TP de Langages de script n° 1 : premiers pas en PYTHON

Coordonnées des enseignants
Ines Klimann `<klimann@liafa.jussieu.fr>`
Julien Cristau `<jcristau@liafa.jussieu.fr>`
Stéphane Jacob `<stephane.jacob@inria.fr>`
Bruno Bernardo `<bruno.bernardo@lix.polytechnique.fr>`

Documents à télécharger
Tous les énoncés et les documents à télécharger seront disponibles sur didel
`http://didel.script.univ-paris-diderot.fr`.

Exercice 1 : Premier programme.

Ces premiers exercices ont pour but de découvrir les différents moyens d'utiliser Python.

1. Lancez l'interpréteur Python (mode interactif) 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 `python hello.py`. Observez le résultat.
3. Lancez à nouveau l'interpréteur Python, et tapez la commande `import hello`. Que s'est-il passé ? Grâce à la commande `dir()`, affichez la liste des identificateurs connus. Que constatez-vous ?
4. Sans quitter l'interpréteur, modifiez `hello.py` pour que le message affiché soit maintenant `Hello everyone!`, puis tapez à nouveau la commande `import hello`. Que s'est-il passé ? Essayez à présent la commande `reload(hello)` (attention aux parenthèses !).
5. Rendez le fichier `hello.py` exécutable et testez l'exécuter. Que s'est-il passé ? Rajoutez la ligne `#!/usr/bin/env python` au début du fichier, puis réessayez.

Indications

- La commande qui sert à afficher une chaîne de caractères est `print`. C'est pas une fonction, elle n'est donc pas forcément suivie de parenthèses. On délimite une chaîne de caractères par des guillemets simples ou doubles.
- Rappel d'IS1 : pour rendre un fichier `monscript.py` exécutable, il faut taper dans un terminal : `chmod u+x /chemin/vers/monscript.py`. On exécute ensuite le script en tapant : `/chemin/vers/monscript.py` (par exemple `./monscript.py` à partir du répertoire contenant le fichier).

Note : Les exercices suivants utilisent exclusivement le mode interactif de Python.

Exercice 2 : Mode calculette.

Placez-vous dans un interpréteur Python.

1. Faites les calculs `3/4` et `3/4.0`. Comment comprenez-vous le résultat ?

2. Calcul z l r st d la division d la somm d 19875 t 77569 par 7. Si vous trouv z un nombr supéri ur à 6, c' st évid mm nt qu'il y a un rr ur... Qu pouv z-vous n déduir sur l s opérat urs + t % ? Qu' n st-il d s autr s opérat urs arithmétiqu s (fait s d s t sts) ?
3. Fait s l calcul $\sqrt{3}+56/9.0\times|-1/4|$, soit n un s ul fois, soit n plusi urs fois n utilisant d s variabl s (**note** : consult z l'aid sur l s nombr s nti rs (`help(int)`) t sur l'opérat ur "puissanc " (`help("POWER")`) pour trouv r la syntax néc ssair).
4. Affich z la val ur d la variabl `_` (tir t bas). À quoi s rt- ll à votr avis ? Expérim nt z.
5. T st z l s instructions `1j**2`, `(1+2j).im g` t `(1+2j).re l`. Qu r prés nt `j` ?

Indications

- La fonction `bs()` p rm t d calcul r la val ur absolu d'un nombr . Pour l calcul d la racin carré , il y a au moins d ux possibilités. On p ut utilis r l'opérat ur `**` (puissanc) d'un façon approprié , ou bi n la fonction `sqrt` du *module* `m th`.

Exercice 3 : Chaînes de caractères.

1. Cré z un variabl `h` cont nant la chaîn d caractèr s `"Hello"` t un variabl `w` cont nant `"World"`. En utilisant la concaténation d chaîn s d caractèr s t c s d ux variabl s (ntr autr s), cré z un nouv ll variabl `hw` cont nant la chaîn `"Hello World!"`. Compar z l résultat d s instructions `hw` t `print hw`.
2. T st z la fonction `len(`*chaine*`)`. A quoi s rt- ll ?
3. En utilisant la syntax *chaine*`[`*debut*`:`*fin*`]` t la variabl `hw`, affich z l s chaîn s `"Hell"`, `"orld!"` t `"llo Wo"`. T st z nsuit l s instructions `print hw[:4]`, `print hw[-4:]`, `print hw[:]` t `print hw[:4] + hw[4:]`. Qu'obs rv z-vous ?
4. Cré z un variabl cont nant la chaîn d caractèr s suivant (n r sp ctant l s r tours à la lign) :

   Dans l vi il étang,
   Un gr nouill saut ,
   Bruit dans l' au.

   Utilis z d'abord d s guill m ts tripl s (`'''` ou `"""`), puis d s guill m ts simpl s.
5. Consult z l'aid d la fonction `r w_input`, puis utilis z c tt fonction pour lir un m ssag donné par l'utilisat ur puis l'affich r à l'écran. T st z nsuit la command `print r w_input("C v ?")`
6. La fonction `int(`*objet*`)` p rm t d traduir un obj t (par x mpl un chaîn) n val ur

Exercice 4 : Première boucle.

1. Créez une variable `x` de valeur `'toto'`. La fonction `type` vous donne le type de cette variable. Comparez le résultat de l'application de la fonction `dir` sur `x` et sur son type.
2. Comment met-on une chaîne de caractères en minuscules ?
3. Téléchargez le fichier `grenades_lettres.py` et importez le module correspondant dans l'interpréteur python. Regardez l'aide en ligne de ce module.
4. À l'aide du module `grenades_lettres`, écrivez une fonction `grenade_message` qui prend en argument une chaîne de caractères et réécrit le texte en grandes lettres.

