# Opérateurs d'affectation, formatage des chaînes

## Doc. 2,2

**Objectif**

Manipuler des variables et des chaînes de caractères dans la console Python pour découvrir comment nommer et mettre à jour une information, puis comment l'insérer proprement dans un message.

## §1. Comment nommer une information à l'aide d'une variable ?

**Q1.** Tester dans la console les instructions suivantes :

```
>>> age = 15
>>> age
```
Python

Puis tester chacune des instructions ci-dessous, une par une :

```
>>> 1age = 15
>>> mon-age = 15
>>> for = 15
```
Python

Laquelle de ces trois dernières instructions fonctionne ? Pour les autres, à quel endroit Python signale-t-il une erreur ?

**Q2.** Tester dans la console :

```
>>> help("keywords")
```
Python

À quoi correspond la liste affichée ? Pourquoi est-il plus utile de retenir l'appel de la fonction `help` : `help("keywords")`, que la liste elle-même ?

**Q3.** Tester dans la console :

```
>>> ma_variable = 1
>>> maVariable = 2
>>> ma_variable
>>> maVariable
```
Python

Ces deux identificateurs désignent-ils la même variable ?

**À retenir**

Une **variable** est repérée par un **identificateur**, qui doit respecter certaines règles :

- il commence par une lettre ou un blanc souligné (_) ;
- il ne peut ensuite contenir que des lettres, des chiffres ou des blancs soulignés ;
- Python distingue les majuscules des minuscules (`age` et `Age` sont deux identificateurs différents) ;
- un **mot réservé** du langage (liste consultable avec `help("keywords")`) ne peut pas être choisi comme identificateur.

## §2. Que se passe-t-il lors d'une affectation ?

**Q4.** Sans utiliser Python, prévoir le résultat de l'évaluation de `a` et de `b` après ces instructions, puis tester dans la console :

```Python
>>> a = 3
>>> b = a
>>> a = 5
>>> a
>>> b
```

La variable `b` a-t-elle changé de valeur lorsque `a` a été modifiée ?

**Q5.** Prévoir le résultat de l'évaluation de `x`, `y` et `z` après ces instructions, puis tester dans la console :

```Python
>>> x = y = z = 1
>>> x
>>> y
>>> z
```

Dans quel sens faut-il lire la première instruction pour comprendre ce qu'elle fait : de la gauche vers la droite, ou de la droite vers la gauche ?

**Q6.** Prévoir le résultat de l'évaluation de `a`, `b` et `c` après cette instruction, puis tester dans la console :

```Python
>>> a, b, c = 1, 2, 3
>>> a
>>> b
>>> c
```

Combien de valeurs sont affectées en une seule instruction ?

**À retenir**

Une **affectation** (opérateur `=`) donne une valeur à une variable ; elle se lit toujours de la **droite vers la gauche**.

- **Affectation simple** : `a = 3`.
- **Affectation multiple** : `x = y = z = 1` affecte la même valeur à plusieurs variables.
- **Affectation groupée** : `a, b, c = 1, 2, 3` affecte plusieurs valeurs différentes en une instruction.

Python est un langage à **typage dynamique** : il n'est pas nécessaire de préciser le type d'une variable avant de lui affecter une valeur.

## §3. Comment mettre à jour la valeur d'une variable ?

**Q7.** Un article coûte 20 € hors taxe ; la TVA est de 18,6 %. Prévoir le résultat de l'évaluation de `prix` après ces instructions, puis tester dans la console :

```Python
>>> prix = 20
>>> tva = 18.6
>>> prix = prix + prix * tva / 100
>>> prix
```

Que représente l'expression à droite du signe = dans la troisième instruction ?

**Q8.** On suit maintenant le score d'un jeu. Prévoir le résultat de l'évaluation de `score` pour chacune des deux séquences suivantes, puis tester dans la console :

```
>>> score = 0
>>> score = score + 10
>>> score
```
Python

```
>>> score = 0
>>> score += 10
>>> score
```
Python

Que remarquez-vous ?

**Q9.** Par analogie avec +=, prévoir le résultat de l'évaluation de x après chacune des instructions suivantes, puis tester dans la console :

```
>>> x = 10
>>> x -= 3
>>> x
>>> x *= 2
>>> x
>>> x //= 5
>>> x
>>> x %= 2
>>> x
```
Python

Pour chaque opérateur d'affectation utilisé, réécrire l'instruction équivalente utilisant = et l'opérateur arithmétique correspondant.

