L1-MPCE2I L1-BBTE L1-ECO-MATHS	Informatique - Travaux Pratiques	TP2

Objectifs du TP

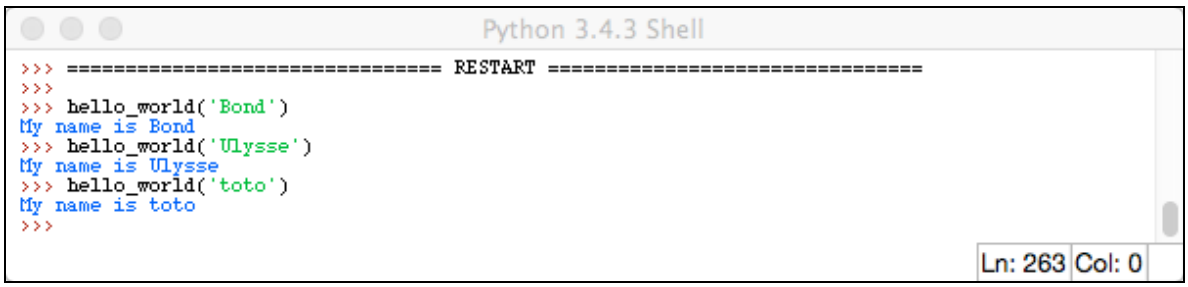
- Les variables, les expressions, les entrées-sorties et les conditions (oui encore !)
- Les exceptions
- Les fonctions

Exercice 1 : définir et exécuter une fonction

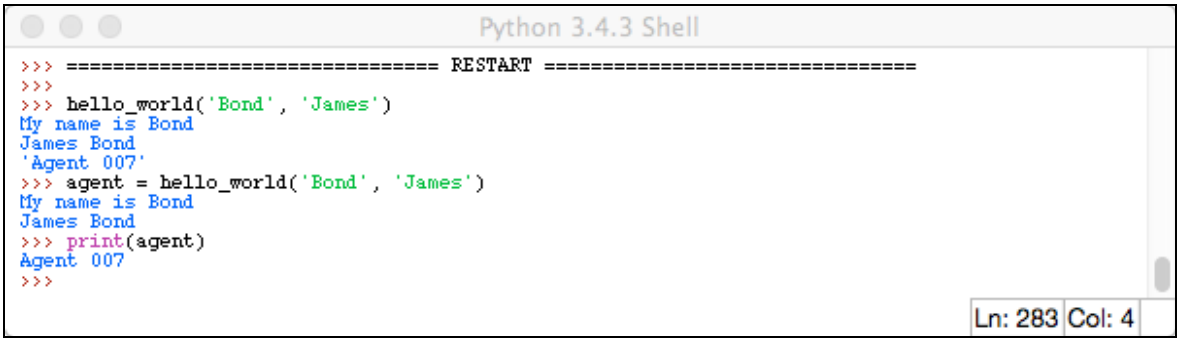
Lancez le logiciel IDLE puis, dans une fenêtre d'édition, saisissez le programme suivant :

```
# -*- coding: utf-8 -*-

def hello_world(name):
    print('My name is', name)
```

Chargez le fichier dans l'interpréteur avec le raccourci « *F5* » ou via le menu « *Run / Run module* » de la fenêtre d'édition, puis testez la fonction dans l'interpréteur comme illustré dans l'image suivante :

