Le 22/09/2023 | TP 1 — Python | CPES

Prénom : Nom :

1 Types de variables

Une variable peut être vue comme une boîte avec un nom dans laquelle on place une information (sa *valeur*). Le symbole = permet d'affecter une valeur dans une variable.
le code `a=8` permet d'affecter la valeur 8 dans la variable `a`.

1.1 Types simples

Les types de base sont les types numériques : `int`, `bool`, `float` et le type `str`.

- `int` permet de représenter les nombres entiers.
- `float` permet de représenter les nombres réels.
- `bool` permet de représenter une valeur booléenne : `True`(Vrai) et `False`(Faux). On peut aussi considérer ces valeurs comme des entiers 1 ou 0 (1 pour `True`).
- `str` permet de stocker des chaînes de caractères, éventuellements séparés par des espaces.

Python sait « deviner » le type de variable lors de l'affectation.

■ **Exemple 1:**

`a = 3`	a est de type `int`
`a = 3.`	a est de type `float`
`a = "3"`	a est de type `str`
`a = (3==3)`	a est de type `bool`

Sur les variables de type "nombres", on peut effectuer des **opérations** :

- Addition : `a+b`, Soustraction : `a-b`, Multiplication : `a*b`
- Puissance : `a**b` pour a^b
- Division décimale : `a/b`, quotient division euclidienne : `a//b`, reste division euclidienne `a%b`.

▶ Exercice 1

Dans l'interpréteur Python, effectuer les opérations suivantes et écrire en face de chaque ligne le résultat renvoyé :

```
>>> a = 15
>>> b = 4
>>> type(a)
>>> type(b)
>>> a + b
>>> a * b
>>> type(_)
>>> a**2
>>> a / b
>>> type(_)
>>> a // b
>>> a % b
```

le symbole "_ " permet d'accéder au dernier résultat renvoyé, c'est l'équivalent de la touche `ans` *de la calculatrice*

Les comparaisons : =, ≠, <, >, ≤, ≥ s'écrivent respectivement en Python : ==, !=, <, >, <= et >= et renvoient un booléen.

■ **Exemple 2:**
`a==b` renvoie `True` si a et égal à b et `False` sinon.

> ► **Exercice 2**
> Toujours dans l'interpréteur, écrire les commandes suivantes et écrire en face les résultats obtenus :
> ```
> >>> a==15
> >>> a<b
> ```

Le type `str` permet de représenter les chaînes de caractères.

■ **Exemple 3:**
`ch="Hello World"` affecte la chaîne de caractères dans la variable `ch`. ⚠Ne pas oublier les guillemets.

On peut également faire des opérations avec le type `str` :

- La concaténation : `ch1+ch2` juxtapose les deux chaînes de caractères
- La répétition : `ch*3` va juxtaposer 3 fois la chaîne contenue dans `ch`.

> ► **Exercice 3**
> Dans l'interpréteur, taper les commandes suivantes et écrire en face les résultats renvoyés :
> ```
> >>> txt = "garde"
> >>> txt2 = "fou"
> >>> txt + txt2
> >>> txt2*2
> ```

1.2 Types composés

On peut citer les *listes*, les *tuples* et les *dictionnaires*.

■ **Exemple 4:**

```
>>> liste = [1,4,"Pierre", 10.2]
>>> tuple = (7,5,8, "Jeannot")
>>> dict = {"nom" : "Dupont", "prenom" : "Pierre", "taille" : 1.78}
```

On reparlera plus tard de ces types de variables.

2 Entrées - Sorties

2.1 Les entrées

On peut demander à l'utilisateur du programme de rentrer des valeurs. On utilise pour cela la fonction `input` qui renvoie une valeur de type `str`.

■ **Exemple 5:**

```
>>>nom = input("Rentrer votre nom")
```

Ce programme affiche le texte `Rentrer votre nom` dans la console, et attend que l'utilisateur tape du texte. Le texte validé est stocké dans la variable `nom`.

