Remise des pièces de monnaie en cours...
(Appuyez sur [ENTER] pour continuer)
```

5. En appuyant sur la touche [ENTER], un message de remerciement reprenant le nom du produit acheté par l'utilisateur apparaît. Il suffit alors d'appuyer à nouveau sur la touche [ENTER] pour revenir à l'affichage du contenu de la machine distributrice et pouvoir, ainsi, répéter toutes les étapes précédentes (donc, pour pouvoir acheter un nouveau produit).

```
Merci d'avoir utilisé notre machine distributrice!
Nous espérons que vous aimerez le produit acheté :

Bouchées aux baies mélangées

(Appuyez sur [ENTER] pour continuer)
```

Critères de programmation en fonction des tâches

Dans cette partie du projet, vous devrez compléter, en Python, le programme fourni pour la machine distributrice. **Ce programme, une fois achevé, devra reproduire le plus fidèlement possible ce qui a été mentionné et illustré dans la partie précédente.**

TÂCHE A : SAISIR UN CODE DE PRODUIT VALIDE — DIFFICILE

- La fonction « saisir_code_produit() » doit demander un code de produit à l'utilisateur.
- Le code de produit doit être redemandé s'il n'est pas valide ou encore si le produit correspondant à ce code est écoulé. La fonction « verifier_disponibilite_produit() » permettra de déterminer si le produit est écoulé ou non.
- Le code de produit est toujours constitué d'une lettre suivie d'un chiffre. La lettre doit exister pour les rangées de la machine distributrice et le numéro doit correspondre à une colonne de cette machine. Les lettres des rangées de la machine distributrice peuvent être récupérées avec l'instruction « machine_distributrice.keys() » (sans les guillemets).
- Cette fonction doit retourner le code inscrit par l'utilisateur s'il est valide et elle ne retourne pas tant que le code n'est pas valide.

TÂCHE B : MODIFIER LA QUANTITÉ D'UN PRODUIT — FACILE

- La fonction « modifier_quantite_produit() » doit vérifier que la quantité reçue en paramètre d'entrée est supérieure à 0.
- Si cette quantité est supérieure à 0, elle est assignée au bon produit en utilisant le code du produit également reçu en paramètre d'entrée. **Observez la ligne de code présente dans la fonction « obtenir_quantite_produit() » pour savoir comment accéder à la quantité d'un produit dans la machine distributrice en fonction de son code de produit.**
- Cette fonction ne retourne rien et ne fait rien si la quantité reçue en paramètre d'entrée est inférieure ou égale à 0.

TÂCHE C : AFFICHER LES DÉTAILS D'UN PRODUIT – FACILE

- La fonction « `afficher_details_produit()` » affiche le nom, le prix et la quantité du produit de la machine distributrice correspondant au code de produit reçu en paramètre d'entrée.
- Chaque information (nom, prix et quantité) doit être affichée sur une ligne distincte.
- Cette fonction ne retourne rien.

TÂCHE D : INSÉRER DES PIÈCES DE MONNAIE – FACILE

- La fonction « `inserer_argent()` » reçoit en paramètres d'entrées la touche enfoncée au clavier de même que le montant d'argent fourni jusqu'alors par l'utilisateur.
- Si la touche enfoncée est [Z], le montant d'argent fourni jusqu'alors doit être augmenté de 2.00 \$. Utilisez à cette fin l'index 0 de la liste « `MONNAIES_ACCEPTABLES` ».
- Si la touche enfoncée est plutôt [X], le montant d'argent fourni jusqu'alors doit être augmenté de 1.00 \$. Utilisez à cette fin l'index 1 de la liste « `MONNAIES_ACCEPTABLES` ».
- Si la touche enfoncée est plutôt [C], le montant d'argent fourni jusqu'alors doit être augmenté de 0.25 \$. Utilisez à cette fin l'index 2 de la liste « `MONNAIES_ACCEPTABLES` ».
- Si la touche enfoncée est plutôt [V], le montant d'argent fourni jusqu'alors doit être augmenté de 0.10 \$. Utilisez à cette fin l'index 3 de la liste « `MONNAIES_ACCEPTABLES` ».
- Si la touche enfoncée est plutôt [B], le montant d'argent fourni jusqu'alors doit être augmenté de 0.05 \$. Utilisez à cette fin l'index 4 de la liste « `MONNAIES_ACCEPTABLES` ».
- Cette fonction doit retourner le montant d'argent fourni par l'utilisateur.

TÂCHE E : PAYER UN PRODUIT – MOYEN

- La fonction « `payer_produit()` » doit récupérer le prix du produit correspondant au code de produit reçu en paramètre d'entrée.
- Elle doit initialiser à 0 le montant d'argent que pourra ensuite fournir l'utilisateur.
- Tant que le montant d'argent fourni par l'utilisateur n'atteint pas ou ne dépasse pas le prix du produit, il faut :
 - Effacer l'affichage produit à l'aide de la fonction « `effacer_ecran()` ».
 - Afficher les détails du produit concerné avec la fonction « `afficher_details_produit()` ».
 - Afficher le montant que l'utilisateur pourra insérer (message « Montant inséré : x.xx \$ »).
 - Récupérer la touche enfoncée au clavier par l'entremise de la méthode « `saisir_touche()` ».
 - Appeler la fonction « `inserer_argent()` » et récupérer le montant d'argent à la sortie de cette fonction.
- Cette fonction doit produire l'affichage suivant (dans cet exemple, l'utilisateur a inséré une pièce de monnaie de 1.00 \$ dans la machine distributrice) :