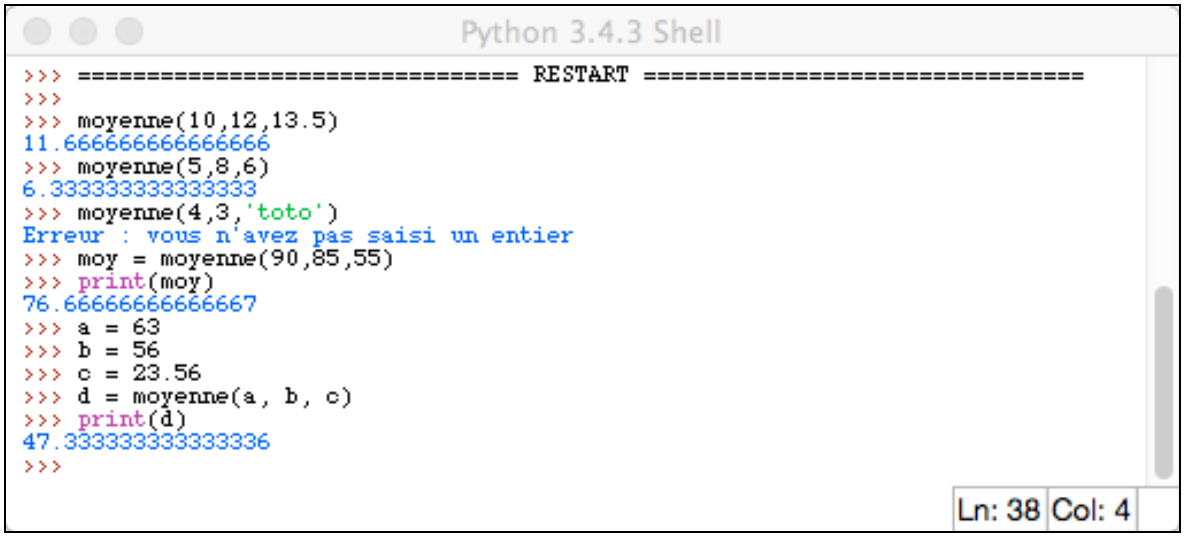


Modifiez la fonction pour qu'elle corresponde au fonctionnement attendu et illustré dans l'image suivante :



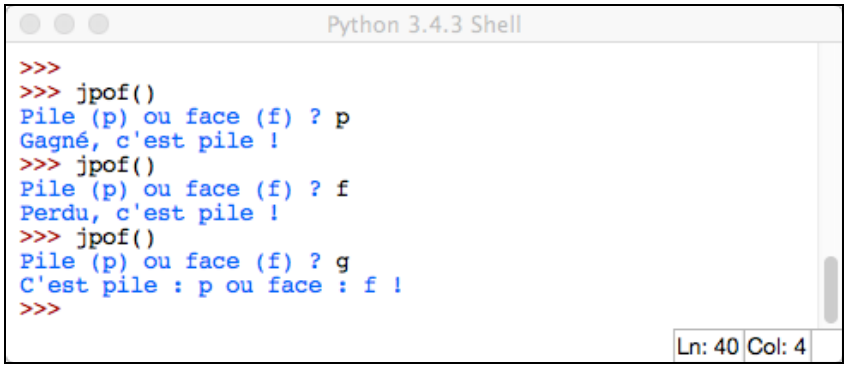
Exercice 2 : ma première fonction

Dans une fenêtre d'édition, écrivez une fonction nommée *moyenne(...)* qui calcul la moyenne de trois valeurs entières. La fonction devra gérer le cas où un (ou plusieurs) des arguments n'est pas convertible en entier comme illustré par l'image suivante :



Exercice 3 : joue avec l'ordinateur

Lancez le logiciel IDLE puis, dans une fenêtre d'édition, écrivez une version par ordinateur du fameux jeu « Pile ou face ? » dans lequel l'ordinateur tire au hasard et l'utilisateur choisit un côté de la pièce « virtuelle ». La fonction appelée *jpof()* devra fonctionner comme dans l'exemple de l'image suivante :



La fonction devra contrôler la saisie en vérifiant la validité ('*p*' ou '*f*' uniquement).

Remarque : vous utiliserez la fonction *randint(a, b)* de la librairie *random* qui retourne une valeur aléatoire comprise dans l'intervalle [*a*, *b*] (*a* et *b* étant des entiers).

Exercice 4 : une fonction qui appel une fonction qui appel des fonctions qui…

Le but de l'exercice est d'écrire un ensemble de fonctions dans le but d'établir un programme de calcul de surface géométrique plane. Une fonction principale fera appel à une autre fonction qui elle-même fera appel à d'autres fonctions et ainsi de suite.

