**CC 1 Scripting Python**

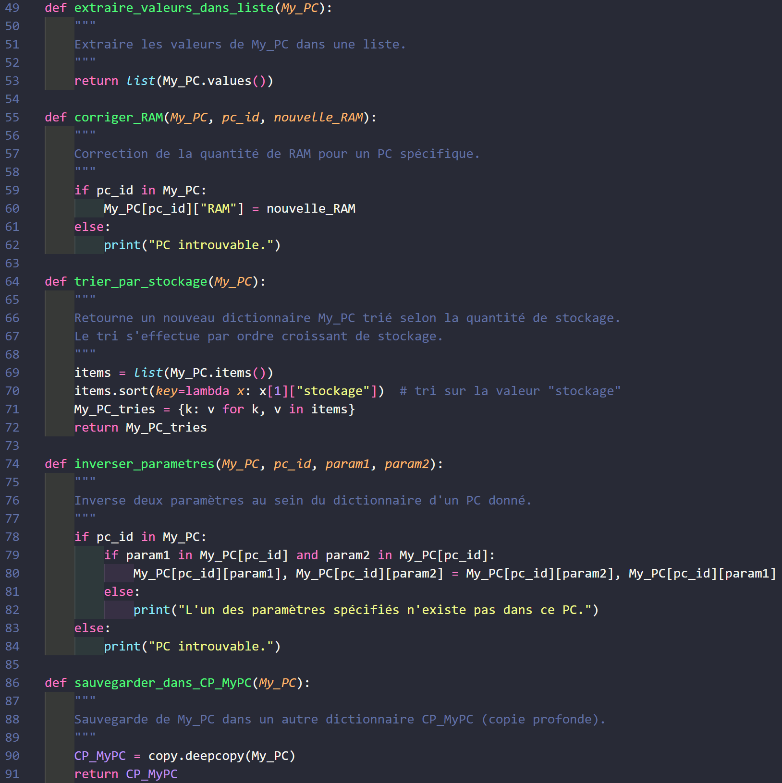
# Énoncé 1 : Manipulation de chaînes

En premier lieu, nous avons décidé de créer une chaîne de caractères MaChaine en répétant l’alphabet minuscule N fois (avec N saisi par l’utilisateur, et N ≥ 5). Après l’avoir affichée, nous avons extrait ses 17 derniers caractères dans une variable TaChaine. Ensuite, nous avons transformé MaChaine afin d’alterner entre minuscules et majuscules, avant de l’inverser et d’en afficher le résultat.

Nous avons également construit une pyramide de caractères à partir de TaChaine, ce qui nous a obligé à réfléchir à une représentation visuelle des données. Par la suite, nous avons recherché une sous-chaîne de taille paramétrable dans TaChaine et affiché sa taille, puis trié TaChaine et présenté le résultat sous forme de liste. Cette première étape nous a permis d’explorer un large éventail d’opérations sur les chaînes de caractère.

Une image contenant texte, capture d’écran

Description générée automatiquement

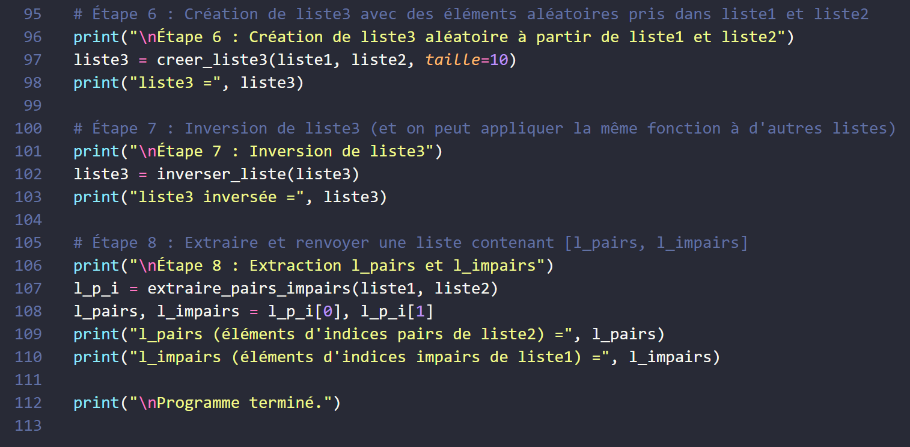




# Énoncé 2 : Manipulation de listes

Nous créons liste1 avec des entiers allant de 1 à 25 grâce à la fonction creer\_liste1(25). À partir de cette liste, liste2 est construite en extrayant simplement les nombres impairs avec extraire\_impairs(liste1).

