Langages de Script - Memento Python

C. Tasson

I) Introduction à Python

a) Installer Python et se documenter

Nous utilisons Python version 3.0. Dans les systèmes Linux, Python est généralement déjà installé par défaut. Pour installer Python sous Windows ou Mac, il suffit de télécharger la version 3.0 disponible à l'adresse : `http://www.Python.org/`.

Parmi les nombreuses ressources disponibles sur Internet, on peut citer :

1. la traduction français du livre *A bite of* Python : `http://www.swaroopch.com/notes/Python_fr:Table_des_Matières`.
2. l'ouvrage *Apprendre à programmer avec* Python de Gérard Swinnen : `http://inforef.be/swi/python.htm`.
3. le tutoriel `http://docs.python.org/py3k/tutorial/index.html`.

b) Écrire des programmes

Python possède un interpréteur qui peut être lancé en utilisant la ligne de commande `python3.1`. On entre alors dans un mode interactif dans lequel on peut tester les instructions.

Pour écrire un programme Python, on ouvre un fichier texte dans un éditeur de texte à côté du terminal. On écrit les instructions, puis on sauvegarde le fichier en choisissant un nom de la forme `nom_du_fichier.py`. Remarquez que l'extension est indispensable. Enfin, on lance l'interprétation dans le terminal via la ligne de commande `python3.1 nom_du_fichier.py`.

L'encodage des caractères par un ordinateur n'est pas encore unifié. La version Python 3.1 est prévue pour fonctionner avec l'`UTF-8`. Pour éviter tout problème, on veillera à configurer son éditeur de texte dans cet encodage.

c) Règles de bonne programmation

Un programme doit pouvoir être lu et relu, pour cela :

* Utiliser des noms de variables et de fonctions *explicites*.
* *Commenter* les différentes étapes du programme avec le caractère `#`.
* *Documenter* les fonctions.

Enfin, quelques règles de présentation doivent être respectées :

* Placer un espace après une virgule, un point-virgule ou deux-points ;
* Ne pas placer d'espace avant une virgule, un point-virgule ou deux-points ;
* Placer un espace de chaque côté d'un opérateur ;
* Ne pas placer d'espace entre le nom d'une fonction et sa liste d'arguments ;
* Indenter (python est un langage indenté, ce qui signifie que l'indentation est obligatoire, elle est normalement de 4 espaces à paramétrer dans son éditeur de texte).

II) Calcul numérique

Opérations

Addition	`+`	Soustraction	`-`
Multiplication	`*`	Division	`/`
Puissanc	`**`		

Division Entière

Parti ntièr	`//`	R st	`%`

Conversions

En nti r	`int()`	En flottant	`flo t()`

III) ariables et Fonctions

Affectation

```
x = 2
```

Retourner la valeur d'une variable

```
return x
```

Définition d'une fonction

```
def <nom_fonction>(< rgument_1>,< rgument_2>,...,< rgument_n>):
    """Description sur plusieurs lignes de l  fonction"""
    <instructions>
```

Documentation sur une fonction

```
help(nom_fonction)
```

I) Calcul propositionnel et conditionnelle

Constantes booléennes

Tru Fals

Comparaisons

Égalité	`==`	Différ nc	`!=`
Strict m nt supéri ur	`>`	Strict m nt inféri ur	`<`
Supéri ur ou égal	`>=`	Inféri ur ou égal	`<=`

Opérateurs booléens

Non	`not`		
Ou	`or`	Et	`nd`

Instruction conditionnelle

```
if <condition_1>:
    <instructions_1>
elif <condition_2>:
    <instructions_2>
elif <condition_3>:
    <instructions_3>
...
else:
    <instructions_n>
```

) Types énumératifs et algorithmes itératifs

Types `list` et `string`

List vid	`[]`	Chaîn vid	`''`
Longu ur	`len(T)`	Concaténation	`+`
Extraction	`T[i]`	Extraction	`T[deb:fin:p s]`

Spécifique au type `list`

Ajout d'un élém nt	`L. ppend( )`	Suppr ssion	`del(L[i])`
Copi sans alias	`L1 = L2[:]`		

Type `r nge`

`r nge(deb,fin+1,p s)`

Boucle de répétition

```
for <element> in <list_string_r nge>:
    <instructions>
```

Boucle *Tant que*

```
while <condition>:
    <instructions>
```