La structure d'appel est la suivante :

```
programme_principal() ---> calcul(forme) -+--> dessine(forme) -+--> dessine_carre()
                                          |                    |
                                          |                    +--> dessine_rectangle()
                                          |                    |
                                          |                    +-> ...
                                          |
                                          +--> calcul_carre(cote)
```

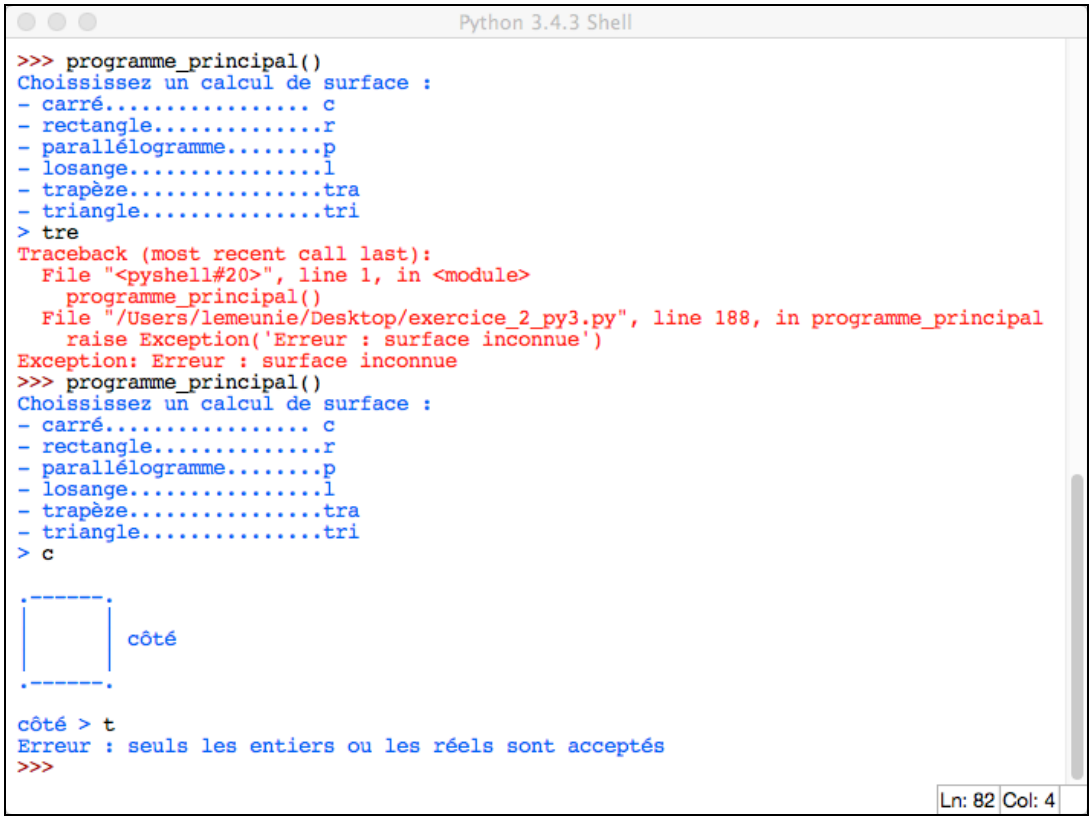
```
                                        |
                                        +--> calcul_rectangle(cote1, cote2)
                                        |
                                        +--> calcul_parallelogramme(base, hauteur)
                                        |
                                        +--> ...
```

Les fonctions sont définies de la manière suivante :

```
programme_principal() : affiche un menu, vérifie la saisie puis lance le calcul

calcul(forme) : affiche la forme puis délègue le calcul de la surface à une autre fonction
après saisie des informations nécessaires pour ce calcul

dessine(forme) : affiche la forme donnée en argument

calcul_carre(cote) : calcul la surface d'un carré et retourne la valeur

dessine_carre() : affiche un carré

calcul_rectangle(cote1, cote2) :  calcul la surface d'un rectangle et retourne la valeur

dessine_rectangle() : affiche un rectangle

etc.
```

