Notes de correction

Exercice 1

1.

	Initialisation	Etape 1	Etape 2	Etape 3	Etape 4	…
`i≤j`		V	V	V	F	
`mot[i] ≠ mot[j]`		F	F	F		
`I`	0	1	2	3		
`J`	4	3	2	1		
P	Vrai	V	V	V		

Renvoie Vrai

2a. il y a 3 comparaisons de caractères

2b. Si n est pair : n/2, sinon (n+1)/2

3. i augmente et j diminue à chaque tour de boucle, donc l'écart j-i diminue strictement tout en restant entier et positif (condition du while). Donc la boucle se termine.

4. Pour routeur, il y a 4 tests alors que 2 suffisent

	Initialisation	Etape 1	Etape 2	Etape 3	Etape 4	…
i≤j		V	V	V	V	F
mot[i] ≠ mot[j]		F	V	V	F	
I	0	1	2	3	4	
J	6	5	4	3	2	
P	Vrai	V	F	F	F	

Proposition d'amélioration : 2 possibilités : sortir dès que p est faux et/ou mettre une inégalité stricte dans le `while` :

```
def palindrome1(mot) :
    i, j, p = 0, len(mot)-1, True
    while i < j and p :
        if mot[i] != mot[j] :
            p = False
        i = i + 1
        j = j - 1
    return p
```

Exercice 2

1a.
id_plat : `INT`
num_plat, type_plat : `VARCHAR(100)`
prix_plat : `FLOAT` (ou les 2 autres types flottants)

1b. Clés primaires :
Plat : `id_plat`
Table_salle : `num_table`
Client : `num_client`
Reservation : `num_reservation`

1c. Clés étrangères : `num_table, num_client.` Cela permet, entre autre, de se mettre des « garde-fous » lors des modifications ou suppressions.

2a. `SELECT nom_plat, type_plat, prix_plat FROM plat`

2b. `SELECT nom_plat FROM plat WHERE type_plat = 'Dessert'`

2c. `UPDATE client SET tel_client = '0602030405' WHERE num_client = 42`

2d. `SELECT nom_client FROM client JOIN reservation ON reservation.num_client = client.num_client WHERE num_table = 13`

Exercice 3

1a. `cd ../projet`

1b. `cd /root/home/sam/projet`

2a. `cd projet` puis `ls` ou directement `ls projet`

2b. `chmod rwx projet/config.txt`

3a. Il vide en profondeur les fichiers et dossiers.

3b. il s'agit d'un parcours en profondeur : on vide les fichiers ou dossiers les plus profonds, puis on « remonte »

4. La fonction renvoie 1

Exercice 4

1. Une solution :

```
def ajouter_beurre(self, qt) :
    self.qt_beurre = self.qt_beurre + qt
```

2.

```
def afficher(self) :
    print("farine :", self.qt_farine)
    print("oeuf :", self.nb_oeufs)
    print("beurre :", self.qt_beurre)
```

3.

```
def stock_suffisant_brioche(self) :
    return self.qt_beurre >= 175 and self.qt_farine >= 350 and self.nb_oeufs >= 4
```

4a. Cela renvoie 2 , c'est le nombre de brioches qui peuvent être produites

4b.

```
Farine : 300
Œuf : 2
Beurre : 650
```

5. Une solution :

```
def nb_brioches(liste_stocks) :
    nb = 0
    for s in liste_stocks :
        nb = nb + s.produire()
    return nb
```

Exercice 5

1a. Renvoie [2,6]
1b. Renvoie les coordonnées du point d'arrivée
2. Une solution :

```
def accessible(dep, arrivee) :
    return mystere(dep) == arrivee
```

3.

```
1 | from random import randint
2 | def chemin(arrivee):
3 |   deplacement = '00000000'
4 |   while not accessible(deplacement, arrivee) :
5 |     deplacement = ""
6 |     for k in range(8):
7 |       pas = str(randint(0,1))
8 |       deplacement = deplacement + pas
9 |   return deplacement
```

4. La plus grande valeur possible est atteinte pour le chemin `"11100000"` soit 224