```
Produit choisi   : Croque-nature au miel et canneberges
Prix unitaire   : 1.60 $
Quantité restante : 10
Montant inséré : 1.00 $
```

- La fonction « `payer_produit()` » doit retourner le montant d'argent fourni par l'utilisateur.

TÂCHE F : DISTRIBUER UN PRODUIT PAYÉ – FACILE

- La fonction « `distribuer_produit()` » doit mettre à jour la quantité du produit concerné en appelant, la fonction « `modifier_quantite_produit()` », parce qu'il y a maintenant un produit de moins dans la machine distributrice (l'utilisateur l'ayant acheté).
- Cette fonction doit reprendre le même affichage généré dans la tâche E, mais, en plus, elle doit faire apparaître le message « Distribution du produit en cours... ».
- La fonction Python « `input()` » avec le message « (Appuyez sur [ENTER] pour continuer) » doit être utilisée. L'affichage suivant devra donc être produit. Dans cet exemple, l'utilisateur achète un jus de pomme à 0.20 \$ en fournissant 0.25 \$:

```

Produit choisi   : Jus de pomme
Prix unitaire    : 0.20 $
Quantité restante : 5

Montant inséré : 0.25 $

Distribution du produit en cours...
(Appuyez sur [ENTER] pour continuer)

```

- La fonction « distribuer_produit() » ne retourne rien.

TÂCHE G : CALCULER LA MONNAIE À RENDRE – **DIFFICILE**

- La fonction « calculer_monnaie() » reçoit, comme paramètres d'entrées, l'argent à rendre (calculé à l'extérieur de cette fonction) et une liste des différentes pièces de monnaie qu'il faudra remettre à l'utilisateur. **Cette liste doit être vide lorsqu'elle est fournie à la fonction.**
- Cette fonction doit remplir la liste « liste_monnaies_rendues » jusqu'à atteindre le montant dicté par « argent_a_rendre » en se fiant aux pièces de monnaie inscrites dans la liste « MONNAIES_ACCEPTABLES ». Par exemple, si le produit coûte 1.70 \$ et l'utilisateur insère 3.55 \$ dans la machine distributrice, il faudra lui rendre une pièce de 1.00 \$, trois pièces de 0.25 \$ et une pièce de 0.10 \$:

| | 2.00 \$ | 1.00 \$ | 0.25 \$ | 0.10 \$ | 0.05 \$ |
|--|---------|---------|---------|---------|---------|
| Quantité de chaque pièce de monnaie à rendre | 0 | 1 | 3 | 1 | 0 |

La liste « liste_monnaies_rendues » doit contenir la quantité de chaque pièce de monnaie à rendre, ce qui reviendrait à la liste suivante selon l'exemple présenté ici :

