Lycée Louis-Le-Grand, Paris 2013/2014
MPSI 4 – Informatique pour tous
J.-P. Becirspahic, A. Troesch

TP n° 1 : À la découverte de Python

Lancez Python en cliquant sur l'icône « Pyzo » situé sur le Bureau. La fenêtre du haut est une console (ou terminal, ou shell Python) que nous dénommerons « Interpréteur Python ». Dans cet interpréteur, vous pouvez taper directement des opérations ou des instructions, qui seront effectuées immédiatement. La fenêtre du bas vous permet d'écrire des programmes complets mémorisés dans des fichiers.

Exercice 1 – Utilisation de la console comme calculatrice

1. À l'aide de la console, calculer $54 + 23$, $24 - 7$, 123×87, $34/5$.
2. À quoi correspondent les opérations `//`, `%` et `**` ?
3. Peut-on indiquer plusieurs calculs différents sur une même ligne ? Si oui, quel(s) séparateur(s) mettre ? Qu'observez-vous ?

Exercice 2 – À la découverte des Variables

Une variable est la donnée d'une place en mémoire, destinée à stocker une valeur. La valeur d'une variable peut changer au cours du temps. L'action de donner une valeur à une variable s'appelle « l'affectation », et s'effectue avec le signe =.

1. Créer une variable x de valeur 3. Peut-on écrire l'égalité d'affectation dans le sens qu'on veut ?
2. Créer une variable y égale à $7*x$, et afficher sa valeur. Modifier la valeur de x. Quel est l'effet de cette modification sur la valeur de y ?
3. À l'aide de la fonction `help`, comprendre ce que fait la fonction `id`, et la tester sur la variable x. De même pour la fonction `type`.
4. Rajouter 1 à x. Quel est l'effet sur le type et l'identifiant ? Même question en rajoutant 0.5.
5. Comprendre l'effet sur x des opérations `+=`, `-=`, `*=`, `/=`.
6. Échanger (sans affectation simultanée) le contenu des variables x et y.
7. • Trouver deux façons différentes d'affecter une valeur à x et à y sur une même ligne de commande.
   • Les variables x et y ayant initialement les valeurs 1 et 4, tester ces deux façons pour faire les affectations $x = 2x + y$ et $y = x - 3$. Les deux méthodes sont-elles équivalentes ?

   L'instruction `x,y= 3,4` *est interprétée comme une unique affectation portant sur le couple* (x, y).
8. Quel est le type de l'objet (x, y) ?
9. Proposer une méthode pour échanger le contenu des variables x et y sans avoir besoin d'introduire une variable auxiliaire.

Exercice 3 – Premiers affichages

1. Définir une variable `Bonjour` de valeur 0.
2. Quel est l'effet de l'instruction `print(Bonjour)` ? Quels délimiteurs mettre autour du mot Bonjour pour afficher le mot Bonjour et non sa valeur ? (deux réponses possibles, voire 3)
3. Comparer le type de `Bonjour` et de `Bonjour` encadré de ces délimiteurs.
4. Afficher « `La valeur de Bonjour est ...` », les ... étant remplacés par la valeur de la variable Bonjour.
5. Afficher les deux textes « `Aujourd'hui` » et « `Il dit: "Bonjour"` ». Afficher « `"Aujourd'hui"` ».
6. À quoi correspond le caratère spécial `\n` ?
7. En vous servant du paramètre sep et d'un autre paramètre que vous trouverez grâce à l'aide associée à `print`, après avoir défini 3 variables x, y et z, afficher, en une seule instruction :
   • x, y et z en ligne, séparés d'un point-virgule.
   • x, y et z en ligne, séparés d'un « et »
   • x, y et z en ligne terminant par un point.

- x, y et z séparés d'un point-virgule et d'un retour à la ligne et finissant par un « CQFD ».
  On verra un peu plus loin quelques autres manipulations concernant l'affichage.

Exercice 4 – Lecture de données rentrées par l'utilisateur

1. Quel est l'effet de l'instruction : `texte = input('Entrez un texte:')` ? Afficher le texte entré par l'utilisateur.
2. Définir de même une variable x en demandant une valeur numérique à l'utilisateur. Essayer de calculer $x + 3$. Constatation ? Type de x ?
3. *La fonction* `eval` *permet d'évaluer l'expression représentée par une chaîne de caractère.*
   Utiliser cette fonction pour définir à partir de l'entrée de l'utilisateur, une variable x de type numérique.

