1 Problématique

Le code 1 lève une erreur.

```
1 def une_fonction():
2       a = 3
3
4 une_fonction()
5 print(a)
```

Code 1 – Variable locale

À l'inverse le code 2 s'exécute sans erreur.

```
1 a = 3
2 def une_fonction():
3     print(a)
4
5 une_fonction()
```

Code 2 – Variable globale

Quelle est la portée d'une variable ?

2 Portée d'une variable

C'est le domaine d'existence d'une variable, c'est à dire la partie du programme où elle peut être utilisée.

2.1 Variable locale

À retenir

Une variable locale n'est utilisable que dans le bloc où elle a été définie.

Activité 1 :

1. Se rendre sur http://pythontutor.com/ et cliquer sur *Start visualizing your code now.*
2. Écrire le code 1 puis cliquer sur *Visualize Execution.*
3. Exécuter le code en *pas à pas* avec le bouton *Next.*
4. Observer la période d'existence de la variable *a* et justifier alors l'erreur d'exécution de la ligne 5.

2.2 Variable globale

Christophe Viroulaud

À retenir

Une variable globale est accessible depuis n'importe quel point du programme.

Activité 2 : Dérouler le code 2 sur *pythontutor* et justifier la bonne exécution.

3 Paramètres et variables d'une fonction

```
1 def cube(x):
2       resultat = x*x*x
3       return resultat
4
5 cube(4)
```

Code 3 – Fonction qui calcule le cube d'un nombre

Dans le code 3, x est un *paramètre* de la fonction *cube*. Cela signifie que lors de l'appel de la fonction dans le programme principal (ligne 5), il faut lui passer un *argument* qui remplacera x dans l'exécution du code de la fonction ; ici on passe l'argument *4*.
La variable *resultat* est une variable *locale* à la fonction. Il n'est pas possible d'accéder à cette variable en-dehors de la fonction *cube*.

À retenir

La fonction *cube* ne renvoie pas la variable *resultat* mais le contenu de cette variable.

Activité 3 : Une boutique de vêtements effectue des soldes sur ses invendus. Elle définit les remises :

- 30% sur les étiquettes rouges,
- 40% sur es étiquettes vertes,
- 50% sur les étiquettes bleues.

Écrire une fonction **remise** qui calcule la valeur en € de la remise effectuée en fonction du prix du vêtement et de la couleur de l'étiquette.

4 Pièges et bonnes pratiques

4.1 Documenter une fonction

Une fonction est un code autonome qui réalise une action. L'utilisateur de la fonction ne veut pas connaître l'implémentation du corps de la fonction mais seulement pouvoir l'utiliser correctement.

À retenir

La **docstring** détaille la manière d'utiliser une fonction. C'est un commentaire spécifique placé sous la signature de la fonction. On accède à cette documentation via la commande **help**.

Christophe Viroulaud

4.2 Typer les paramètres d'une fonction

Une fonction attend des paramètres d'un type précis. Il est possible de préciser le type attendu.

```
1 # La fonction attend un entier et une chaîne de caractère et renvoie un
     flottant
2 def calcul_remise(prix: int, remise: str)->float:
```

Code 4 – Typer les paramètres d'une fonction

À retenir

Typer les variables d'un programme ou les paramètres d'une fonction permet d'obtenir un code plus clair. Ce typage n'est qu'indicatif.

4.3 Effet de bord

Les élèves de NSI ont obtenu les notes suivantes lors du dernier devoir. Avant de les enregistrer définitivement dans Pronote, l'enseignant désire simuler des bonifications pour favoriser les élèves les plus en difficultés.

```
1  def simulation_bonification(limite):
2      for i in range(len(notes)):
3          if notes[i] <= limite:
4              notes[i] += 2
5      return notes
6
7  notes=[7,12,8,5,19,10,7,6,1,15,13,8]
8  print(notes)
9  print(simulation_bonification(6))
10 print(simulation_bonification(8))
```

Code 5 – Bonification

Le code 5 effectue une bonification de deux points pour deux cas de figure :

— les élèves ont 6 ou moins,

— les élèves ont 8 ou moins.

Activité 4 :

1. Recopier le code 5 dans un EDI et repérer l'erreur dans les calculs effectués.
2. Copier le code dans *pythontutor* et expliquer cette erreur.

Effet de bord

En première approche nous pouvons retenir qu'une variable mutable est modifiée quand elle est utilisée dans une fonction. Il est souvent difficile de maîtriser les modifications ce qui peut créer des comportements non désirés du programme.

Christophe Viroulaud

4.4 Bonnes pratiques

Il faut visualiser une fonction comme un outil autonome, réutilisable dans plusieurs programmes. Elle ne doit donc pas dépendre de variables globales.

À retenir

On évitera au maximum d'utiliser une variable globale.

Activité 5 : Modifier la fonction **simulation_bonification** pour qu'elle crée une copie du tableau de notes et effectue les modifications sur cette copie.

Christophe Viroulaud