Langages de script
TP n° 1 : Premiers pas en python

Coordonnées des enseignants
Peter Habermehl <Peter.Habermehl@liafa.univ-paris-diderot.fr.fr>
Sophie.Laplante <Sophie.Laplante@liafa.univ-paris-diderot.fr.fr>
Anne Micheli <Anne.Micheli@liafa.univ-paris-diderot.fr.fr>

Documents à télécharger
Tous les énoncés et les documents à télécharger seront disponibles sur didel `http://didel.script.univ-paris-diderot.fr` (le code du cours est LS6). Inscrivez-vous dès la première semaine pour faciliter la communication avec l'équipe enseignante.

Version de Python Nous utiliserons la version 3.4.2.

Exercice 1 : Premier programme.

Ce premier exercice a pour but de découvrir les différents moyens d'utiliser Python.

1. Lancez la version 3 de l'interpréteur Python (mode interactif) grâce à la commande python3 depuis un terminal. Écrivez et exécutez une commande pour afficher la ligne de texte `Hello World!`. Quittez l'interpréteur en tapant `^D` (touches `Control` et `D` simultanément) ou en appelant la fonction `exit()`.
2. Créez un fichier `hello.py` contenant comme unique ligne de texte la commande précédente. Sauvegardez ce fichier, et tapez dans un terminal la commande `python3 hello.py`. Observez le résultat.
3. Lancez à nouveau l'interpréteur Python, et tapez la commande `import hello`. Que se passe-t-il ? Grâce à la commande `dir()`, affichez la liste des identificateurs connus. Que constatez-vous ?
4. Rendez le fichier `hello.py` exécutable et tentez de l'exécuter. Que se passe-t-il ? Rajoutez la ligne `#!/usr/bin/env python3` au début du fichier, puis réessayez.

Exercice 2 : Mode calculette.

Placez-vous dans un interpréteur Python.

1. Faites les calculs `3/4`, `4/2`. Que constatez vous ? Essayez encore `3//4`, `4//2`. Quelle est la différence ?
2. Calculez le reste de la division de la somme de 19875 et 77569 par 7. Si vous trouvez un nombre supérieur à 6, c'est évidemment qu'il y a une erreur... Que pouvez-vous en déduire sur les opérateurs + et % ? Qu'en est-il des autres opérateurs arithmétiques (faites des tests) ?
3. Faites le calcul $\sqrt{3}+56/9.0\times|-1/4|$, soit en une seule fois, soit en plusieurs fois en utilisant des variables (**note** : consultez l'aide sur les nombres entiers (`help(int)`) et sur l'opérateur "puissance" (`help("POWER")`) pour trouver la syntaxe nécessaire).

4. Testez les instructions `1j**2`, `(1+2j).imag` et `(1+2j).real`. Que représente `j` ?
5. Tapez `hex(30)`. Quel format est utilisé pour représenter les hexadecimaux ? Les octaux ? Les chaînes binaires ?
6. La fonction int(string,int) permet de faire la conversion de base. Quel est le résultat des expressions suivantes ? `int('101',2), int('101',8), int('101', 16)`.
7. Ecrivez une expression qui convertit la chaine binaire '111001' à une chaîne repreprésentant la même valeur en hexadécimal.

Exercice 3 : Chaînes de caractères.

1. Créez une variable `h` contenant la chaîne de caractères `"Hello"` et une variable `w` contenant `"World"`. En utilisant la concaténation de chaînes de caractères et ces deux variables (entre autres), créez une nouvelle variable `hw` contenant la chaîne `"Hello World!"`. Comparez le résultat des instructions `hw` et `print(hw)`.
2. Testez la fonction `len(`*chaine*`)`. A quoi sert-elle ?
3. En utilisant la syntaxe *chaine*`[`*debut*`:`*fin*`]` et la variable `hw`, affichez les chaînes `"Hell"`, `"orld!"` et `"llo Wo"`. Testez ensuite les instructions `print(hw[:4])`, `print(hw[-4:])`,
   `print(hw[:])` et `print(hw[:4] + hw[4:])`. Qu'observez-vous ?
4. Définissez la chaîne `x='12345678'`. En n'utilisant que la syntaxe des "slice", donnez des expressions qui affichent :
   (a) un symbole sur deux de x
   (b) les symboles de x dans l'ordre inverse.
   (c) un symbole sur deux en partant de la fin, dans l'ordre inverse.
5. Quel est le résultat de l'expression `h+2*(h+2*w)` ?
6. Créez une variable contenant la chaîne de caractères suivante (en respectant les retours à la ligne) :

   Dans le vieil étang,
   Une grenouille saute,
   Bruit dans l'eau.

