Type de données en python

Pour connaître le type d'un objet en python on utilise `type`

type entier : int

```
>>> type(2)
<class 'int'>
```

type réel : `float`

```
>>> type(2.0)
<class 'float'>
```

chaine de caractère : `str`

```
>>> type('chaine')
<class 'str'>
```

type booléen : `'bool'`

```
>>> type(True)
<class 'bool'>
```

Variables

→ Une variable est identifiée par un nom

→ Un nom de variable en `python` obéit à certaines règles :

- c'est une suite de lettres et de chiffres sans espace
- il commence par une lettre
- majuscule $\neq$ minuscule
- certains noms sont réservés à `python`
  (Il y en a 32 comme `True, and, ...`)

→ En plus des règles, il y a des usages qu'il faut essayer de respecter :

- nom de préférence « relativement » court mais explicite
- nom écrit plutôt en lettres minuscules avec un `_` à la place des espaces : `joueur1 , nb_joueurs`

Affectation variables

→ On appelle affectation l'attribution d'une valeur à une variable.

→ En `python`, on utilise le signe égale. Lorsqu'on tape `a=8`, l'ordinateur « fait » les choses suivantes :

- créé un nom de variable *a*
- attribue à *a* le type `int`
- crée et mémorise la valeur 8
- établit un lien (avec un pointeur) entre le nom et la valeur.

→ Pour afficher la valeur d'une variable, on utilise la fonction `print`

→ La valeur/type d'une variable peut changer au cours du temps

→ en `python`, on peut affecter plusieurs variables en même temps

Opérations et expressions

→ Un expression est une combinaison d'opérations appliquées à des valeurs et/ou à des variables

→ Les opérateurs classiques en python sont

  → Opérateurs arithmétiques `+,-,*, **,/,//,%`

  → la concaténation avec + pour les chaînes de caractères

  → les opérateurs logiques `and, or, not` pour les booléens

  → les opérateurs de comparaisons : `<,<=,==,!=` qui renvoient un booléen

→ Lors de l'affectation d'une variable, on peut mettre une expression à droite du signe égale =

fonctions mathématiques

→ Pour utiliser certaines fonctions mathématiques, il faut importer le module `math` :

```
>>> import math
>>> math.sqrt(4)
2.0
```

→ Dans le module math on trouve `sqrt`,`log`,`exp` etc.

→ Dans le module random on trouve les fonctions :

- `random` : qui renvoie un nombre aléatoire dans $]0,1[$
- `randint(n,m)` : qui renvoie un entier aléatoire entre n et m

```
>>> import random
>>> random.random()
0.7361642307042312
```

```
>>> import random
>>> random.randint(2,6)
5
```

Les instructions de choix

→ Pseudo-code vs `python`

Si . . . alors . . .

Pseudo-code

si $a > 0$ **alors**
 | affiche('a est positif')
sinon
 | affiche('a est négatif')
fin

`python`

```
if a>0:
    print('a est positif')
else:
    print('a est négatif')
```

→ **L'indentation est indispensable en `python`**

elle indique le début et la fin des blocs d'instructions.

Indentation = 4 espaces

→ On peut rajouter des conditions dans un `if`.

Tant que

→ Syntaxe pseudo-code et `python`

Tant que

Pseudo-code

$i = 1$
tant que $i \leq 5$ **faire**
  affiche(i)
  $i = i + 1$
fin

`python`

```
i=1
while i<=5:
    print(i)
    i=i+1
```

→ **L'indentation est indispensable en `python`**

Indentation = 4 espaces