Nom :	Prénom :	Classe :

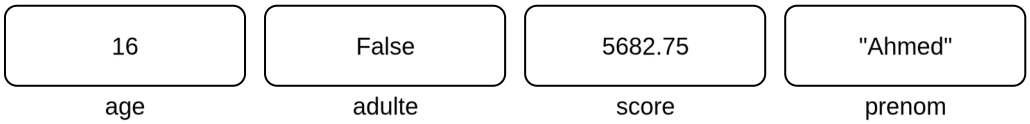
NSI 1re — Affectation, calculs et affichage (print)

Élément du programme : Langages et programmation → Constructions élémentaires. Séquences, affectation.

Qu'est-ce qu'une variable ?

Une **variable** est un peu comme une "**boîte de rangement étiquetée**", dans laquelle on stocke une **information**.

Prenons un exemple :



Dans l'illustration ci-dessus, nous avons créé (déclaré) 4 variables différentes :

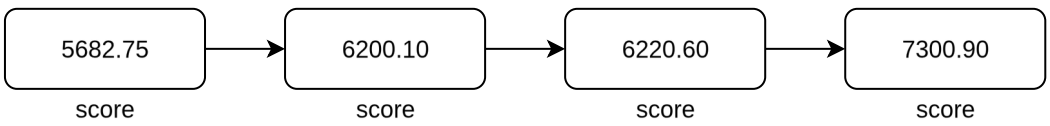
- Une variable `age` contenant la valeur `16`.
- Une variable `adulte` contenant la valeur `False`.
- Une variable `score` contenant la valeur `5682.75`.
- Une variable `prenom` contenant la valeur `"Ahmed"`.

L'idée des variables vient des mathématiques, où une variable représente une valeur qui peut changer selon les circonstances, comme “x” dans une équation. En informatique, on a reprit ce principe pour que les programmes puissent se souvenir de certaines données (par exemple, garder en mémoire le score d’un joueur ou le prénom d’un utilisateur).

La valeur contenue dans une variable peut changer

Les variables permettent à un programme de fonctionner de façon dynamique : il peut réagir et s’adapter selon ce qui se passe ou ce que l’utilisateur fait. Sans variables, il serait impossible de faire évoluer un jeu vidéo, de lancer une appli de messagerie, ou de demander un mot de passe : tout serait figé.

Avec les variables, les programmes gagnent en souplesse, car on peut stocker, modifier, puis réutiliser facilement des informations au fil de l’exécution.



Dans l'illustration ci-dessus, la valeur stockée dans la variable `score` a été modifiée quatre fois.

Qui utilise des variables ?

Les variables concernent toutes celles et ceux qui créent ou utilisent des programmes : développeuses et développeurs, créatrices et créateurs de jeux, étudiantes et étudiants qui apprennent à coder, mais aussi toute application ou logiciel, car même une appli météo utilise des variables pour retenir `temperature`, `date`, `ville`, etc. Les variables sont **indispensables** dans tous les langages de programmation, du plus simple (comme Scratch) au plus complexe.

À quoi ça sert, une variable ?

Une variable sert à **retenir une information** qui peut changer (score d'un jeu, âge d'un utilisateur, nombre de tentatives, etc.).

Elle peut aussi permettre au programme de **faire des calculs**, de prendre des décisions, ou de modifier son comportement selon les circonstances. Elle donne aux développeuses et développeurs la possibilité d'écrire des programmes qui réagissent à ce qui se passe ou à ce que l'utilisatrice ou l'utilisateur choisit.

Une variable, c'est un outil **fondamental** pour rendre les programmes interactifs et adaptables : sans elle, l'informatique moderne n'existerait tout simplement pas.

Comment créer une variable en Python ?

Pour créer une variable en Python, il suffit d'écrire le **nom de la variable**, puis le **signe égal** =, puis la **valeur** qu'elle doit contenir : par exemple : `age = 15` ou `pseudo = "Louise_xoxo"`.