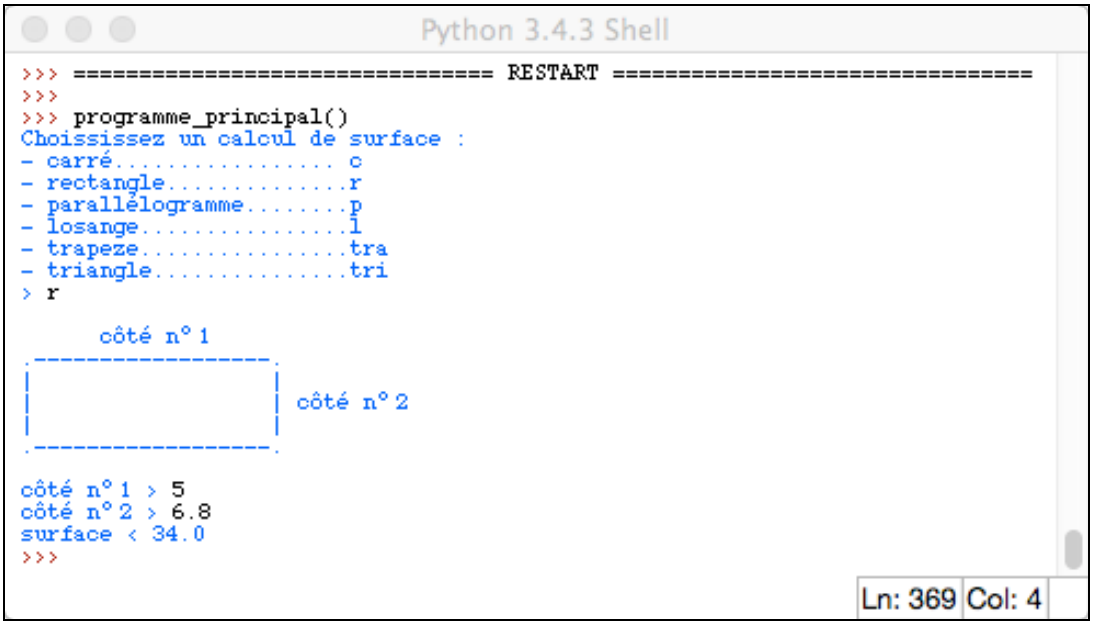
On se limitera dans un premier temps aux trois surfaces suivantes : carré, rectangle et trapèze. Si vous avez du temps en fin de TP, vous pourrez ajouter d'autres surfaces : parallélogramme, losange, triangle, cercle et ellipse.

