DS d'informatique n°4 (durée : 2h)

Exercice 1 : Chaines de caractères

1.
```
def contient(ch,c):
  for car in ch:
    if(car==c):
      return True
  return False
```

2.
```
def occurences(ch,c):
  total=0
  for car in ch:
    if(car==c):
      total = total+1
  return total
```

3. La boucle `for` de la fonction `occurences` s'effectue exactement `n` fois. La complexité de cette fonction est donc linéaire : $O(n)$.

4.
```
def listeOccurences(ch,cars):
  tots = []
  for car in cars:
    tots.append(occurences(ch,car))
  return tots
```

5. La boucle de la fonction `listeOcucrences` est effectuée exactement m fois et la complexité de la fonction `occurences` est n. La complexité de la fonction `listeOccurences` est donc $O(n \times m)$

6. La fonction `truc(chaine)` renvoie la lettre qui apparaît le plus de fois dans la chaîne `chaine`.
`truc("Bonjour tout le monde")` renvoie le caractère `'o'`.

7.
```
def position(ch,c):
  for i in range(len(ch)):
    if(ch[i]==c):
      return i
  return -1
```

8. On souhaite crypter un texte pour le rendre illisible par une personne ne possédant pas la clé de décryptage. Pour cela on utilise la fonction suivante :
```
def crypt(chaine,cle):
  lettres = "abcdefghijklmnopqrstuvwxyz"
  ch = chaine.lower()
  res = ""
  for c in ch:
    index = position(lettres,c)
    if index>=0:
      index = index+cle
      if index>len(lettres):
        index = index-len(lettres)
      res = res+lettres[index]
    else:
      res = res+c
  return res
```

Le paramètre `chaine` est de type chaîne de caractère et `cle` est un entier. La fonction `crypt` opère un décalage de chaque lettre de la chaîne `chaine` vers la droite de `cle` positions. Si le résultat du décalage est plus loin que le caractère 'z', on continue à compter à partir de 'a'. La fin de la question comporte évidemment une erreur, il manque la clé, il fallait lire par exemple : `crypt("message secret",8)` dont le résultat est : `'umaaiom amkzmb'`

9.

```
def decrypt(chaine,cle):
  lettres = "abcdefghijklmnopqrstuvwxyz"
  res = ""
  for c in ch:
    index = position(lettres,c)
    if index>=0:
      index = index-cle
      if index<0:
        index = index+len(lettres)
      res = res+lettres[index]
    else:
      res = res+c
  return res
```

10. Si le texte crypté est suffisamment long, il suffit de déterminer la lettre la plus fréquente du texte crypté. Comme on sait que cette lettre est le 'e', on peut connaître le décalage. Cette algorithme est évidemment très peu sécurisé, on peut l'améliorer en utilisant une clé de plusieurs décalages à utiliser successivement pour chaque lettre du message à coder. Si le message codé est assez long, ce deuxième algorithme est aussi facilement cassable (trouver comment on pourrait faire !).

Exercice 2 : Méthode d'Euler

On résout l'équation différentielle en executant l'instruction `tps,vals = euler(fct,0,10,2,0.01)`.

1. L'équation différentielle résolue est $y' + 4y = 5$
2. `t0` est le temps initial, `y0` est la valeur de $y(t_0)$, `tmax` est le temps maximum auquel on souhaite calculer la solution, `h` est le *pas* utilisé lors de la résolution. Pour que la solution soit précise il faut choisir h aussi petit que possible, le problème est que le temps de calcul est d'autant plus grand que h est petit.
3. `tps = [0,0.01,0.02,0.03,0.04,...]` et `vals = [10, 9.65, 9.31, 8.99, 8.68]`
4.

```
def f(y,t):
  return 2*y**2-3*y+2
```

5. Pour résoudre une équation différentielle d'ordre 2 du type $y'' = f(y', y, t)$, on la décompose en un système de deux équations d'ordre 1 couplées en introduisant $y_2 = y'$. On a alors les deux équations différentielles d'ordre 1 : $y_2' = f(y_2, y_1, t)$ et $y_1' = y_2$ que l'on peut résoudre avec la méthode d'Euler.

Exercice 3 : Intégration numérique

```
def integRectangles(f,a,b,n):
  dx = (b-a)/n
  S = 0
  for i in range(n):
    xi = a + dx*(i+1/2)
    S += dx*f(xi)
  return S
```