Nous avons ensuite affiché liste1 et liste2 triées en ordre décroissant grâce à afficher\_listes\_decroissantes(liste1, liste2). Pour modifier liste1, nous avons utilisé ajouter\_elements(liste1, [99,100]) afin d’ajouter des éléments en fin de liste, puis ajouter\_elements(liste1, [50,51], index=5) pour insérer des valeurs à un emplacement précis.

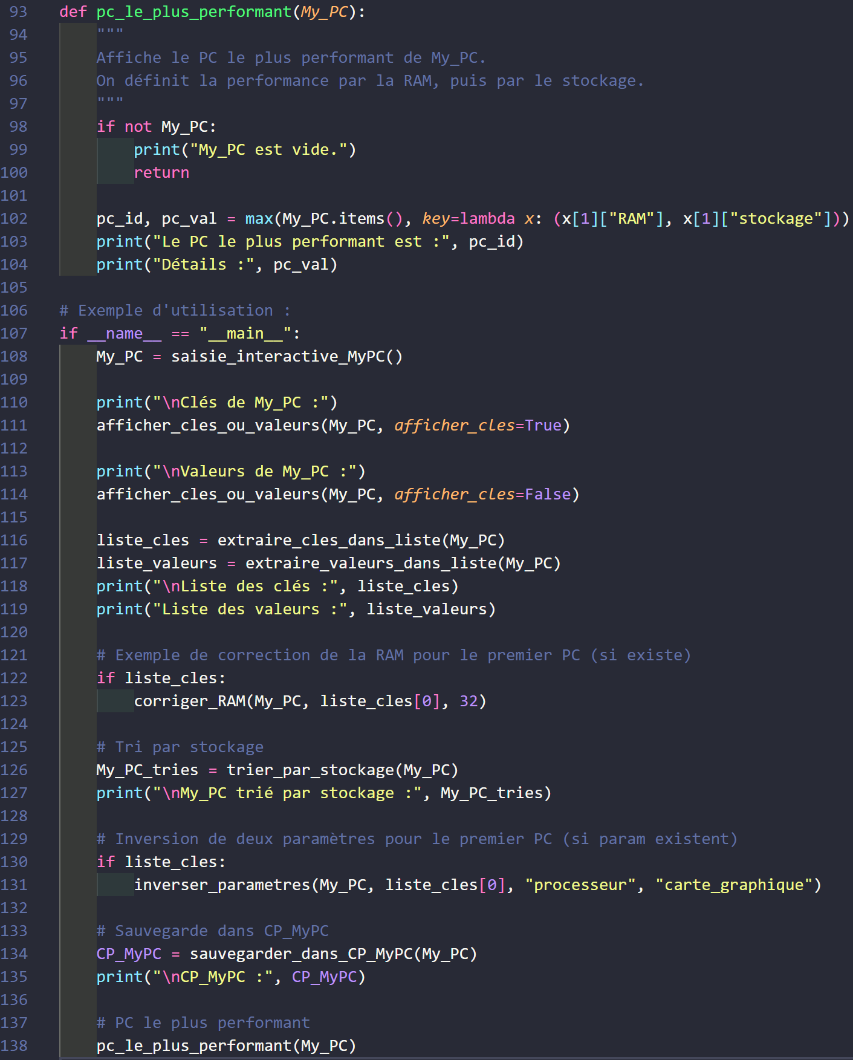
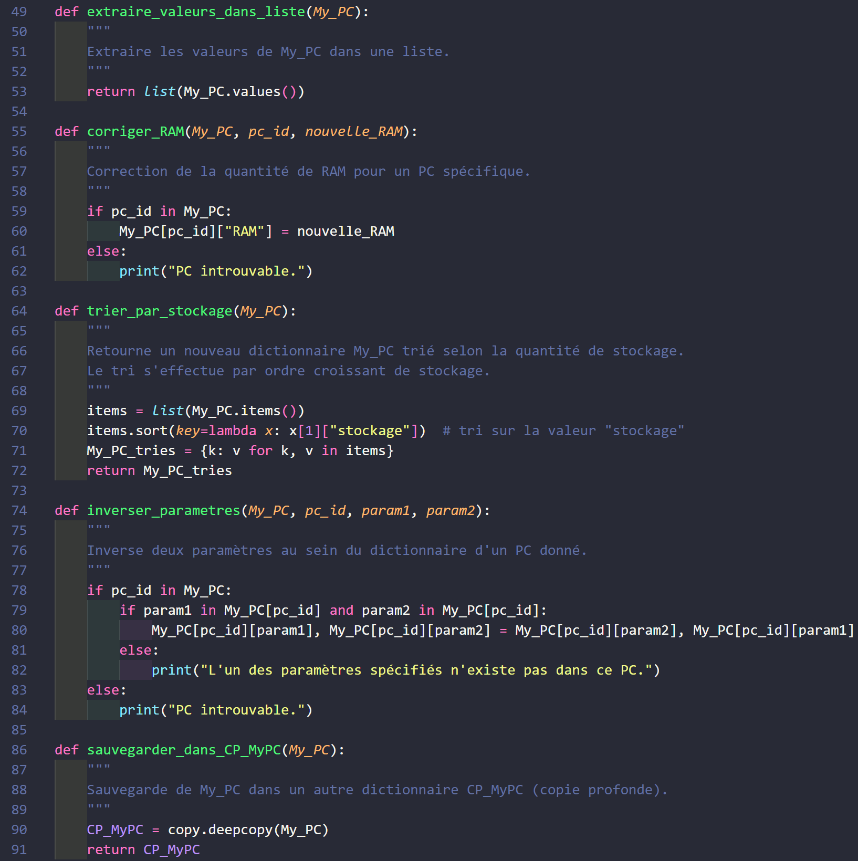
La suppression d’occurrences a été simplifiée en utilisant supprimer\_toutes\_occurrences(liste1, valeur\_a\_supprimer), qui renvoie une nouvelle version de liste1 sans la valeur spécifiée.  
Nous avons également créé une liste3 avec des éléments aléatoires de liste1 et liste2 grâce à creer\_liste3(liste1, liste2, taille=10). Après avoir inversé liste3 avec inverser\_liste(liste3), nous avons extrait, via extraire\_pairs\_impairs(liste1, liste2), deux sous-listes : l’une contenant les éléments de liste2 situés à des indices pairs, et l’autre les éléments de liste1 situés à des indices impairs.   
  