⚠Si on veut que l'entrée soit un autre type qu'une chaîne de caractères, il faut « forcer » le typage comme ceci :

```
>>>age = int(input("Donner votre âge"))
>>>taille = float(input("Entrer votre taille en mètres"))
```

2.2 Les sorties

Pour afficher une variable, on utilise la fonction `print()`.

■ **Exemple 6:**

```
a = 5.2
b = 3
print(a)
print(2*b)
print(a,2*b)
```

Ce programme affichera `5.2`, puis la ligne suivante : `6` puis la ligne suivante : `5.2 6`.

On peut également utiliser la syntaxe suivante lorsqu'on veut mélanger du texte avec des variables :

```
print(f"L'élève {nom} a {age} ans et mesure {taille}m.")
```

On écrit donc : `print(f"texte {variable}")`

▶ Exercice 4

Écrire un programme qui :

- Demande à l'utilisateur de rentrer la longueur d'un carré ;
- affiche le périmètre du carré ;
- affiche l'aire du carré.

▶ Exercice 5

Écrire un programme qui demande à l'utilisateur d'entrer les mesures des trois côtés a, b et c d'un triangle et qui affiche le périmètre puis l'aire du triangle par la formule de Héron : $\sqrt{p(p-a)(p-b)(p-c)}$, où p est le demi-périmètre du triangle : $p = \dfrac{a+b+c}{2}$.

On commencera le programme par la ligne :

```
from math import sqrt
```

pour utiliser la fonction sqrt (racine carrée)

▶ Exercice 6

Écrire un programme qui :

- demande à l'utilisateur d'entrer les coordonnées de deux points (abscisses, ordonnées) ;
- affiche les coordonnées du milieu du segment ;
- affiche la distance entre les deux points.

3 Instructions conditionnelles

La structure la plus simple est :

```
if condition:
   Instructions
```

■ **Exemple 7:**

```
if n%2==0:
   n=n/2
```

Mais une structure plus fréquente est :

```
if condition:
   Instructions 1
else:
   Instructions 2
```

■ **Exemple 8:**

```
if n%2==0:
   n=n/2
else:
   n=3*n+1
```

On peut également créer des « cas » avec une structure `if - elif - ...- else`.

▶ Exercice 7

Créer un programme demandant l'entrée du nom, de l'âge et du sexe (M/F) à l'utilisateur et qui permettra d'afficher un message d'accueil personnalisé :

```
Bonjour Monsieur Dupont
Bonjour Madame Dupont
Bonjour Mademoiselle Dupont
```

▶ Exercice 8

Écrire un programme qui demande à l'utilisateur d'entrer deux nombres l'un après l'autre puis qui affiche la plus grande des deux valeurs.

4 Boucle For

Une des structures de répétition proposées en Python est la boucle `for`, aussi appelée « boucle bornée » ou « non conditionnelle ». On décide en amont du nombre d'itérations effectuées par la boucle.

```
for i in range(n):
   Instructions
```

Ici, les instructions seront répétées n fois. Concrètement, `range(n)` crée une pseudo-liste de valeurs : $\{0,1,2,\ldots,n-1\}$ qui est parcourue par la variable i.

Plus généralement, on peut utiliser la syntaxe suivante :

```
for element in objet:
   Instructions
```

■ **Exemple 9:**

```
for car in "Bonjour":
   print(4*car
```

Remarque 1 (`range()`).

- `range(n)` renvoie une séquence d'entiers entre 0 et $n-1$ compris (donc n valeurs)
- `range(d,f)` renvoie une séquence d'entiers entre d et $f-1$ (donc $f-d$ valeurs)
- `range(d,f,p)` renvoie une séquence d'entiers entre d et $f-1$ avec un pas de p.

Remarque 2 (Interrompre une boucle).

L'instruction `break` permet d'interrompre une boucle.

▶ Exercice 9

Écrire un programme qui va afficher :

```
Ligne 1
Ligne 2
...
Ligne 20
```

▶ Exercice 10

Écrire un programme qui demande à l'utilisateur de rentrer un certain nombre (demandé à l'utilisateur) de valeurs consécutivement puis qui va afficher la somme des valeurs entrées puis la moyenne des valeurs entrées.