Exercice 1

1) (a) $\overline{1\,1011}^{2} = 2^4 + 2^3 + 2^1 + 2^0 = 16 + 8 + 2 + 1 = \boxed{27}$

   (b) $\overline{123}^{8} = 8^2 + 2 \times 8^1 + 3 \times 8^0 = 64 + 16 + 3 = \boxed{83}$

   (c) $\overline{FA}^{16} = 15 \times 16^1 + 10 \times 16^0 = 240 + 10 = \boxed{250}$

2) Avec des divisions euclidiennes successives (ou bien une décomposition en sommes de puissances de la base), on arrive sans problème aux résultats suivants :

   (a) $75 = 64 + 8 + 2 + 1 = 2^6 + 2^3 + 2^1 + 2^0$ s'écrit ainsi $\boxed{\overline{100\,1011}^{2} \text{ en base 2.}}$

   (b) $75 = 64 + 8 + 3 = 8^2 + 8^1 + 3 \times 8^0$ s'écrit ainsi $\boxed{\overline{113}^{8} \text{ en base 8.}}$

   (c) $75 = 64 + 11 = 4 \times 16^1 + 11 \times 16^0$ s'écrit ainsi $\boxed{\overline{4B}^{16} \text{ en base 16.}}$

3) (a) Groupons les bits par paquets de 4 :

   - $\overline{1010}^{2}$ donne 8 + 2 = 10 soit A en base 16.
   - $\overline{11}^{2}$ donne 2 + 1 = 3.

   Ainsi, l'entier représenté par $\overline{11\,1010}^{2}$ s'écrit $\boxed{\overline{3A}^{16} \text{ en base 16.}}$

   (b) Là, c'est l'opération inverse :

   - $\overline{F}^{16}$ donne 15 soit $\overline{1111}^{2}$ en base 2.
   - $\overline{D}^{16}$ donne 13 soit $\overline{1101}^{2}$ en base 2.
   - $\overline{E}^{16}$ donne 14 soit $\overline{1110}^{2}$ en base 2.

   Ainsi, l'entier représenté par $\overline{EDF}^{16}$ s'écrit $\boxed{\overline{1110\,1101\,1111}^{2} \text{ en base 2.}}$

Exercice 2

1) Les seuls codes corrects sont le $\boxed{\text{code B}}$ et le $\boxed{\text{code C}}$.

   A) Toutes les variables reçoivent la valeur initiale de a, et la valeur de b est effacée.

   Voici en effet les contenus des variables a, b et c à chaque étape :

a	b	c
a	b	None
a	b	a
a	a	a
a	a	a

   B) C'est un code vu en cours, donc il est forcément correct.

   Pour ceux qui auraient le mauvais goût d'en douter, voilà la preuve :

a	b	c
a	b	None
a	b	a
b	b	a
b	a	a

C) Ce code est peut-être un peu tordu, mais il fonctionne quand même.

En effet :

a	b
a	b
a+b	b
a+b	a
b	a

D) On a repris le code précédent en modifiant la dernière instruction. Du coup, ça ne marche plus !

La preuve :

a	b
a	b
a+b	b
a+b	a
2a+b	a

2) On a vu en cours que le plus simple est de taper le code suivant : `a,b = b,a`

Exercice 3

Dans les deux cas, comme la variable a vaut 0, on exécute les instructions après le `if` mais pas celles après `else`.

1) On exécute d'abord `b=4` puis `b=1`. Ainsi, à l'issue du programme, on a

a=0, b=1 et c=1

2) On exécute `b=4` et c'est tout. Ainsi, à l'issue du programme, on a

a=0, b=4 et c=1

Exercice 4

Voici le code attendu :

```
def amende(poule, chien, vache, ami, promeneur):
    return poule+3*chien+5*vache+10*ami+20*promeneur
```

Exercice 5

1) Voici la procédure demandée :

```
def Tableur(f,a,b,n):
    pas=(b-a)/(n-1)
    X=[a+k*pas for k in range(n)
    Y=[f(x) for x in X]
    return Y

# Variante avec numpy
X=numpy.linspace(a,b,n)
```

2) Et voilà le travail :

```
def f(x):
    return 2*x**3+x-5

print(Tableur(f,0,1,11))
```

Exercice 6

1) On utilise évidemment une liste en tant que variable locale.

2) On crée une liste initialement vide, puis on lui ajoute les résultats successivement rentrés à l'aide d'une boucle infinie. Cette boucle est interrompue lorsque l'on saisit 6.

```
def Resultats():
    Lancers=[]
    while 1>0:
        res=eval(input("Résultat : "))
        Lancers.append(res)
        if res==6:
            break
    return Lancers
```

3) L'algorithme le plus rapide consiste à parcourir la liste `Lancers` et à incrémenter un des termes de la liste `[N1,N2,N3,N4,N5]` selon la valeur de l'élément ... cette astuce qui permet d'éviter d'écrire 5 tests.

On peut aussi utiliser un dictionnaire pour faire à peu près la même chose ... sachant qu'il aurait mieux valu le créer et le mettre à jour lors de la saisie des résultats.

```
def Effectifs(Lancers):
    # On vire d'abord le 6
    Lancers.pop()
    # On initialise les effectifs
    effs=[0,0,0,0,0]
    # On parcourt la liste Lancers
    for i in Lancers:
        effs[i-1]=effs[i-1]+1
    return effectifs
```

```
def Effectifs(Lancers):
    dico={}
    for i in Lancers:
        if i in dico:
            dico[i]=dico[i]+1
        else:
            dico[i]=1
    L=[dico[i] for i in range(1,6)]
    return L
```

Beaucoup plus simple mais moins efficace : on utilise deux boucles imbriquées pour comptabiliser les 1, puis les 2, les 3, etc...

```
def Effectifs(Lancers):
    Liste=[]
    for i in (1,2,3,4,5):
        N=0
        for res in Lancers:
            if res==i:
                N=N+1
        Liste.append(N)
    return Liste
```

4) Pas de difficulté, c'est un bête calcul de somme.

```
def Voyance(Lancers):
    somme=0
    for res in Lancers:
        somme=somme+res
    return somme<30
```