```
liste_monnaies_rendues = [0, 1, 3, 1, 0]
```

- La fonction « calculer_monnaie() » ne doit rien retourner. Puisque la liste « liste_monnaies_rendues » a été passée en paramètre d'entrée de cette fonction, toutes les modifications apportées à cette liste seront également effectives à l'extérieur de la fonction.

TÂCHE H : REMETTRE L'ARGENT AU CLIENT – **MOYEN**

- La fonction « remettre_argent() » doit récupérer le prix du produit correspondant au code de produit reçu en paramètre d'entrée.
- Elle doit calculer la somme à rendre à l'utilisateur, c'est-à-dire la différence entre l'argent fourni par cet utilisateur et le prix du produit.
- Elle doit déclarer une liste vide, soit « liste_monnaies_rendues », puis appeler la fonction « calculer_monnaie() » en lui fournissant cette liste.
- La fonction « remettre_argent() » doit ensuite accomplir ces actions :
 - Effacer l'affichage produit jusqu'à maintenant à l'aide de la fonction « effacer_ecran() ».
 - Afficher le montant d'argent fourni par l'utilisateur (message « Montant inséré : x.xx \$ »).
 - Afficher le prix du produit (message « Prix du produit : x.xx \$ »).
 - Afficher la somme à rendre à l'utilisateur (message « Argent à rendre : x.xx \$ »).
 - Afficher la quantité de chaque pièce de monnaie contenue dans la liste « liste_monnaies_rendues » et ce, sous la forme « [quantite] x [monnaie] \$ » (où [quantite] et [monnaie] doivent être respectivement remplacés par la quantité et la valeur de la pièce de monnaie concernée).
 - Afficher le message « Remise des pièces de monnaie en cours... ».
- La fonction Python « input() » avec le message « (Appuyez sur [ENTER] pour continuer) » doit être utilisée. L'affichage suivant devra donc être produit. Dans cet exemple, l'utilisateur achète un produit à 1.70 \$ en fournissant 3.55 \$. Il faut donc lui remettre une somme de 1.85 \$, soit une pièce de 1.00 \$, trois pièces de 0.25 \$ et une pièce de 0.10 \$:

```
Montant inséré : 3.55 $
Prix du produit : 1.70 $
Argent à rendre : 1.85 $

1 x 1.00 $
3 x 0.25 $
1 x 0.10 $

Remise des pièces de monnaie en cours...
(Appuyez sur [ENTER] pour continuer)
```

Partie 4. Remise du projet

La remise de votre projet se fera sur deux plateformes :

- Sur LÉA (au même endroit où vous avez récupéré cet énoncé de projet), remettez un document Word contenant les captures d'écrans de votre tableau Trello.

** Assurez-vous que vos captures d'écrans illustrent bien l'avancement des tâches A à H tout au long des périodes de cours allouées pour le projet et qu'on y voit le nom de chaque membre de l'équipe assigné à ces tâches.*

- Sur votre dépôt distant *GitHub*, poussez (commande `git push`) votre code final « `machine_distributrice.py` ».

** Assurez-vous que tous les membres de votre équipe ont contribué au code et on fait des « push » (commande `git push`) à différents moments dans les périodes de cours allouées au projet, car l'enseignant(e) récupérera votre code sur cette plateforme et vérifiera l'historique des modifications. Vérifiez également que l'enseignant a accès à votre dépôt *GitHub* (il faut l'ajouter comme « contributor » à votre projet).*