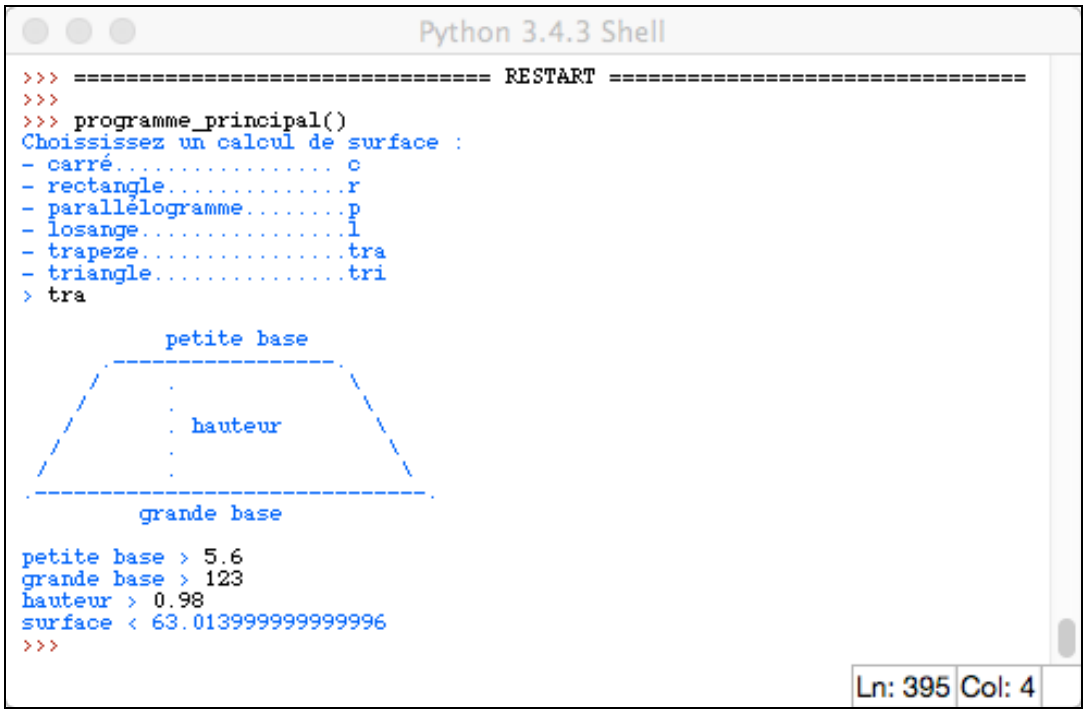
Le programme devra fonctionner comme dans les exemples des images suivantes :



Dans l'image précédente on voit que l'utilisateur doit saisir une surface connue sinon une exception est levée et le programme s'arrête. Il faut aussi contrôler la saisie des dimensions qui doivent être des entiers ou des réels.

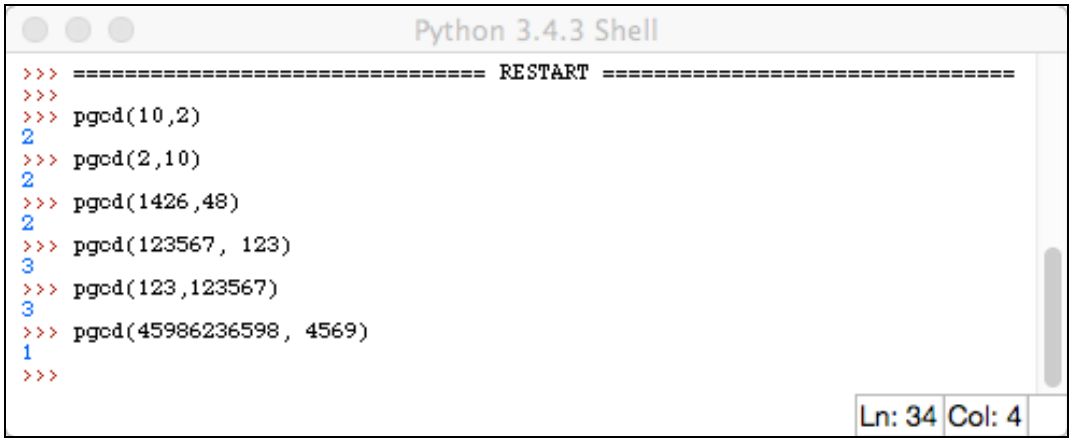
Dans les deux images suivantes, l'utilisateur demande successivement le calcul de la surface d'un rectangle et le calcul de la surface d'un trapèze :





Exercice 5 : calcul récursif du PGCD

Dans une nouvelle fenêtre d'édition, écrivez une fonction récursive calculant le PGCD de deux entiers. La fonction devra fonctionner comme illustré par l'image suivante :



Rappel : le plus grand diviseur commun de deux entiers naturels *a* et *b* est noté *pgcd(a, b)*. Le *pgcd(a,b)* divise *a* et divise *b*. Le pgcd est défini mathématiquement ainsi :

Soit $a = bq + r$ (r est le reste de la division euclidienne tel que $r = a \bmod(b)$)