Exercice 5 – Comment importer un module

1. Essayer de calculer sin(1).
   Certains outils sont disponibles dans des « modules » spécialisés. C'est le cas d'un certain nombre de fonctions mathématiques usuelles, disponibles dans le module `math`. *Il y a 3 façons de faire appel à une fonction définie dans un module. Nous les étudions, de la moins coûteuse à la plus coûteuse.*
2. Pour pouvoir faire appel à des fonctions d'un module par exemple `math`, on utilise l'instruction `input math`. On peut alors utiliser la fonction sinus par exemple en écrivant `math.sin(x)`.
   - Déterminer sin(1)
   - Calculer $\sqrt{1 + \ln^2(2)}$ (utilisez `help` pour lister les fonctions disponibles dans le module `math`).
3. Si on fait appel un grand nombre de fois à une fonction précise, on peut l'importer pour de bon, ce qui évite par la suite de devoir faire référence au module `math` à chaque utilisation. Cela se fait (pour le sinus) avec l'instruction `from math import sin`
   - Calculer $\sin(1) + \sin(2) + \sin(3) + \sin(4) + \sin(5) + \sin(6)$.
   - Calculer $\sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{2}}}}}}$
4. Enfin, si on fait appel à un grand nombre de fonctions du module, on peut les importer toutes, en écrivant `from math import *`
   - Trouver expérimentalement une relation entre la fonction `gamma` et la factorielle.
   - Quelle différence fondamentale voyez-vous entre ces deux fonctions ?

Exercice 6 – Premières fonctions

Une fonction est un moyen de donner un nom à une succession d'instructions, dépendant en général d'un ou plusieurs paramètres ; une fonction retourne en général un résultat (de type non imposé). La syntaxe est :

```
def nom_de_la_fonction(parametre1,parametre2,...):
    instruction1
    instruction2
```

N'oubliez pas l'indentation (4 caractères). Cette indentation est importante : elle sert à délimiter les blocs d'instructions. Pour signaler la fin du bloc, ne plus indenter !

1. Écrire une fonction f qui fait afficher la valeur de $1 + x^n$, pour des arguments x et n. Le résultat devra s'afficher par exemple sous la forme $1 + 4\text{^}3 = 65$.
2. Peut-on calculer $f(3,3) + f(4,4)$? Quel est le type de f ? de $f(3,3)$?

Le problème vient du fait qu'on n'a pas défini de valeur de sortie. Pour le faire, il faut rajouter en dernière ligne de la fonction :

```
return valeur_à_retourner
```

1. Corriger la fonction précédente (on supprimera au passage l'affichage). Quel est maintenant le type de $f(3,3)$. Faire calculer $f(3,3) + f(4,4)$.

Exercice 7 – Premiers programmes

Nous quittons maintenant la console, pour écrire notre premier programme. Il s'agit donc d'écrire dans un fichier une succession d'instructions qui ne seront effectuées que lorsque nous lancerons l'exécution du programme.

1. Écrire un programme affichant `Bonjour`. Pour lancer l'exécution de votre programme, utiliser le menu Run.
2. Écrire un programme définissant la fonction $f : x \mapsto \frac{1}{1+x^2}$, demandant à l'utilisateur une valeur de x, et affichant la valeur de $f(x)$ (sous la forme d'une égalité).

Exercice 8 – Structures conditionnelles simples

Il est fréquent de devoir différencier l'action à effectuer suivant les cas. On utilise pour cette situation la structure conditionnelle

```
if bool:
    instructions
else:
    instructions
```

Le booléen `bool` est le plus souvent obtenu sous forme d'un test (`==`, `!=`, `>`, `>=`, `<`, `<=`, `is`, `in`*).*

1. Définir dans un programme (fenêtre du bas) la fonction :

$$\forall x \in \mathbb{R}, \quad f(x) = \begin{cases} \sqrt{1+x} & \text{si } x \geqslant -1 \\ 0 & \text{sinon.} \end{cases}$$

   Demandez ensuite à l'utilisateur une valeur de x et afficher $f(x)$.

Si la discussion porte sur plus de deux termes, on peut ajouter des tests intermédiaires grâce à `elif` *(abréviation de else if). Nous vous laissons deviner la syntaxe.*

1. Dans le même programme que ci-dessus, au lieu d'afficher $f(x)$, proposez à l'utilisateur, sous forme d'un menu à choix multiples, de calculer $f(x)$, $f(-x)$, $f(x^2)$ ou $f(-x^2)$, et effectuez l'opération correspondant au choix de l'utilisateur (avec l'affichage d'un message d'erreur en cas de choix incohérent).
2. Écrire dans un programme une fonction prenant en paramètre une année, renvoyant un booléen, égal à `True` si et seulement si l'année est bissextile. On demandera une année à l'utilisateur, et on lui affichera en réponse un texte disant si l'année est bissextile ou non.

   On rappelle que depuis octobre 1582, une année n est bissextile si et seulement si n est divisible par 4, sauf si n est divisible par 100, mais pas par 400. On rappelle également qu'avant 1582, les années bissextiles étaient exactement les années multiples de 4.

Exercice 9 – Structures itératives conditionnelles

Les structures itératives (boucles) permettent de répéter un bloc d'instructions un grand nombre de fois. Nous n'étudions pour le moment que les boucles dont l'arrêt est conditionné par une condition, ou plutôt dont l'arrêt est conditionné par la non réalisation d'une certaine condition. Il s'agit de la boucle :

```
while bool:
    instructions
```

1. Soit pour tout $n \in \mathbb{N}^*$, $S_n = \sum_{k=1}^{n} \frac{1}{n}$. Écrire un programme déterminant la plus petite valeur de n pour laquelle $S_n > A$, A étant un réel entré par l'utilisateur. N'essayez pas votre programme avec des valeurs de A supérieures à 22.
2. On rappelle que si a et b sont deux entiers strictement positifs si r le reste de la division euclidienne de a par b, alors, si $r \neq 0$, le pgcd de a et b est égal au pgcd de b et r. En répétant cette opération jusqu'à obtenir un reste nul, on peut donc calculer le pgcd (c'est l'algorithme d'Euclide).

