Exercice 1

1) Un code à 3 bits comporte $2^3 = 8$ combinaisons différentes.
Donc il ne peut pas produire plus de huit couleurs différentes.

2) le complément du noir 000 est 111 donc blanc.
le complément du bleu 001 est 110 donc jaune.
le complément du vert 010 est 101 donc magenta
le complément du cyan 011 est 100 donc rouge
le complément du rouge 100 est 011 donc cyan
le complément du magenta 101 est 010 donc vert
le complément du jaune 110 est 001 donc bleu
le complément du blanc 111 est 000 donc noir.

3) a) bleu = 0b001
rouge = 0b100

bleu | rouge vaut 0b101 donc magenta.

b) magenta = 0b101
cyan = 0b011

magenta & cyan vaut 0b001 donc bleu.

c) vert = 0b010
blanc = 0b111

vert ^ blanc vaut 0b101 donc magenta.

Exercice 2

1) La variable message est de type tuple.

2) George Boole

3) a) si on décode message_2 selon la table Ascii on obtient:

cod... en UTF 8
deux caractères absents de la table Ascii

Donc "codé en UTF 8" la séquence qui représente le "é" est constituée des deux caractères absents de la table Ascii c'est à dire C3 A9

b). Si message_2 avait été interprété en latin-1, nous aurions vu s'afficher:

codÃ© en UTF8

Exercice 3

1) a) carre4 est une liste de listes.

b) len(carre4) est le nombre d'éléments. Il y en a 4.
(il y a 4 listes dans la liste de listes).

c) carre3[1] est la 2e liste de carre3 donc [9, 5, 1].

d) carre3[0][2] est le 3e nombre de la 1ère liste de carre3 donc c'est 6.

e) 3 est le 2e nombre de la 3e liste de carre4 donc carre4[2][1] permet de récupérer 3.

2) a) la fonction somme_ligne renvoie la somme des nombres se trouvant sur la ligne i.
Donc somme_ligne(carre4, 2) renvoie la somme des nombres de la ligne d'indice 2 dans le carré magique d'ordre 4.
$6 + 3 + 13 + 12 = 34$

somme_ligne(carre4, 2) renvoie 34.

b) la fonction somme_ligne renvoie la somme des nombres sur la ligne d'indice i.

3)
```
def verifie_lignes(carre, n):
    somme = somme_ligne(carre, 0) # Initialisation avec la
                                  # somme sur la 1re ligne.
    for i in range(1, n):
        cette_somme = somme_ligne(carre, i)
        if cette_somme != somme:
            return False
    return True.
```

Exercice 4

Partie 1

1) Les descripteurs de la table sont 'nom_pays', 'nom_capitale', 'latitude', 'longitude', 'code_pays', 'continent'.

2) Il faut écrire l'instruction
table_capitales = lecture_fichier('capitales.csv')

3) table_capitales est du type liste de listes

4) La variable recherche_1 va contenir une liste de listes des capitales dont la latitude est négative.

5) Le critère de sélection doit être
lambda x: x[5] == 'South America'

Ainsi on doit saisir :
recherche_2 = selectionner(table_capitales, lambda x: x[5] == 'South America')

6) Le critère de sélection doit être
lambda x: x[5] == 'South America' and float(x[2]) > 0
Ainsi : recherche_3 = selectionner(table_capitales, lambda x: x[5] == 'South America' and float(x[2]) > 0)

Partie 2

1) les_descripteurs = table_sacs[0]

2) S1 contient la liste de listes des sacs triées par longueurs de sacs croissantes. La ligne des descripteurs est absente.

3) S2 contient la liste de listes des sacs triées par prix H.T de sacs décroissants. La ligne des descripteurs est absente.

4) ma_selection = S2[0:5]

En effet S2 ne contient pas la liste des descripteurs puisqu'elle provient de table_sacs[1:]

la ligne de rang 5 ne sera pas prise, ce qui fait bien 5 lignes dont les rangs sont 0, 1, 2, 3, 4.

5) a) S3 = sorted(table_sacs[1:], key=lambda ligne: ligne[1])

pour chaque ligne de la liste de listes table_sacs[1:] la fonction sorted se basera sur l'élément de rang 1 qui est le nom du sac.

b) [ligne for ligne in S3 if ligne[5] == 'noir']

condition de filtrage

Exercice 5

1) Oui les deux tables ont les mêmes descripteurs.

2) La façon de procéder de Betty est incorrecte pour deux raisons :
   - Les identifiants pour les enregistrements de la table 2 sont les mêmes que pour les enregistrements de la table 1.
   - Les températures n'ont pas les mêmes unités.

3) Première boucle for.
Elle réalise la conversion des températures de t2 de degrés F en degrés C.

```
for ligne in t2[1:]:
    ligne[1] = (ligne[1] - 32) * (5/9)
```

Remarque :
On peut améliorer en arrondissant à 1 décimale.
La boucle for à écrire est alors :

```
for ligne in t2[1:]:
    ligne[1] = round((ligne[1] - 32) * (5/9), 1)
```

Deuxième boucle for
Elle renumérote les id_esp de t2 en leur ajoutant le nombre de lignes de t1 (en ne comptant pas les descripteurs)

```
for ligne in t2[1:]:
    ligne[0] = ligne[0] + len(t1) - 1
```

len(t1) - 1 est le nombre de lignes dans la table t1 en ne comptant pas la ligne des descripteurs.