Unité 1 : Débuter la programmation en Python | **Compétence 2 : Les types de données en Python**

Dans cette deuxième leçon de l'unité 1, vous allez découvrir comment utiliser le type des données en Python.

Objectifs :

- Connaitre les différents types de données en langage Python.
- Mettre en forme le format d'une donnée numérique

Connaitre le type des grandeurs utilisé.

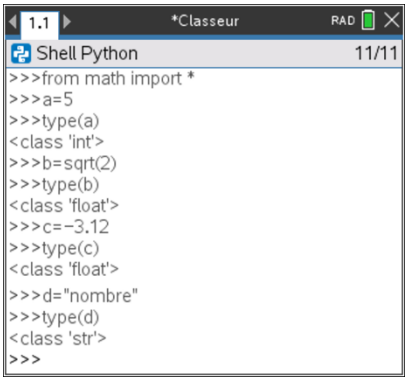
Lorsque vous utilisez un script en langage Python, il peut s'avérer nécessaire de connaitre le type de variable utilisé, ou bien de modifier ces variables en vue d'une utilisation ultérieure. Par exemple si la grandeur renvoyée par un script Python renvoie une grandeur correspondante à une mesure en sciences physiques, il n'est pas nécessairement opportun de conserver un résultat à 6 décimales.

Vous allez créer un script dans l'application **Python** permettant de distinguer les types des grandeurs utilisées.

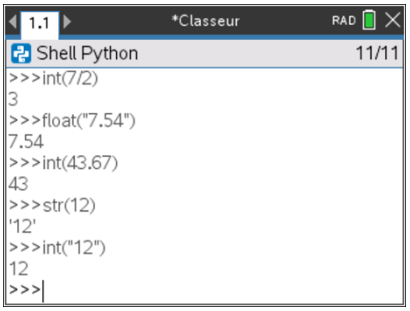
- Une chaîne de caractères.
- Un nombre réel.
- Un nombre irrationnel, $\sqrt{2}$ par exemple.

Vous afficherez le nombre ainsi que son type à l'aide de l'instruction **type**.

- Créer un nouveau document Python comportant uniquement la console.
- Importer le module **maths** (menu puis **4 Maths**) et enfin **1 from maths import***.
- L'ensemble des commandes utilisées peut être directement entré au clavier. La commande type qui permet de connaître la nature d'une variable est tapée à la main.



Astuce : L'utilisation des touches de direction ▲ puis **enter** permet de recopier une ligne lorsque l'utilisateur travaille dans la console.



Remarques :

- L'opérateur **int()** extrait lorsque c'est possible, un entier d'une chaîne de caractères et renvoie la partie entière d'un nombre comprise entre $\pm$ 2 147 483 648 (codage sur 32 bits, soit 4 octets)
- L'opérateur **str()** transforme un nombre en une chaîne de caractères.
- L'opérateur **float()** extrait lorsque c'est possible, un flottant d'une chaîne de caractères.

Une autre astuce :

Pour incrémenter une variable dans un compteur, on dispose de deux possibilités :

a) Écrire par exemple :

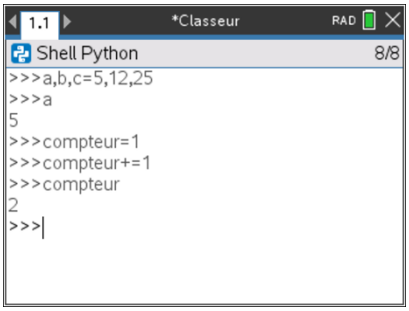
```
compteur = 0
compteur = compteur +1
```

demander l'affichage de la variable

```
compteur
```

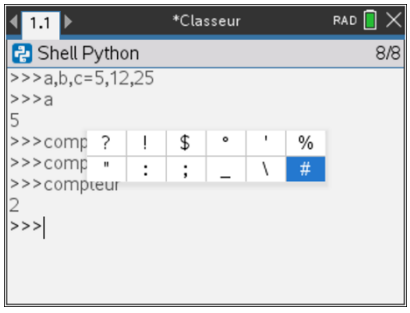
b) Ou bien

```
compteur = 0
compteur+=1
compteur
```



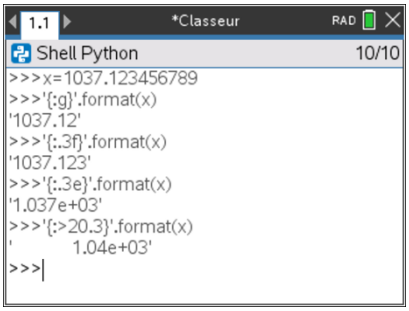
Conseils à l'enseignant : Utiliser les fonctions de « copier-coller » **Ctrl C** et **Ctrl V**
Lors de la rédaction, la touche del permet d'effacer un caractère entré par erreur.

Vous avez la possibilité de commenter vos scripts en ajoutant devant le commentaire, un (# commentaire). Le fait de mettre un # indique que la ligne ne sera pas interprétée. Pour l'obtenir, appuyer sur la touche ?!



Pour aller plus loin :

Le format des nombres : La méthode **format** de l'objet string est un outil très puissant permettant de créer des chaînes de caractères en remplaçant certains champs (entre accolades) par des valeurs (passées en argument de la fonction format) après conversion de celles-ci. On peut préciser à l'intérieur de chaque accolade un code de conversion, ainsi que le gabarit d'affichage. Donnons quelques exemples.




```
>>> x=1037.123456789
>>> '{ :g}'.format(x) # choisit le format le plus approprié '1.04e+03'
>>> '{ :.3f}'.format(x) # fixe le nombre de décimales
'1037.123'
>>> '{ :.3e}'.format(x) # notation scientifique
'1.037e+03'
>>> '{0 :20.3f}'.format(x) # précise la longueur de la chaîne
' 1037.123'
>>> '{0 :>20.3f}'.format(x) # justifié à droite
' 1037.123'
>>> '{0 :<20.3f}'.format(x) # justifié à gauche
'1037.123 '
>>> '{0 :^20.3f}'.format(x) # centré
' 1037.123 '
>>> '{0 :+.3f} ; {1 :+.3f}'.format(x, -x) # affiche toujours le signe '+1037.123 ; -
1037.123'
>>> '{0 : .3f} ; {1 : .3f}'.format(x, -x) # affiche un espace si x>0
```