Informatique

Fiche

1 – Passage de Python 2.x à Python 3.x

Objectifs :

1. Connaître les principaux changements

I. Les commandes d'interfaces

1 Les « print »

Important : *Changements...*

⇒ `print` devient une fonction. Il faut donc l'utiliser avec des parenthèses.

a) Dans la version 2.x

code Python

```
>>> print 1.5                    # Affiche un élément
1.5
>>> print 1.5,2,3.14             # Plusieurs éléments séparés par des virgules
1.5 2 3.14                               # ...affiche la liste des éléments à afficher
>>> print (1,2,3)                # Si on mets des parenthèses...
(1,2,3)                          # ... renvoie un t-uple.
```

b) Dans la version 3.x

code Python

```
>>>print(1.5)                    # Affiche un élément
1.5
>>> print(1.5,2,3.14)            # Plusieurs éléments séparés par des virgules
1.5 2 3.14                               # (attention ce n'est plus un t-uple)
>>> print((1,2,3))               # Affiche d'un t-uple
(1,2,3)
```

2 Afficher les « range »

Important : *Changements...*

La fonction `range` fonctionne encore très bien avec les fonctions d'itérations (boucles `for`,...).
Mais elle ne s'affiche pas dans le terminal comme on le souhaiterait.
⇒ Il faut explicitement lui imposer un type `list` pour l'afficher.

a) Dans la version 2.x

code Python

```
>>> range(5,10)                    # Si on affiche un "range"...
[5, 6, 7, 8, 9]                    # ... cela nous renvoie la liste
```

b) Dans la version 3.x

code Python

```
>>> range(5,10)                    # Si on affiche un "range"...
range(5,10)                        # ... cela nous renvoie le range tel quel.
>>> list(range(5,10))              # Si on explicite le type "list"...
[5, 6, 7, 8, 9]                    # ... cela nous renvoie la liste
```

3 Affichage des chaînes de caractères avec accents

Important : *Changements...*

Le codage est déjà au format `unicode`. Plus besoin de préciser le codage lorsque l'on veut afficher une chaîne de caractères avec accents.

a) Dans la version 2.x

code Python

```
>>> print "éléphant"                 # Sans codage particulier
~A©l~A©phant             # ... problème d'accents
>>> print(u"éléphant")              # avec codage unicode
éléphant                 # ... cela s'affiche bien
```

b) Dans la version 3.x

code Python

```
>>> print("éléphant")                # Sans codage particulier
éléphant
```

II. Calculs

1 La division

Important : *Changements...*

Maintenant, le symbole divisé `/` renvoie nécessairement un `float`. Notamment dans le cas des divisions d'entiers, le résultat sera bien la division classique, et non plus la division euclidienne. Pour retrouver le comportement de division euclidienne, on utilise le symbole `//`.

Note : il est possible d'avoir le même comportement avec Python 2.x en mettant en entête

a) Version 2.x

code Python

```
>>> 1./2.                  # Division de floats...
0.5                                # ...renvoie un float
>>> 1/2                    # Division d'entiers...
0                                  # ...renvoie un entier (division euclidienne)
```

b) Version 3.x

code Python

```
>>> 1/2                    # Division (entier ou float)...
0.5                                # ...renvoie un float
>>> 2/1
2.0                                # ...même si le résultat pourrait être entier
>>> 1//2                   # Division avec double //
0                                  # ... renvoie le quotient de la division euclidienne.
```