**À retenir**

Un **opérateur d'affectation** met à jour la valeur d'une variable à partir de sa propre valeur, en une seule écriture (à lire de la droite vers la gauche) :

| Opérateur | Équivalent |
|---|---|
| `+=` | `variable = variable + expression` |
| `-=` | `variable = variable - expression` |
| `*=` | `variable = variable * expression` |
| `/=` | `variable = variable / expression` |
| `//=` | `variable = variable // expression` |
| `%=` | `variable = variable % expression` |

## §3.1. Peut-on échanger les valeurs de deux variables ?

**Q10.** Dans la partie principale de l'éditeur Thonny, écrire :

```
a = 605
b = 506
```
Python

```
a = b
b = a
print(a, b)
```

Lancer le programme en mode **débogage** puis utiliser la touche F7.

Ce programme permute-t-il bien les valeurs de `a` et `b` ? Compléter le tableau d'état ci-dessous.

| Instruction | a | b |
|---|---|---|
| `a = 605` | | |
| `b = 506` | | |
| `a = b` | | |
| `b = a` | | |

**Q11.** **Essentiel pour l'épreuve écrite.** Sans utiliser Python, compléter le tableau d'état ci-dessous en suivant, ligne par ligne, l'exécution de la séquence d'instructions suivante (la colonne `c` reste vide tant que cette variable n'existe pas encore). Vérifier ensuite en testant la séquence dans la console.

```
a = 7
b = 4
b = a + b
a = b - a
c = a * b
a += c
b -= 1
```
Python

| Instruction | a | b | c |
|---|---|---|---|
| `a = 7` | | | |
| `b = 4` | | | |
| `b = a + b` | | | |
| `a = b - a` | | | |
| `c = a * b` | | | |
| `a += c` | | | |
| `b -= 1` | | | |

**Q12.** Corriger le programme précédent, dans la partie principale de l'éditeur, en utilisant **une** variable supplémentaire nommée `tmp`, de façon à obtenir `a = 506` et `b = 605` après exécution.

**Q13.** **Plus difficile.** En utilisant uniquement les opérateurs `+=` et `-=`, permuter dans la partie principale de l'éditeur les valeurs de `a` et `b` **sans utiliser aucune variable supplémentaire**. Lancer le programme en mode débogage (F7) pour vérifier votre proposition en complétant le tableau d'état ci-dessous, avec `a = 605` et `b = 506`.

## §4. Comment insérer une valeur dans une chaîne de caractères ?

**Q14.** Tester dans la console :

```
>>> type("abcd")
>>> type('abcd')
```
Python

```
>>> type("Python, c'est top !")
>>> type('Python, c'est top !')
```

L'une de ces quatre instructions provoque une erreur. Laquelle, et pourquoi ?

**Q15.** Prévoir le résultat de l'évaluation de chacune des expressions suivantes, puis tester dans la console :

```
>>> prenom = "Alice"
>>> "Bonjour " + prenom
>>> age = 15
>>> "Tu as " + age + " ans"
```

L'une de ces deux dernières instructions échoue. Laquelle, et que dit le message d'erreur ?

**Q16.** Python propose une écriture appelée **f-string** pour éviter ce problème. Prévoir le résultat de l'évaluation de chacune des expressions suivantes, puis tester dans la console :

```
>>> prenom = "Alice"
>>> age = 15
>>> f"Bonjour {prenom}, tu as {age} ans."
>>> f"L'année prochaine, tu auras {age + 1} ans."
```

Que remarquez-vous sur le contenu des accolades `{}` dans la seconde instruction ?

**f-string**

Une **f-string** s'écrit `f"..."` : la lettre `f` juste avant les guillemets indique à Python de rechercher des accolades `{}` dans le texte. Chaque expression entre accolades est **évaluée**, convertie en `str`, puis insérée à cet endroit.

**Q17.** On souhaite afficher un prix avec exactement deux chiffres après la virgule. Prévoir le résultat de l'évaluation de chacune des expressions suivantes, puis tester dans la console :

```
>>> montant = 3333.3333
>>> f"{montant:.0f} €"
>>> f"{montant:.1f} €"
>>> f"{montant:.2f} €"
```

À quoi correspond le nombre placé juste avant la lettre `f`, après les deux points `:` ?

**Q18.** La méthode `.format()`, plus ancienne, permet d'obtenir le même résultat et se rencontre encore dans d'anciens programmes. Prévoir le résultat de l'évaluation de chacune des expressions suivantes, puis tester dans la console :

```
>>> prenom = "Alice"
>>> age = 15
>>> "Bonjour {}, tu as {} ans.".format(prenom, age)
>>> f"Bonjour {prenom}, tu as {age} ans."
```

Ces deux instructions produisent-elles le même résultat ?

**À retenir**

Pour insérer une valeur dans un texte, on préfère une **f-string** :

`f"...{expression}...{expression:.Nf}..."`

— `{expression}` insère le résultat de l'évaluation de l'expression, converti en `str` ;
— `{expression:.Nf}` affiche un nombre flottant arrondi à `N` chiffres après la virgule ;
— la méthode `"...{}...".format(...)`, plus ancienne, produit le même résultat mais reste moins lisible.

