Python : Variables

1ère NSI

Plan du cours

Variable informatique

Une **variable** informatique est une **donnée stockée dans la mémoire de l'ordinateur**, caractérisée par :

- son **adresse mémoire** (automatique en Python !)
- son **nom**
- sa **valeur**

Variable informatique

Une **variable** informatique est une **donnée stockée dans la mémoire de l'ordinateur**, caractérisée par :

- son **adresse mémoire** (automatique en Python !)
- son **nom**
- sa **valeur**

La **création** et la **modification** d'une variable se fait toujours de la même manière : `nom = valeur`

Variable informatique

Une **variable** informatique est une **donnée stockée dans la mémoire de l'ordinateur**, caractérisée par :

- son **adresse mémoire** (automatique en Python !)
- son **nom**
- sa **valeur**

La **création** et la **modification** d'une variable se fait toujours de la même manière : `nom = valeur`

Exemples :

```
a = 3
```

```
pi = 3.1415
```

```
texte = "Salut"
```

Variable informatique

Le **nom d'une variable** joue le rôle d'**identifiant** dans la mémoire de l'ordinateur.

La **valeur**, quant à elle, peut être amenée à changer.

```
a = 3       # La valeur de a est initialisée à 3
a = 7       # La valeur de a est modifiée
a = a + 4   # Maintenant, la valeur de a est 7 + 4 = 11
```

Bien nommer ses variables

Le nom d'une variable doit respecter les règles suivantes :

- Il ne peut être composé que de **lettres**, majuscules ou minuscules, de **chiffres** et du symbole souligné _
- Il ne doit **pas commencer par un chiffre**
- Il doit être différent des mots suivants, qui sont des mots-clés / instructions utilisées par Python :

```
1  and as assert break class continue def del elif else
2  except False finally for from in is lambda none nonlocal
3  not or pass raise return True try while with yield
```

De plus, Python est **sensible à la casse**, ce qui signifie que l'on distingue les lettres minuscules et majuscules. Ainsi, deux variables nommées `variable` et `Variable` seront différentes.

Variables numériques : `int` et `float`

Python distingue 2 types de variables numériques :

- les **nombres entiers**, de type `int` pour *integer* (entier en anglais)
- les **nombres décimaux**, de type `float`, aussi appelés *flottants* (nombres à virgule)

En Python, **on ne choisit pas le type de variable**. Il est automatiquement déterminé par la valeur donnée à la variable.

```
a = 3       # De type int
b = 3.14    # De type float
```

Variables numériques : `int` et `float`

La taille d'un nombre entier (`int`) n'est limitée que par la mémoire de l'ordinateur. Python gère donc sans soucis les (très) grands nombres entiers.

```
>>> 2**1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542167660429831652624386837205668069376
```

Variables numériques : `int` et `float`

Le nombre maximum de décimales d'un `float` est de 16.

Contrairement aux nombres entiers, il existe des « limites »pour les valeurs, dûes à la façon de représenter les nombres décimaux en binaire.
Pour les nombres positifs :

- minimum : $2{,}2250738585072014 \times 10^{-308}$
- maximum : $1{,}7976931348623157 \times 10^{308}$

Le calcul avec les flottants produit parfois des résultats étranges (mais explicables) :

```
>>> 0.1 + 0.1 + 0.1
0.30000000000000004
```

Chaînes de caractères : `str`

Une **chaîne de caractères** est le type de variable associé à du **texte**. En anglais, on parle de *string* (pour *chaîne*), abrégé `str` en Python.

Pour définir une chaîne de caractères, on écrit la valeur entre guillemets :

```
phrase = "Voici une chaîne de caractères"
```

Chaînes de caractères : `str`

Il existe deux opérations sur les chaînes de caractères :

Chaînes de caractères : `str`

Il existe deux opérations sur les chaînes de caractères :

- la **concaténation**, qui correspond à la mise bout à bout de deux chaînes de caractères

```
1 >>> "Bonjour" + "Alice"
2 'BonjourAlice'
```

Chaînes de caractères : `str`

Il existe deux opérations sur les chaînes de caractères :

- la **concaténation**, qui correspond à la mise bout à bout de deux chaînes de caractères

```
>>> "Bonjour" + "Alice"
'BonjourAlice'
```

- la multiplication par un entier n, qui concatène n fois la chaîne donnée

```
>>> "to" * 5
'tototototo'
```

Mais encore : `bool`, `list`, `tuple`...

Les types `int`, `float` et `str` ne sont pas les seuls à exister. Il en existe un grand nombre que nous découvrirons cette année :

Mais encore : `bool`, `list`, `tuple`...

Les types `int`, `float` et `str` ne sont pas les seuls à exister. Il en existe un grand nombre que nous découvrirons cette année :

- les **booléens** (`bool`), qui sont des variables binaires (Vrai ou Faux)

Mais encore : `bool`, `list`, `tuple`...

Les types `int`, `float` et `str` ne sont pas les seuls à exister. Il en existe un grand nombre que nous découvrirons cette année :

- les **booléens** (`bool`), qui sont des variables binaires (Vrai ou Faux)
- les **listes** (`list`), sortes de *catalogues* de plusieurs variables

Mais encore : `bool`, `list`, `tuple`...

Les types `int`, `float` et `str` ne sont pas les seuls à exister. Il en existe un grand nombre que nous découvrirons cette année :

- les **booléens** (`bool`), qui sont des variables binaires (Vrai ou Faux)
- les **listes** (`list`), sortes de *catalogues* de plusieurs variables
- les **tuples** (`tuple`), équivalent mathématiques d'un couple, d'un triplet...

Mais encore : `bool`, `list`, `tuple`...

Les types `int`, `float` et `str` ne sont pas les seuls à exister. Il en existe un grand nombre que nous découvrirons cette année :

- les **booléens** (`bool`), qui sont des variables binaires (Vrai ou Faux)
- les **listes** (`list`), sortes de *catalogues* de plusieurs variables
- les **tuples** (`tuple`), équivalent mathématiques d'un couple, d'un triplet...
- ... et plein d'autres !

Astuces pratiques

Pour terminer, quelques astuces bien pratiques :

Astuces pratiques

Pour terminer, quelques astuces bien pratiques :

- Double affectation, pour donner des valeurs à plusieurs variables simultanément :

```
1  a,b = 3,8   # a vaut 3 et b vaut 8
```

Astuces pratiques

Pour terminer, quelques astuces bien pratiques :

- Double affectation, pour donner des valeurs à plusieurs variables simultanément :

```
1  a,b = 3,8   # a vaut 3 et b vaut 8
```

- Opération *à la volée* :

```
1  a += 4      # Ajoute 4 à la variable a
2  b *= 2      # Multiplie la variable b par 2
```