  
  


# Énoncé 3 : Manipulation de dictionnaires

L’utilisateur indique le nombre de PC à décrire, puis saisit leurs caractéristiques (processeur, RAM, stockage, carte graphique, OS) pour remplir le dictionnaire My\_PC. Les fonctions afficher\_cles\_ou\_valeurs nous permettent ensuite d’afficher soit les clés (identifiants de chaque PC), soit les configurations complètes.

Les clés et valeurs peuvent être extraites avec extraire\_cles\_dans\_liste(My\_PC) et extraire\_valeurs\_dans\_liste(My\_PC). Pour modifier la configuration, la fonction corriger\_RAM(My\_PC, pc\_id, nouvelle\_RAM) met à jour la RAM d’un PC spécifique. Avec trier\_par\_stockage(My\_PC), nous obtenons un nouveau dictionnaire trié selon la quantité de stockage, et inverser\_parametres(My\_PC, pc\_id, param1, param2) permet d’échanger deux paramètres internes d’un PC. Grâce à sauvegarder\_dans\_CP\_MyPC(My\_PC), nous pouvons effectuer une copie profonde de My\_PC dans CP\_MyPC, préservant ainsi l’état courant de nos données. Enfin, pc\_le\_plus\_performant(My\_PC) identifie et affiche le PC offrant la meilleure combinaison RAM/stockage.

****

  
  
  
Pour conclure je dirais que le temps accordé pour le CC était très court en plus d’être en période d’entreprise ça a été compliqué de finir dans les temps, je vous joint aussi le [lien github](https://github.com/Wilylejardinier/CC-1-Scripting-Python-) du CC.