Voici quelques exemples de créations (déclarations) de variables :

```
1  age = 16
2  adulte = False
3  score = 5682.75
4  prenom = "Ahmed"
```

Dans le programme ci-dessus, nous avons créé des variables qui stockent un nombre (ligne 1), un *booléen* (ligne 2, nous en reparlerons prochainement), un nombre à virgule (aussi appelé un « flottant », ligne 3), et un texte (aussi appelé une « chaîne de caractères », ligne 4).

➤ Exercice 1 (à vous de jouer)

Observez le code ci-dessous, puis répondez aux différentes questions posées.

Code	Vos réponses (à compléter)

`pays = "France"`
`capitale = "Paris"`
`superficie = 672051`
`densite = 107.2`
`europe = True`
	Combien de variables sont déclarées ? .....…
	Quel est le contenu de la variable `densite` ? .....
	Quel est le contenu de la variable `capitale` ? .....

Règles à respecter

Nous pouvons déclarer **autant de variables que désirées**.
Cependant, il existe quelques règles à respecter :

- La **première lettre** du nom de la variable doit être une **lettre de l'alphabet**.
  (ou un tiret du bas : `_` appelé « underscore »).
- Les **lettres suivantes** peuvent être des lettres de l'alphabet, des chiffres, ou des underscores.
  Tous les autres caractères sont interdits. Le caractère "espace" est donc interdit.
- Le nom de la variable est **sensible à la casse**.
  Par exemple `artist` et `Artist` sont deux variables différentes.

En Python, il est généralement *déconseillé* d'utiliser des noms de variables avec des accents. Car, malgré le fait que soit possible, cela peut entraîner des problèmes de lisibilité et de compatibilité, surtout si le code est partagé ou utilisé dans des environnements différents. Il est aussi *recommandé* d'utiliser des noms de variables en minuscules, avec des mots séparés par des underscores.

Notons que ces deux *indications* ne sont pas des *règles* mais des *préconisations*.

➤ Exercice 2

Observez les déclarations de variables qui suivent, et dites si le code est valide ou incorrect, en **justifiant vos réponses**.

Code	Valide ? (à compléter)	Pourquoi ? (à compléter)
`mario_pv = 100`		
`artist_1 = "Dr. Dre"`		
`2_city = "Compton"`		
`sport-1 = "bmx"`		
`_naissance = 1971`		
`CHiLdRen = 3`		
`_ = "salut"`		
`personnage = "Chun-Li"`		
`-age = 18`		

Effectuer des calculs avec des variables

Nous l'évoquions au début de ce document : l'intérêt des variables, c'est qu'elles nous permettent de garder une trace des choses, sans avoir à tout compter à chaque fois. *Les variables donnent aux développeuses et développeurs la possibilité d'écrire des programmes qui réagissent à ce qui se passe ou à ce que l'utilisatrice ou l'utilisateur choisit.*

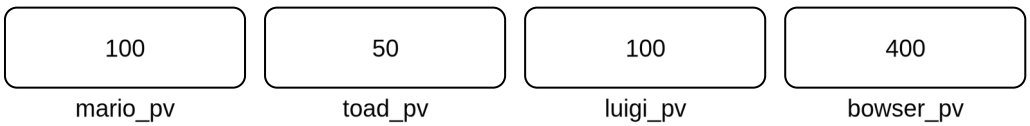
Prenons l'exemple d'un jeu vidéo bien célèbre, à travers cet extrait de code dans lequel nous définissons quatre variables (note : « pv » signifie « point de vie ») :

```
1  mario_pv = 100
2  toad_pv = 50
3  luigi_pv = 100
4  bowser_pv = 400
```

Dans le code ci-dessus, nous pouvons dire que nous venons **d'assigner** :

- La valeur `100` à la variable `mario_pv`.
- La valeur `50` à la variable `toad_pv`.
- La valeur `100` à la variable `luigi_pv`.
- La valeur `400` à la variable `bowser_pv`.

En reprenant notre illustration de « boîte de rangement étiquetée », cela donnerait :



Changer les valeurs contenues dans les variables

Les variables contiennent des informations qu'il est possible de **modifier** ; par exemple :

- Mario pourrait avoir perdu 10 points de vie (PV).
- Toad aurait perdu 30 PV.
- Et Bowser aurait gagné 50 PV.

