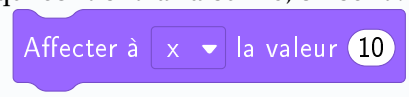
AP01 CONSTRUCTIONS ÉLÉMENTAIRES

I. Les variables en Python

Déclarer une variable en Python

Pour déclarer une variable en Python, il suffit de lui donner un nom et de lui affecter une valeur à l'aide du symbole =. Par exemple, pour déclarer une variable `x` qui contient la valeur 10, on écrit :

</> Code Python
```
a = 10
```



Le nom des variables en Python

- Elles doivent commencer par une lettre ou un underscore (_).
- Elles peuvent contenir des lettres, des chiffres et des underscores.
- Elles ne peuvent pas être des mots réservés du langage Python (comme `if`, `for`, `while`, etc.).
- `Variable` et `variable` sont considérées comme deux variables différentes.

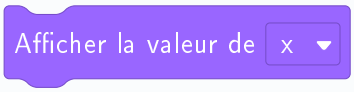
Exemple : Noms de variables valides et invalides

Bonne variable	`ma_variable`
Mauvaise variable	`Ma_variable`, `ma variable`, `1_variable`, `variable@`
A éviter	`Variable non explicites`

Afficher une variable en Python

Pour afficher la valeur d'une variable en Python, on utilise la fonction `print()`. Par exemple, pour afficher la valeur de la variable `x`, on écrit :

</> Code Python
```
print(x)
```



Les types de variables en Python

- **int** : pour les nombres entiers (ex : 1, 42, -5)
- **float** : pour les nombres à virgule flottante (ex : 3.14, -0.001)
- **str** : pour les chaînes de caractères (ex : "Bonjour", 'Python')
- **bool** : pour les valeurs booléennes (ex : True, False)

```
# type (int)
a = 5
b = 6
```

```
# type (str)
a = 'Bonjour'
b = '6'
```

```
# type (float)
a = 5.0
b = 6.8
```

```
# type (bool)
a = True
b = False
```

Opérations mathématiques de base en Python

- `+` addition : `5 + 3` donne 8
- `-` soustraction `5 - 3` donne 2
- `*` multiplication `5 * 3` donne 15
- `/` division `6 / 2` donne 3.0 (`float`)
- `**` puissance `2 ** 3` donne 8
- `%` reste de la division entière `7 % 3` donne 1
- `//` quotient de la division entière `7 // 3` donne 2

```
>>> print(17+2)
19
>>> print(17-2)
15
```

```
>>> print(17*2)
34
>>> print(17/2)
8.5
```

```
>>> print(17%2)
1
>>> print(17//2)
8
```

Exercice 1 : Repérer les les types de variables et leurs valeurs dans les programmes suivants

- `x = 5`
- `y = "Hello"`
- `z = True`
- `a = 3.14`
- `c = 6/2`
- `d = 6//2`
- `f = '1' + '2'`
- `g = 2**3`

II. Les variables de type chaine de caractère `str`

Les chaines de caractères

Les chaines de caractères sont des variables qui contiennent du texte. Elles sont délimitées par des guillemets simples ou doubles.

```
>>> a = 'Bonjour'
>>> b = ' a tous'
>>> c = a + b
>>> print(c)
Bonjour a tous
```

```
>>> a = 'Bonjour'
>>> b = ' a tous'
>>> c = a * 3
>>> print(c)
BonjourBonjourBonjour
```

```
>>> a = 'Bonjour'
>>> b = ' a tous'
>>> c = a[0]
>>> print(c)
B
```

Dans l'exemple ci-dessus, la variable `c` contient la concaténation de `a` et `b` dans le premier cas, la répétition de `a` trois fois dans le deuxième cas et le premier caractère de `a` dans le troisième cas.

indexation des chaines de caractères

Les chaines de caractères peuvent être vues comme un ensemble de caractères "collés" les uns aux autres Chaque caractère d'une chaine de caractères est indexé : il y a un index qui donne sa position dans la chaine

Indice	0	1	2	3	4	5	6
Caractère	B	o	n	j	o	u	r
Indice Négatif	-7	-6	-5	-4	-3	-2	-1

```
>>> mot = 'Bonjour'
>>> mot[0]
'B'
>>> mot[4]
'o'
```

```
>>> mot = 'Bonjour'
>>> n = len(mot)
>>> n
7
>>> mot[-1]
'r'
```

