

# TD2 : Premiers programmes en python

MPSI1 - Lycée Thiers

2013/2014

## Exercice 1

Exercice 1 : Enoncé

Exercice 1 : Corrigé

## Exercice 2

Exercice 2 : Enoncé

Exercice 2 : Corrigé

## Exercice 3

Exercice 3 : Enoncé

Exercice 3 : Corrigé

## Exercice 4

Exercice 4 : Enoncé

Exercice 4 : Corrigé

## Exercice 5

Exercice 5 : Enoncé

Exercice 5 : Corrigé

## Exercice 6

Exercice 6 : Enoncé

Exercice 6 : Corrigé

# Exercice 1 : Enoncé

Ecrire une fonction `abs()` prenant en paramètre un nombre et qui retourne sa valeur absolue, sans utiliser le module `math`.

# Exercice 1 : Corrigé

Solution :

```
# Fonction valeur absolue
def abs(x) :
    if x >= 0 :
        return x      # résultat retourné lorsque x est positif
    else :
        return -x     # résultat retourné lorsque x est négatif
```

L'exécution donne :

```
>>> abs(-1)
1
>>> abs(3.5)
3.5
>>> abs(0)
0
```

l'exécution de `abs(x)` avec `x` de type ni `'int'` ni `'float'` produira une erreur...

## Exercice 2 : Énoncé

Écrire une fonction `fact()` prenant en paramètre un nombre entier positif et qui retourne sa factorielle (sans utiliser le module `math`).

On rappelle que :  $\forall n \in \mathbb{N}, n! = \prod_{k=2}^n k$  (remarquer que  $0! = 1! = 1$ ).

## Exercice 2 : Corrigé

Réponse :

```
# Fonction factorielle
def fact(x) :
    x = int(x)
    result = 1
    while (x > 1) :
        result = result * x
        x = x - 1
    return result
```

```
>>> fact(0)
1
>>> fact(3)
6
>>> fact(4)
24
>>> fact(5)
120
```

fact(x) avec x négatif ou de type non 'int' produira un résultat absurde...

## Exercice 3 : Énoncé

Écrire une fonction  $E()$  prenant en paramètre un nombre et qui retourne sa partie entière (sans utiliser le module `math`).

(la partie entière d'un réel  $x$ , notée  $\lfloor x \rfloor$ , est le plus grand entier inférieur ou égal à  $x$ ).

## Exercice 3 : Corrigé

Réponse :

```
# Fonction partie entière E()
def E(x) :
    if (x == int(x)) : return x
    elif (x >= 0) : return int(x)
    else : return -E(-x)-1
```

```
>>> E(3.14)
3
>>> E(-3.14)
-4
>>> E(0)
0
```



## Exercice 4 : Énoncé

Améliorer le programme permettant de donner l'écriture d'un nombre en base 16, pour que les chiffres soient 0-9, A, B, C, D, E, F et que l'écriture débute par le préfixe 0x.

## Exercice 4 : Corrigé

Réponse :

```
a = input('Saisissez un nombre ')
resultat = ''
while a>0 :
    b = a%16
    if b == 10 : b = 'A'
    elif b == 11 : b = 'B'
    elif b == 12 : b = 'C'
    elif b == 13 : b = 'D'
    elif b == 14 : b = 'E'
    elif b == 15 : b = 'F'
    resultat = str(b) + resultat
    a=a/16
resultat = '0x' + resultat
print resultat
```

```
Saisissez un nombre 15 * 16 ** 2 + 13 * 16 + 9
0xFD9
```

## Exercice 4 : Corrigé

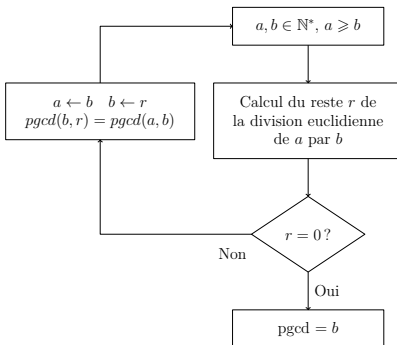
Algorithme un peu plus rapide :

```
a = input('Saisissez un nombre ')
resultat = ''
while a>0:
    b = a%16
    if b < 10: b = str(b)
    elif b == 10 : b = 'A'
    elif b == 11 : b = 'B'
    elif b == 12 : b = 'C'
    elif b == 13 : b = 'D'
    elif b == 14 : b = 'E'
    elif b == 15 : b = 'F'
    resultat = b + resultat
    a=a/16
resultat = '0x' + resultat
print resultat
```

```
Saisissez un nombre 15 * 16 ** 2 + 13 * 16 + 9
0xFD9
```

# Exercice 5 : Énoncé

Écrire un programme qui calcule le PGCD de deux nombres à l'aide de l'**algorithme d'Euclide**, que l'on commencera par justifier :



## Exercice 5 : Corrigé

Preuve. Tout d'abord lorsque  $r = 0$ ,  $a$  est un multiple de  $b$  et donc  $\text{pgcd}(a, b) = b$ . Lorsque  $r > 0$ . Puisque  $\exists q \in \mathbb{N}$ ,  $a = qb + r$ , tout diviseur de  $b$  et de  $r$  est aussi diviseur de  $a$  (ainsi que de  $b$ ). De même  $r = a - qb$  et donc tout diviseur de  $a$  et de  $b$  est aussi diviseur de  $r$  (ainsi que de  $b$ ). Ainsi  $a, b$  et  $b, r$  ont mêmes diviseurs communs ; en particulier :  $\text{pgcd}(a, b) = \text{pgcd}(b, r)$ .

