MÉMENTO PYTHON

SIO1, novembre 2023

Principaux types

`int` : entier
`float` : flottant
`bool` : booléen
`str` : chaîne de caractères
`list` : liste
`range` : plage d'entiers

Affectation

`a = 2` : affectation simple
`a, b = 2, 3` : affectations multiples
`a += 1` : équivaut à `a = a + 1`
`a *= 2` : équivaut à `a = a * 2`

Transtypage

`int("12")` vaut `12`
`str(12)` vaut `"12"`
`float("12.5")` vaut `12.5`

Opérations

quotient : `a = 11 // 4 # a vaut 2`
reste : `a = 11 % 4 # a vaut 1`
division : `a = 11 / 4 # a vaut 2.75`
puissance : `a = 2 ** 10 # a vaut 1024`

Entrées / sorties

```
print(f"les valeurs sont {a} et {b}")
```

affiche la valeur de `a` et de `b`

```
a = input("Entrez une valeur")
```

stocke une entrée utilisateur de type `str` dans `a`

Commun à str et list

`len(a)` : longueur de l'objet
`a[i]` : élément d'indice `i` de l'objet
`a[-1]` : dernier élément de l'objet

Tests

Opérateur	Signification
`or`	ou logique
`and`	et logique
`not`	négation logique
`<`	strictement inférieur
`<=`	inférieur ou égal
`>`	strictement supérieur
`>=`	supérieur ou égal
`==`	égal
`!=`	différent
`in`	appartient à
`not in`	n'appartient pas à

Test `if ... else ...` classique

```
if <condition>:
    # bloc conditionnel 1
else: # facultatif
    # bloc conditionnel 2
```

Test `if ... elif ... else ...`

```
if <condition 1>:
    # bloc conditionnel 1
elif <condition 2>:
    # bloc conditionnel 2
# autant de elif que nécessaire
else: # facultatif
    # bloc conditionnel final
```

Boucle `while`

```
while <condition>:
    # bloc conditionnel
# en sortie de boucle la condition
# n'est plus vérifiée
```

Boucle `for ... in objet` avec `objet` de type `str` ou `list`

```
for x in objet:
    # x prend successivement toutes
    # les valeurs des éléments de
    # l'objet
```

Boucle `for ... in range()`

```
for i in range(...):
    # i prend successivement toutes
    # le valeurs du range
```

Utilisation de `range`

`range(<debut>,fin,<increment>)`
par défaut `debut=0` et `increment=1`

dans `range(<debut>,fin,<increment>)` :
on part de la valeur de début, appelons la `val`
tant que `val < fin` :
- ajouter `val` à la plage
- ajouter `increment` à `val`

Utilisation de `list`

```
a = [] # liste vide
a.append(objet) # ajoute objet à a
a.remove(objet) # retire objet à a
a.insert(i,objet) # insère objet à
# l'indice i
del a[i] # retire l'élément d'indice i
```