   Utilisez d'abord des guillemets triples (`'''` ou `"""`), puis des guillemets simples.
7. Consultez l'aide de la fonction `input`, puis utilisez cette fonction pour lire un message donné par l'utilisateur puis l'afficher à l'écran. Testez ensuite la commande `print(input("Ca va ?"))`
8. La fonction `int(`*objet*`)` permet de traduire un objet (par exemple une chaîne) en valeur entière (lorsque c'est possible). Écrivez une suite d'instructions permettant de lire un entier puis d'afficher le **double** de sa valeur.

Exercice 4 : Variables, expressions

Vous pouvez utiliser `del Z` pour supprimer la variable Z lors de vos tests.

1. Expliquer la différence entre une variable dont la valeur est `None` et une variable qui n'a pas de valeur (variable non-définie). Que se passe-t-il lorqu'on évalue `print(Z)` lorsque `Z` est non-définie ? Lorsque `Z` vaut `None` ? Lorque `Z` contient la chaîne vide ?

2. Écrire une expression qui évalue à True lorsque `Z` vaut `None` et False sinon.
3. Écrire une expression qui évalue à True lorsque `Z` contient la chaîne vide et False sinon.
4. Écrire une expression qui évalue à la chaîne "Sans valeur" si la variable `Z` vaut `None`, "Chaîne vide" si elle contient la chaîne vide, et "Autre" dans tous les autres cas. (Écrire une expression et non pas un segment de code qui affiche la valeur.) L'expression retourne une erreur si la variable n'est pas définie.
5. Écrire une expression qui retourne True si $x > 0$, False si $x \leqslant 0$ et None si x vaut None.
6. Écrire une expression qui retourne True si $x > 0$ et False si $x <= 0$ ou si x vaut None.

Exercice 5 : Première boucle, première fonction

1. Créez une variable `x` de valeur `100`. Calculez la somme S_x de tous les entiers de 1 à x (en les additionnant tous). Vérifiez que la formule $S_x = x(x+1)/2$ pour cette somme est satisfaite.
2. Que signifient `range(10)` et `range(3, 40, 5)` ? Confirmez votre opinion.
3. Programmez deux fonctions : `somme_bete(x)` et `somme_rapide(x)` qui calculent la somme de tous les entiers de 1 à x avec les deux méthodes vues dans le premier point.
4. Testez vos fonctions pour quelques valeurs de `x`.
5. Qu'affiche l'aide en ligne `help(somme_bete)` ?
6. Modifiez le code de la fonction pour que l'aide en ligne affiche un bref descriptif de votre fonction.

Exercice 6 : Fonctions sur les chaînes

Écrire les fonctions

1. `sans_e(str)` qui renvoie la chaîne de caractères en argument sans les 'e'.
2. `compte_espace(str)` qui renvoie le nombre d'espaces dans la chaîne de caractères passée en argument.
3. Écrire une fonction `palindrome(str)` qui retourne `True` si la chaîne de caractères passée en argument, sans les espaces, est un palindrome, et `False` sinon.

Indications : penser à utiliser les slices et les fonctions du module ***`str`****.*

Exercice 7 : Horloge simple

1. Écrire une fonction `sec_to_hms` qui convertit un temps exprimé en secondes en un temps exprimé en h :m :s.
2. Écrire la fonction opposée `hms_to_sec`.
3. En utilisant le module `time`, écrire une fonction `now` qui retourne la date et l'heure de l'ordinateur, sachant que le nombre de secondes écoulées entre le 1er janvier 1970 0h0min0s et 1er janvier 2014 0h0min0s est 1388530800.
   Vous pouvez trouver la documentation sur le module `time` ici :
   `http://docs.python.org/3/library/time.html`

Exercice 8 : Scripts et modules

Le script `time.py` se trouve sur la page du cours sur Didel. Télécharger le script et le compléter.

```
#time.py

def convert:
    pass

def now:
    pass

if __name__=="__main__":
   print("ici j'écrirai l'heure")
```

1. À quoi sert la variable `__name__`, quelles sont ses valeurs possibles.
2. Compléter le script pour que la commande `./time.py`, exécutée dans un terminal, affiche sur la sortie standard l'heure actuelle en hms en la mettant à jour toutes les 5s.