Informatique 1ere année, CPBX, TD machine 1

Carole Blanc, Paul Dorbec

Nous vous rappelons que vous trouverez le site avec le matériel du cours à l'adresse `http://dept-info.labri.fr/~blanc/ENS/CPBx/`. En particulier, vous y trouverez les guides de prise en main de `idle3` et de Python Tutor.

1 Découverte de l'environnement de travail

L'environnement d'édition que nous utilisons est `idle3`, qui vous permettra d'écrire des programmes en python, de les sauvegarder (contrairement à Python Tutor) et des les exécuter dans un interpreteur. Pour organiser votre espace de travail, créez un répertoire "`Informatique`" sur votre compte dans lequel vous enregistrerez les fichiers utilisés en TP pendant l'année. Vous pouvez utiliser l'interface graphique ou les commandes shell, si vous les connaissez.

Démarrez `idle3` depuis le menu `applications>developpement` ou en tapant idle3 dans le terminal (attention, le 3 est important pour utiliser la dernière version de Python). Une fenêtre s'ouvre avec la version de Python utilisée et un *prompt* "`>>>`". C'est la fenêtre de **l'interpréteur**.

Voici un premier programme en Python, pour faire quelques tests.

```
print("Bonjour à tous")
print("Bienvenue dans Python")
```

Question 1.1 *Ouvrez un nouveau fichier (*`file>open`*) et enregistrez le sous le nom* `bonjour.py`*. Notez la nouvelle fenêtre qui s'ouvre, appelée* ***fenêtre d'édition****.*
Tapez les deux instructions précédentes et sauvegardez le ficher.
Testez-le en pressant `F5`*.*
Dans la fenêtre d'édition, refaites `F5`*. Observez le résultat.*

Question 1.2 *Dans la fenêtre d'édition, insérez une ligne* `print()` *entre les deux lignes. Testez avec* `F5`*. Que se passe-t-il ?*

Question 1.3 *Ajoutez un* `#` *au début de la ligne* `print("Bienvenue dans Python")`. *Que se passe-t-il ?*

Le `#` au début d'une ligne sert à mettre un commentaire (une portion de code non éxécutée lors du programme). Vous verrez que ceci peut être très utile pour débugger un programme. Vous pouvez aussi commenter plusieurs lignes en tapant `'''` au début et à la fin de la zone que vous voulez commenter.

Question 1.4 *Décommentez la 3ème ligne du programme, insérez un (ou plusieurs) espace(s) au début de la troisième ligne, tapez* `F5`*, que se passe-t-il ? Pourquoi ?*

Prenez l'habitude quand l'interpréteur vous affiche un message d'erreur de le lire soigneusement et d'essayer de le comprendre. Cela vous aidera à reconnaître et à corriger les erreurs dans les programmes plus compliqués que vous écrirez bientôt.

2 Fonctions

2.1 Premiers pas

Voici un exemple de définition de fonction :

```
def f(x):
    y = x*x + x + 1
    return y
```

Question 2.1 *Ouvrez un nouveau fichier enregistrez le sous le nom* `premiereFonction.py`. *Dans ce fichier programmez la fonction* `f` *comme ci-dessus. Sauvegardez et interprétez le fichier avec* `F5`.

Dans l'interpréteur[1]*, tapez les instructions suivantes qui appellent cette fonction en lui passant différents arguments et prenez le temps d'expliquer en détail les résultats obtenus.*

```
y = f(2)
print(y)
t = 4
z = f(t)
```

1. Soyez attentifs à taper vos instructions dans la bonne fenêtre ! De manière générale, on écrit les définitions de fonctions dans la fenêtre d'édition, et on les appelle pour les tester dans l'interpréteur.

```
print(z)
x = (3*t + 1)/2
f(x)
f((3*t + 1)/2)
f(f(x))
t = f(t)
```

Question 2.2 *Voici une réécriture de la fonction précédente. Testez la avec les tests précédents et expliquez ce que vous obtenez.*

```
def f(x):
    y = (x*x + x + 1)
    print(y)
```

Vous noterez la différence fondamentale entre `print` et `return`. Bien que dans l'interpreteur, entrer une expression revient à l'afficher, leur rôle dans une fonction est très différent.