Exemple : 'toto' sera réécrit

```
*****  **  *****  **
  *   *  *   *   *  *
  *   *  *   *   *  *
  *   *  *   *   *  *
  *    **    *    **
```

Pour cela il faut évidemment écrire le texte ligne à ligne.

Indications
- Pour consulter l'aide en ligne, on utilise bien sûr la fonction `help()`.
- Pensez à la fonction `range` décrite plus haut pour réaliser des boucles sur les entiers.

Exercice 5 : Codes secrets.

1. À partir de votre terminal (et sans avoir lancé l'interpréteur Python), consultez grâce à `pydoc` l'aide en ligne des fonctions `chr` et `ord`.
2. Dans un interpréteur Python créez la chaîne de caractères `message` contenant la valeur `'ceci est mon message chiffrer'`. À l'aide d'une boucle `for` sur cette chaîne, chiffrez-la par un décalage de 3 (*chiffre de César*). Par exemple la lettre `a` sera chiffrée par la lettre `d` (et la lettre `x` par la lettre).
3. Créez maintenant une chaîne de caractères `clef` dont la valeur est `'secret'`. Chiffrez la chaîne `message` par décalage grâce à la clef `clef` (*chiffre de Vigenère*). Ce chiffrement se fait en décalant la i^{e} lettre du message grâce à la i^{e} lettre de la clef (on reprend au début de la clef quand on a fini de lire celle-ci). Mathématiquement on peut écrire : lettre chiffrée = (lettre + clef) modulo 26.
4. Un chiffrement, plus ancien que ces deux exemples est dû à Polybius, consistant à utiliser un carré de chiffrement :

	1	2	3	4	5
1	a	b	c	d	
2	f	g	h	ij	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

Avec ce chiffrement, le chiffré de e est 51 et celui de r est 24. Définissez deux fonctions `chiffre_polybius` et `dechiffre_polybius` telles que si l'on passe à la seconde le message `'Ceci est un message extremement secret !'`.

Indications

- Pour simplifier, on ne modifiera pas les caractères blancs (espace, tabulation et retour chariot), la ponctuation...

Exercice 6 : Premier module.

1. Créez un module `tp1` dans un fichier `tp1.py`, en suivant la structure donnée en cours. Ne pas oublier d'inclure une chaîne de documentation décrivant le module.
2. Définissez une variable `auteur` contenant votre nom, ainsi qu'une fonction `copyright()` affichant à l'écran que vous êtes l'auteur de ce programme et que vous poursuivrez en justice quiconque le copiera sans autorisation. N'oubliez pas d'inclure une chaîne de documentation décrivant la fonction.
3. Faites en sorte que, lorsque le fichier est exécuté en tapant son nom dans le *shell*, il affiche le nom de son auteur et le message de copyright.
4. Importez le fichier dans l'interpréteur Python. Que se passe-t-il ? Modifiez votre script pour que les commandes de la question précédente ne s'exécutent que si le programme est appelé depuis le *shell*, et pas quand il est importé dans l'interpréteur.
5. Consultez l'aide en ligne de votre module grâce à la commande `help` et son environnement local grâce à la commande `dir()`. Importez ensuite la variable `auteur` du module directement dans l'environnement global grâce à la commande `from ...import ...`

Indications

- Pour créer un texte explicatif décrivant le module et accessible grâce à l'aide en ligne, on fait commencer le fichier par une chaîne de caractères (qui peut être sur plusieurs lignes).
- Pour documenter une fonction, on s'y prend de la même façon que pour documenter un module : on fait débuter la définition de la fonction (juste après l'en-tête) par une chaîne de caractères.
- Indice : quand un module Python est exécuté directement depuis le terminal, la variable prédéfinie `__name__` prend la valeur `__main__`. Quand un module est importé, cette variable prend une valeur différente ....

Exercice 7 : Mot sans cube.

Pour cet exercice vous écrirez votre code dans un module que vous pourrez importer dans l'interpréteur.

On considère les mots sur un alphabet à deux lettres $\{a, b\}$. On dit qu'un mot (c'est-à-dire une suite de lettres) possède un cube s'il possède un facteur de la forme uuu où u est un mot non vide. Par exemple *baaa* et *bababa* possèdent un cube alors que *bababb* est sans cube. On peut montrer qu'il existe un mot infini sans cube (mais ce n'est pas l'objet de cet exercice). Nous allons être plus modestes et écrire une fonction qui teste si un mot est sans cube [1].

1. Que fait la méthode `endswith` du module `str` ? Trouvez la méthode symétrique.

[1] Dans un TP ultérieur nous construirons la liste des mots sans cube d'une longueur donnée.

2. Dans un int rprét ur Python, r gard z c qu donn `3*' b'`.
3. Écriv z un fonction `est_s ns_prefixe_cube` qui pr nd n argum nt un chaîn d caractèr s t dét rmin si ll possèd un cub comm préfix .
4. Écriv z un fonction `est_s ns_cube` qui pr nd n argum nt un chaîn d caractèr s t dét rmin si ll st sans cub . Pour c la vous pourr z t st r l s suffix s d l'argum nt grâc à la fonction précéd nt .

Exercice 8 : Anagrammes.

1. Écrir un fonction dét rminant si d ux mots sont d s anagramm s.
2. À l'aid d c tt fonction t d la list `dictionn ire` situé dans l modul `dictionn ire.py`, écrir un fonction qui ch rch l s anagramm s d'un mot donné n argum nt.