Voici le code Python permettant de réaliser cette opération :

```
1  mario_pv = mario_pv - 10
2  toad_pv = toad_pv - 30
3  bowser_pv = bowser_pv + 50
```

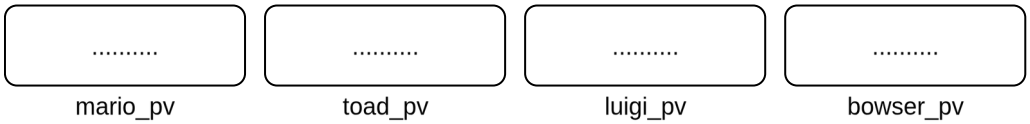
Et voici désormais une **explication du code** ci-dessus :

- Soustrait `10` à la valeur stockée dans `mario_pv`, ligne 1.
- Soustrait `30` à la valeur stockée dans `toad_pv`, ligne 2.
- Ajouté `50` à la valeur stockée dans `bowser_pv`, ligne 3.

➤ Exercice 3

Une fois le code précédent exécuté (voir le code Python en bas de la page précédente), quelles seraient désormais — selon vous — les valeurs stockées dans les différentes variables ?

Complétez les valeurs dans l'illustration ci-dessous :



Opérations arithmétiques

Dans les exemples précédents, nous avons utilisé les opérateurs permettant de réaliser :

- Une addition, avec le symbole `+`.
- Une soustraction, avec le symbole `-`.

Pouvons-nous utiliser tous les autres **opérateurs arithmétiques**[1] ?
La réponse est oui ! Voici un tableau qui résume ces opérateurs :

Opération	Symbole en Python	Commentaire
Addition	+	Symbole « plus »
Soustraction	-	Symbole « tiret du six »
Multiplication	*	Symbole « astérisque »
Division	/	Symbole « slash »
Résultat entier d'une division	//	Deux symboles « slash » à la suite
Reste d'une division	%	Symbole « pourcentage »
Puissance	**	Deux symboles « astérisque »

Voyons, page suivante, quelques exemples d'utilisation de ces opérateurs sur des variables ...

1 Revoir à ce propos le cours « Séquence d'exécution de code (calculs, commentaires) »

Exemple de la recette de crêpes

```
# recette v1

farine = 125 * 2
beurre = 7.5 * 2
oeuf = 1 * 2
sucre = 37.5 * 2
lait = 250 * 2
```

Dans cette « recette v1 », nous assignons 5 variables — dont les noms sont ceux des ingrédients — et nous leur donnons une valeur, qui est le **résultat** d'une opération mathématique : une multiplication.

➤ *Exercice 4*

Après avoir analysé le code de la « recette v1 », écrivez les valeurs stockées dans les différentes variables :

Variable	farine	beurre	oeuf	sucre	lait
Valeur (à compléter)	.....	.....	.....	.....	.....

Une recette adaptable

Imaginons maintenant que la développeuse ou le développeur nous dise qu'il a multiplié chaque ingrédient par `2`, car sa recette de crêpes est faite pour deux personnes.

Que faire si nous devons préparer cette recette pour 3 personnes ?
Puis pour 8 personnes ? Faudrait-il *à chaque fois* modifier chaque ligne de code ?

N'y aurait-il pas une autre possibilité ?

➤ *Exercice 5*

Écrivez votre suggestion permettant de ne pas avoir à réécrire l'intégralité de cette recette à chaque fois :

➤ *Exercice 6*

Voici le code « recette v1 », légèrement modifié (d'où son nom : « recette v2 ») :

```
# recette v2

amis = 2
farine = 125 * ........
beurre = 7.5 * ........
oeuf = 1 * ........
sucre = 37.5 * ........
lait = 250 * ........
```

À vous de jouer, **complétez le code ci-dessus**, de façon à trouver une utilisation de la nouvelle variable `amis`.

➤ *Exercice 7*

Imaginons que nous donnions la valeur de `8` à la variable `amis`.
Quelles seraient alors les valeurs stockées dans les différentes variables du programme ?

Variable	farine	beurre	oeuf	sucre	lait
Valeur	.....	.....	.....	.....	.....