Question 2.3 *Encore une variation de la fonction* `f` *:*

```
def f(x):
    y = x*x + x + 1
    return y
    print("done")
```

Qu'observez vous ? Expliquez

2.2 Et c'est parti !

Question 2.4 *Écrivez une fonction* `moyenne(a,b)` *qui retourne la moyenne de deux nombres* `a` *et* `b`*. Testez-la avec les arguments* 42 *et* 23*.*

Question 2.5 *Écrivez une fonction* `moyennePondere(a, coef_a, b,coef_b)` *qui retourne la moyennne pondérée par le coefficient* `coef_a` *pour la note* `a` *et par le coefficient* `coef_b` *pour la note* `b`*. Testez-la en appelant* `moyennePondere(5,2,12,3)`*.*

Question 2.6 *Utilisez votre fonction* `moyennePondere` *pour écrire une autre version de la fonction* `moyenne(a,b)`*. Testez-la.*

Question 2.7 *Écrivez une fonction* `pair(n)` *qui teste si* `n` *est pair ; la valeur calculée doit être* booléenne*, c'est-à-dire égale à* `True` *ou* `False`*.*

3 Conditionnelles

La syntaxe `if ... else ...` permet de tester des conditions pour exécuter ou non certaines instructions. Exemple :

```
def testPrix(n, age):
    prix = 15 * n + 10
    if age <= 18:
        prix = 10 * n + 5
    return prix
```

```
def testPrixBis(n, age):
    if age <= 18:
        prix = 10 * n + 5
    else:
        prix = 15 * n + 10
    return prix
```

Question 3.1 *Écrire et tester une fonction* `compare(a,b)` *qui retourne* -1 *si* $a < b$, 0 *si* $a = b$, *et* 1 *si* $a > b$.

Question 3.2 *Écrire une fonction* `max2(x,y)` *qui calcule le maximum de deux nombres* x *et* y. *Attention : bien appeler cette fonction* `max2`, *car la fonction* `max` *est prédéfinie en python.*

Question 3.3 *Écrire une fonction* `max3(x,y,z)` *qui calcule le maximum de trois nombres* `x`, `y`, `z`. *Donner plusieurs versions de cette fonction dont une utilise la fonction* `max2`.

4 Suppléments

Question 4.1 *Le service de reprographie propose les photocopies avec le tarif suivant : les 10 premières coûtent 20 centimes l'unité, les 20 suivantes coûtent 15 centimes l'unité et au-delà de 30 le coût est de 10 centimes. Écrire une fonction* `coutPhotocopies(n)` *qui calcule le prix à payer pour* `n` *photocopies.*

Question 4.2 *Écrire une fonction* `uneMinuteEnPlus` *qui calcule l'heure une minute après celle passée en paramètre sous forme de deux entiers que l'on suppose cohérents. Exemples :*

- `uneMinuteEnPlus(14,32)` *vaut (14, 33).*
- `uneMinuteEnPlus(14,59)` *vaut (15, 0).*

Ne pas oublier le cas de minuit.

Question 4.3 *Écrire une fonction* `solve(a,b,c)` *qui prend en entrée les valeurs a,b et c et affiche le nombre de solutions réelles de l'équation* $ax^2 + bx + c = 0$ *puis ces solutions. Pour calculer la racine carrée de* x, *on écrira* `(x**0.5)`.
On pourra obtenir par exemple :

```
>>> solve(4,-4,1)
Une seule solution réelle:
0.5
>>> solve(1,5,5)
Deux solutions réelles:
-3.618033988749895
-1.381966011250105
```