Les opérations sur les chaines de caractères

`+` pour la concaténation : `'Bonjour' + ' a tous'` donne `'Bonjour a tous'`

`*` pour la répétition : `'Bonjour' * 3` donne `'BonjourBonjourBonjour'`

`[]` pour l'indexation : `'Bonjour'[0]` donne `'B'`

`in` pour le test d'appartenance : `'on' in 'Bonjour'` donne `True`

`not in` pour le test d'appartenance : `'on' not in 'Bonjour'` donne `False`

`len()` pour la longueur : `len('Bonjour')` donne 7

```
>>> mot1 = 'Bonjour'
>>> mot2 = 'Thomas'
>>> mot1 + ' ' + mot2
'Bonjour Thomas'
```

```
>>> 'j' in 'Bonjour'
True
>>> 'b' in 'Bonjour'
False
```

```
>>> 'abracadabra'[4]
'c'
>>> len('abracadabra')
11
```

Opérations avancées sur les chaines de caractères : le Slicing

`[a:b]`	conserve les caractères d'indice `[a:b]` : `'Bonjour'[1:4]` donne `'onj'`
`[a:]`	conserve les caractères d'indice `[a:]` : `'Bonjour'[3:]` donne `'jour'`
`[:b]`	conserve les caractères d'indice `[:b]` : `'Bonjour'[:3]` donne `'Bon'`
`[::-1]`	inverse les caractères : `'Bonjour'[::-1]` donne `'ruojnoB'`
`[1:-1]`	élimine le premier et dernier caractère : `'Bonjour'[1:-1]` donne `'onjou'`

Exercice 2 : Prédire ce que vont afficher les codes suivants

```
>>> mot = 'Numerique'
>>> n = len(mot)
>>> mot[5]
>>> mot[n]
>>> mot[-1]
>>> mot[8]
```

```
>>> mot = 'Numerique'
>>> m in mot
>>> e not in mot
>>> mot+mot[2]
>>> mot*2
>>> 'abc'+mot
```

```
>>> mot = 'Numerique'
>>> mot[1:5]
>>> mot[:4]
>>> mot[3:]
>>> mot[1:-1]
```

Principe du transtypage

Les types des variables en python sont dynamiques, on peut modifier sa valeur ainsi que son type.

```
>>> a = '5'      # a est de type str
>>> b = int(a)   # b est de type int
>>> print(b)
5
```

```
>>> a = 4        # a est de type int
>>> b = str(a)   # b est de type str
>>> print(b)
4
```

III. Les variables de type booléen `bool`

Les booléens

Les booléens sont des variables qui ne peuvent prendre que deux valeurs : `True` (vrai) ou `False` (faux).

Les booléens peuvent être utilisés dans des opérations logiques.

```
>>> a = True
>>> b = False
>>> c = a and b
>>> c
False
```

```
>>> a = True
>>> b = False
>>> c = a or b
>>> c
True
```

```
>>> a = True
>>> c = not a
>>> c
False
```

Table de vérité du `not`

Expression	Résultat
`not True`	False
`not False`	True

Table de vérité du `and`

Expression	Résultat
True and True	True
True and False	False
False and True	False
False and False	False

Table de vérité du `or`

Expression	Résultat
True or True	True
True or False	True
False or True	True
False or False	False

Booléen comme résultat de comparaison

Les booléens sont le résultat d'une comparaison. Les opérateurs de comparaison sont :

- `==` pour le test d'égalité
- `!=` pour le test de différence
- `<` pour le test d'infériorité
- `>` pour le test de supériorité
- `<=` pour le test d'infériorité ou d'égalité
- `>=` pour le test de supériorité ou d'égalité

Booléen comme résultat de comparaison

```
>>> a = 5
>>> b = 6
>>> c = (a == b)
>>> c
False
```

```
>>> a = 5
>>> b = 6
>>> c = (a != b)
>>> c
True
```

```
>>> a = 5
>>> b = 6
>>> c = (a < b)
>>> c
True
```

Exercice 3

Quelle est la valeur de la variable `c` après l'exécution des codes suivants ?

```
>>> a = 3
>>> b = 1
>>> c = a > b
```

```
>>> a = True
>>> b = False
>>> c = (a or b) and (not a)
```

```
>>> a, b = True, False
>>> c = (a and b) or (not a)
```

```
>>> a = True
>>> b = False
>>> x, y = 7, 9
>>> c = (b or (x == y)) or (a and (x != y))
```