$pgcd(a, 0) = a$ et $pgcd(a, b) = pgcd(b, r)$

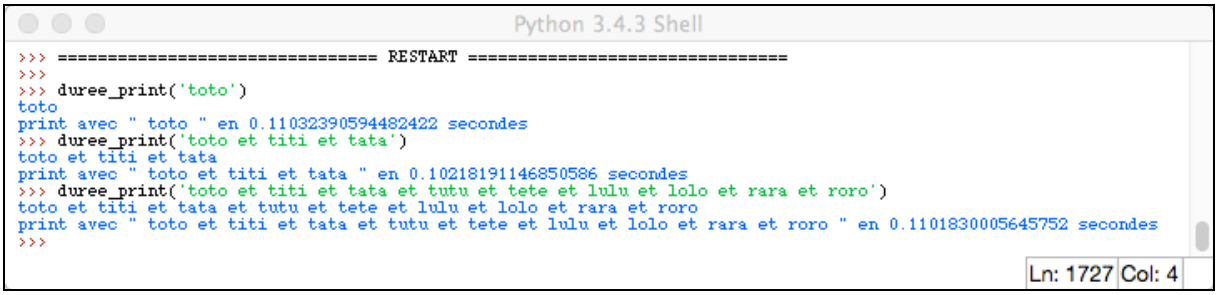
Exercice 6 : temps d'exécution et fonction récursive de la suite de Fibonacci

a) Mesurer le temps d'exécution d'une fonction ou d'un programme est quelque fois important car il permet de comparer différentes versions de codage d'un même algorithme ou simplement de connaître le temps d'attente possible lors de l'exécution d'une fonction.

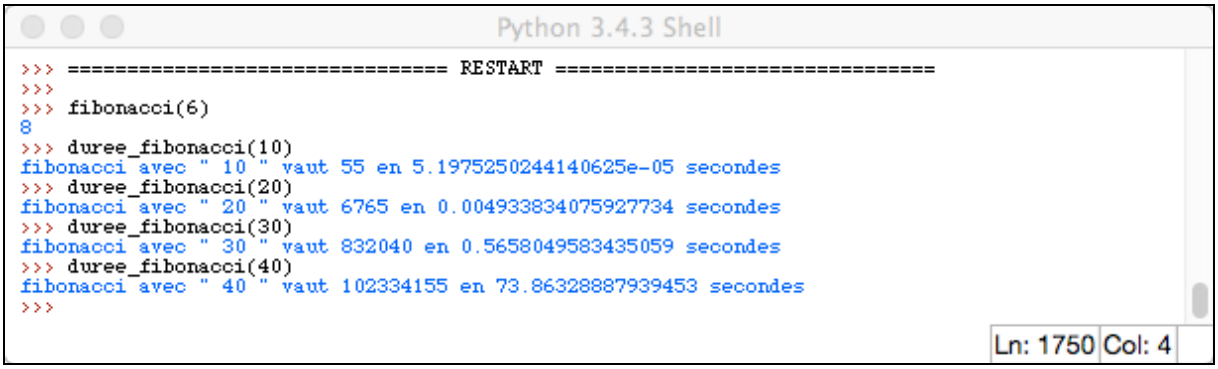
Dans une fenêtre d'édition, saisissez le code suivant :

```
# coding: utf-8

from time import *

def duree_print(chaine):
    debut = time()
    print(chaine)
    fin = time()
    print('print avec "', chaine, '" en', fin - debut, 'secondes')
```

Testez la fonction *duree_print(...)* avec différents arguments :



b) Ecrivez une fonction récursive qui calcul le *n*ième terme de la suite de Fibonacci et étudiez son temps d'exécution. La fonction appelée *fibonacci(...)* et la fonction *duree_fibonacci(...)* devront fonctionner comme dans l'exemple de l'image suivante :



Rappel : la suite de Fibonacci est la suite d'entier définie de la manière suivante

$$F(0) = 0, F(1) = 0 \text{ et } F(n) = F(n-1) + F(n-2)$$

Attention : ce n'est pas la même fonction qu'en TD !