Exemple des sushis à partager

Le restaurant japonais de votre quartier vous demande d'écrire le petit programme suivant :

- Assignez la valeur `14` à une variable `sushis`.
- Assignez la valeur `4` à une variable `amis`.
- Calculez le nombre de sushis entiers que chaque ami aura,
  en assignant ce résultat à une variable `sushis_par_ami`.
- Calculez le nombre de sushis restants,
  en assignant ce résultat à une variable `sushis_restants`.

➤ *Exercice 8*

En utilisant les **opérateurs arithmétiques appropriés**, écrivez votre proposition de programme :

Afficher la valeur d'une variable avec `print()`

La commande `print()` de Python permet d'afficher la valeur stockée dans une variable :

```
1  titre = "Arcane"
2  saisons = 2
3  episodes = 18
4
5  print(titre)
6  print(saisons)
7  print(episodes)
```

Le programme Python ci-dessus affichera :

```
Arcane
2
18
```

Surveiller la valeur stockée dans une variable

Puisque la commande `print()` permet **d'afficher la valeur** stockée dans une variable, rien ne nous empêche de l'utiliser **autant de fois que désiré**, notamment afin de « **surveiller** » des **changements de valeurs**. Exemple :

```
1  age = 23
2  print(age)
3  age = age + 1
4  print(age)
5  age = age // 2
6  print(age)
```

Le programme Python ci-dessus affichera :

```
23
24
12
```

Dans cet exemple :

- Nous assignons la valeur `23` à la variable `age` (ligne 1).
- Nous affichons la valeur de `age` à cet instant précis (ligne 2).
- Nous ajoutons `1` à la valeur de `age` (ligne 3) — on dit alors qu'on *incrémente* `age` de `1`.
- Nous affichons la valeur de `age` à ce *nouvel instant* précis (ligne 4).
- Puis nous stockons dans `age` le résultat entier d'une division de sa valeur par `2` (ligne 5).
- Puis nous affichons la valeur de `age` à ce nouvel instant précis (ligne 6).

➤ *Exercice 9*

Voici un nouveau programme :

```
1   a = 1
2   b = a
3   print(b)
4   a = a + 1
5   print(a)
6   print(b)
7   b = b + 4
8   print(b)
9   c = a + 2
10  print(c)
```

Analysez le code ci-dessus, puis déterminez quelle **valeur** est **affichée** par Python aux lignes indiquées :

Ligne	Valeur affichée par Python (à compléter)
3	
5	
6	
8	
10	

Vous avez terminé cet exercice ?
Alors vous pouvez prendre une *petite* pause de 5 minutes en passant voir vos camarades, afin de leur proposer votre aide *dans le calme*.

Vous poursuivrez sur votre copie juste après.

➤ *Exercice 10*

Déterminez ce que le programme suivant va afficher :

Code	Ce que Python va afficher (à compléter)

`perso_pv = 100`
`monstre_pv = perso_pv`
`perso_pv = perso_pv // 2`
`print(monstre_pv)`
`monstre_pv = monstre_pv - 10 * 2`
`print(perso_pv * 2)`
`print(perso_pv)`	

Pour aller plus loin

🌶️ Supplément pimenté – Voici un nouveau programme :

```
1   # Initialisation des pokémons
2   goupix = 100
3   onix = goupix
4   onix = onix - 10
5   roucool = 90
6   raichu = onix
7   bulbizarre = goupix
8   salameche = 80
9   print(goupix)
10  print(onix)
11  print(roucool)
12  print(raichu)
13  print(bulbizarre)
14  print(salameche)
15
16  # Le match commence !
17  goupix = goupix // 2
18  print(goupix)
19  salameche = salameche * 2
20  print(salameche)
21  raichu = raichu % 8
22  print(raichu)
23  bulbizarre = bulbizarre - raichu
24  print(bulbizarre)
25  raichu = raichu ** 10
26  print(raichu)
27  roucool = roucool - raichu
28  print(roucool)
```

Analysez le code ci-dessus, puis déterminez quelle valeur est affichée par Python aux lignes indiquées :

Ligne	Valeur affichée par Python (à compléter)	Ligne	Valeur affichée par Python (à compléter)
9		18	
10		20	
11		22	
12		24	
13		26	
14		28	