L'algorithme se termine : la valeur de  $b$  est un entier naturel qui diminue strictement à chaque itération de la boucle tandis que le reste  $r$  est un entier naturel strictement inférieur à  $b$ , ainsi il y aura au plus  $b$  passages dans la boucle avant que le critère d'arrêt  $r = 0$  ne soit satisfait. D'après ce qui précède la valeur retournée est  $\text{pgcd}(a, b)$ .

```
# Algorithme d'Euclide pour le pgcd
def pgcd(a,b) :
    while a%b != 0:
        a, b = b, a%b
    return b
```

```
>>> pgcd(120,36)
12
>>> pgcd(2,3)
1
>>>
```

Si au départ  $a < b$  la division euclidienne de  $a$  par  $b$  aura pour reste  $a$  (et quotient 0) et après la première itération de la boucle `while` les valeurs de  $a$  et  $b$  auront été échangées, il est donc inutile de traiter séparément le cas  $a < b$ .

## Exercice 6 : Enoncé

Ecrire un programme pour l'arithmétique qui affiche le menu suivant et effectue les actions correspondantes :

MENU :

- 1 - Calculer le pgcd de 2 entiers
- 2 - Calculer le ppcm de 2 entiers
- 3 - Déterminer si un nombre est premier
- 4 - Donner la décomposition en facteurs premiers d'un nombre
- q - Quitter

Indication : l'instruction `break` permet la sortie d'une boucle.

## Exercice 6 : Corrigé

L'écriture du menu s'obtient par :

```
# Corps du programme
while True :
    print """MENU :
    1 - Calculer le pgcd de 2 nombres
    2 - Calculer le ppcm de 2 nombres
    3 - Déterminer si un nombre est premier
    4 - Donner la décomposition en facteur premier d'un nombre
    q - Quitter"""
    choix = raw input("\nVotre choix :")
    # Quitter
    if (choix == 'q' or choix == 'Q') : break
```

Une chaîne de caractère placée entre 3 guillemets `"""` permet le saut à la ligne.

```
# choix 1 : pgcd
elif (choix == '1') :
    print "\nSaisissez 2 entiers "
    entier1 = input()
    entier2 = input()
    print "le pgcd de",entier1,"et",entier2,"est",pgcd(entier1,entier2)
```

Le choix 1 appelle la fonction `pgcd()` déjà écrite (elle doit figurer avant le programme).

## Exercice 6 : Corrigé

Le choix 2 :

```
elif (choix == '2') :  
    print "\nSaisissez 2 entiers "  
    entier1 = input()  
    entier2 = input()  
    print "le ppcm de",entier1,"et",entier2,"est",ppcm(entier1,entier2)
```

appelle la fonction ppcm() :

```
def ppcm(a,b) :  
    return (a * b) / pgcd(a,b)
```

à faire figurer avant le programme.

Elle utilise  $a.b = \text{pgcd}(a, b) . \text{ppcm}(a, b)$ .



## Exercice 6 : Corrigé

Les deux derniers choix appellent des fonctions `premier()` et `Pdecomp()`, qu'il reste à écrire :

```
elif (choix == '3') :  
    entier = input("Saisissez un entier ")  
    if premier(entier) : print entier,"est premier"  
    else : print entier,"n'est pas premier"  
elif (choix == '4') :  
    entier = input("Saisissez un entier ")  
    print "La décomposition de",entier,"en premiers est",Pdecomp(entier)
```

La fonction `premier()` prend en argument un entier et retourne un booléen selon si l'argument est un nombre premier ou pas.

La fonction `Pdecomp()` prend en argument un entier et retourne une chaîne de caractère donnant la décomposition en nombres premiers de l'argument.

## Exercice 6 : Corrigé

La fonction `premier()` doit figurer avant le programme :

```
from math import sqrt

def premier(a) :
    result = True
    n = 2
    top = sqrt(a)
    while (n <= top) :
        if a % n == 0 : return False
        else : n += 1
    return True
```

Elle utilise la fonction `sqrt()` du module `math.py`, qu'il s'agit d'importer auparavant.

## Exercice 6 : Corrigé

La fonction `Pdecomp()` doit figurer avant le programme :

```
def Pdecomp(a) :
    result = ''
    n = 2
    ancienne = a
    premier = True
    while (n <= a) :
        if (a % n == 0) :
            premier = False
            if result == '' : result = str(n)
            else : result = result + '.' + str(n)
            a = a / n
        else : n = n + 1
    if premier : return str(ancienne)
    else : return result
```

## Exercice 6 : Corrigé

Exemple :

```
MENU :  
  1 - Calculer le pgcd de 2 nombres  
  2 - Calculer le ppcm de 2 nombres  
  3 - Déterminer si un nombre est premier  
  4 - Donner la décomposition en facteur premier d'un nombre  
  q - Quitter  
  
Votre choix :4  
Saisissez un entier 120  
La décomposition de 120 en premiers est 2.2.2.3.5
```

On peut encore pour améliorer le programme, gérer des exceptions lorsque l'utilisateur ne saisit pas un entier positif pour lui renvoyer un message d'erreur. C'est assez facile : le plus gros du travail est fait. Nous verrons ça plus tard.