   Écrire un programme demandant à l'utilisateur deux entiers a et b, puis calculer et afficher leur pgcd. Qu'en est-il du ppcm ?

Exercice 10 – Formater l'affichage des nombres.

Il est parfois utile de créer à partir d'une valeur numérique une chaîne de caractères, pour pouvoir l'insérer par exemple dans une chaîne de caractères plus grande. Cela peut être utile aussi bien pour un affichage à l'écran mieux maîtrisé que pour la création d'une chaîne de caractères en vue d'un traitement ultérieur.

- *L'insertion d'une valeur numérique x dans une chaîne de caractères se fait à l'aide de la « méthode »* `format` *de la classe «* `str` *». La syntaxe pour appliquer une méthode* `meth` *à un objet* `Obj` *est* `Obj.meth(paramètres)`.

- *La méthode* `format` *remplace dans une chaîne de caractères les symboles* `'{}'` *par le paramètre précisé. Essayez par exemple* `print('Les {} petits cochons'.format(3))`
- *On peut remplacer plusieurs accolades par plusieurs valeurs simultanément. les accolades sont remplacées par les valeurs successives dans l'ordre.*
- *Pour modifier l'ordre, ou pour utiliser à plusieurs reprises la même valeur, on peut numéroter les accolades en partant de* 0*, de la façon suivante :* `{:0}`, `{:1}` *etc. Si deux accolades sont numérotées de la même manière, elles utilisent le même paramètre de la méthode* `format`
- *Il existe différentes options d'affichage des nombres :*
  - `{:g}` *: choisit le format le plus adapté*
  - `{:.4f}` *: Écriture en virgule flottante, fixe le nombre de décimales, ici 4.*
  - `{:.5e}` *: Écriture en notation scientifique, fixe le nombre de décimales, ici 5.*
  - `{:<15.2e}` *: Fixe la longueur de la chaîne (elle est remplie par des espaces), et justifie à gauche. Le* `2e` *a la même signification que plus haut.*
  - `{:>15.2e}` *: Fixe la longueur de la chaîne, et justifie à droite.*
  - `{:^15.2e}` *: Fixe la longueur de la chaîne, centre.*

1. (Dans l'interpréteur) Donner la valeur $2/7$ à une variable x, et affichez « `Mon nombre vaut x` », la lettre x étant remplacée par sa valeur. Essayez les différents formats d'affichage.
2. Écrire un programme affichant toutes les valeurs de la suite u_n définie par la récurrence $u_0 = 1$ et pour tout $n \geqslant 0$, $u_{n+1} = \sin(u_n)$ jusqu'à obtenir une valeur inférieure ou égale à 10^{-20}. L'affichage devra être fait sous le format « u_{n} = $\{u_n\}$ » où $\{n\}$ et $\{u_n\}$ sont remplacées par leur valeur.
3. Même question avec les suites

$$u_0 = 5, \quad v_0 = 15, \quad \forall n \in \mathbb{N}, \quad \begin{cases} u_{n+1} &= \frac{1}{3}(2u_n + v_n) \\ v_{n+1} &= \frac{1}{3}(u_n + 2v_n), \end{cases}$$

   en s'arrêtant dès que $|v_n - u_n| < 10^{-10}$, l'affichage de u_n et v_n se faisant sous le même format que ci-dessus, les valeurs de u_n et v_n étant affichées sur une même ligne.
4. Essayer différents formats d'affichage pour l'exemple ci-dessus.
5. Écrire un programme affichant en 3 colonnes les premières puissances de 2, tant qu'elles sont inférieures à 10^{20}, en alignant les unités. On rappelle que deux chaînes de caractères peuvent être concaténées en utilisant l'opération $+$.

Exercice 11 – En admettant l'existence de cette limite, calculer une valeur approchée à 10^{-8} près de

$$\lim_{n\to+\infty} \sum_{k=2}^{n} \frac{(-1)^k}{k\ln(k)}.$$

Exercice 12 – Écrire un programme de résolution des équations de degré 2 à coefficients réels (on exprimera les racines complexes).

Exercice 13 – On définit la suite de Syracuse par $u_0 \in \mathbb{N}^*$, et

$$u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

On veut vérifier la propriété suivante : il existe un rang N tel que $u_N = 1$ (et à partir de ce rang, la suite boucle sur la séquence 4, 2, 1, 4, 2, 1, etc.). Écrire un programme demandant à l'utilisateur une valeur initiale u_0, calculant les différents termes de la suite tant qu'ils ne sont pas égaux à 1, et affichant pour terminer la première valeur de N pour laquelle $u_N = 1$, ainsi que la plus grande valeur obtenue pour u_n.
Cette propriété, à l'énoncé pourtant très simple, est encore aujourd'hui une conjecture, malgré les efforts acharnés de nombreux mathématiciens.