## §4.1. Exercices de synthèse

**⚠ Consigne**

Écrire chaque fonction dans la partie principale de l'éditeur, en respectant sa spécification et en utilisant une f-string pour construire le message retourné. Tester ensuite son appel dans la console.

**Q19.** Écrire la fonction `conversion_franc_euro` dont la spécification est :

```python
def conversion_franc_euro(francs: float) -> str:
    """Conversion d'une somme en francs en euros.
    Le message retourné donne les deux valeurs.

    >>> conversion_franc_euro(100)
    '100 francs correspondent à 15.24 euros.'
    """
```

Le franc était la monnaie utilisée en France avant l'euro (jusqu'en 2002) ; 1 euro correspond à 6,55957 francs. Le montant en euros sera affiché avec deux chiffres après la virgule.

**Q20.** Écrire la fonction `salutation` dont la spécification est :

```python
def salutation(nom: str, prenom: str, naissance: int) -> str:
    """À partir des nom, prénom et date de naissance, retourne un
    message d'accueil indiquant l'âge de la personne en 2026.

    >>> salutation("Dupond", "Patrick", 1961)
    'Bonjour Patrick Dupond, vous avez 65 ans.'
    """
```

**Q21.** Écrire la fonction `age_aleatoire` dont la spécification est :

```python
def age_aleatoire(age_min: int, age_max: int) -> str:
    """À partir de deux ages limites, retourne un message contenant
    un age déterminé aléatoirement entre age_min et age_max inclus.

    >>> age_aleatoire(20, 100)
    'Votre age est : 85 ans !'
    """
```

**Remarque :** utiliser la fonction `randint` du module `random` pour déterminer aléatoirement un nombre.

**Q22.** Écrire la fonction `ticket` qui réutilise le principe de la section précédente (prix hors taxe et TVA), et dont la spécification est :

```python
def ticket(article: str, prix_ht: float, tva: float) -> str:
    """Retourne une ligne de ticket de caisse donnant le nom de
    l'article et son prix TTC, arrondi à deux chiffres après la
    virgule.

    >>> ticket("Cahier", 2.5, 5.5)
    'Cahier : 2.64 €'
    """
```

## §4.2. Exercices de synthèse (Doc. 2,1 et Doc. 2,2)

**Consigne :** réinvestir les fonctions du Doc. 2,1 (spécification, module `math`) avec les notions du Doc. 2,2 (opérateur d'affectation, f-string). Écrire chaque fonction dans la partie principale de l'éditeur, puis tester son appel dans la console.

**Q23.** Écrire la fonction `prix_ttc` dont la spécification est :

```python
def prix_ttc(prix_ht: float, tva: float) -> float:
    """Retourne le prix TTC d'un article, à partir de son prix HT
    et du taux de TVA en pourcentage.

    >>> prix_ttc(20, 18.6)
    23.72
    """
```

Dans la partie principale de l'éditeur, à la suite de cette fonction, calculer le montant total TTC d'un panier contenant trois articles de prix HT 12 €, 8 € et 25 € (TVA à 20 % pour chacun), en partant de `total = 0` et en ajoutant chaque article avec l'opérateur `+=`. Afficher enfin, avec une f-string affichée par `print`, un message de la forme `Total du panier : 54.00 €`.

**Q24.** On reprend la fonction `distance` du Doc. 2,1 (à recopier dans la partie principale de l'éditeur si nécessaire) :

```python
def distance(x1: float, y1: float, x2: float, y2: float) -> float:
    """Retourne la distance entre deux points du plan."""
    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
```

Écrire, à la suite, la fonction `message_distance` dont la spécification est :

```python
def message_distance(x1: float, y1: float, x2: float, y2: float) -> str:
    """Retourne un message donnant la distance entre les points
    (x1, y1) et (x2, y2), arrondie à deux chiffres après la virgule.

    >>> message_distance(0, 0, 3, 4)
    'La distance entre les deux points est de 5.00 unités.'
    """
```

La fonction `message_distance` doit utiliser la fonction `distance`.

**Bilan : où en suis-je ?**

Pour chaque ligne, cochez la colonne qui correspond. Toute case **Pas encore** indique une partie à reprendre.

| Je sais… | Oui | Pas encore | À revoir |
|---|---|---|---|
| choisir un **identificateur** valide pour une variable | | | Partie 1 |
| prévoir l'effet d'une affectation simple, multiple ou groupée | | | Partie 2 |
| utiliser un **opérateur d'affectation** (+=, -=, ...) pour mettre à jour une variable | | | Partie 3 |
| échanger les valeurs de deux variables à l'aide d'une variable supplémentaire | | | Partie 3 |
| insérer une valeur ou une expression dans un texte avec une **f-string** | | | Partie 4 |
| afficher un nombre flottant avec un nombre choisi de chiffres après la virgule | | | Partie 4 |