Travail sur les fichiers texte et les chaines de caractères

Séance 7

Remplir la fiche de cours à mesure que vous allez rencontrer les nouvelles fonctions. Vous indiquerez pour chaque fonction rencontrée l'utilité de la fonction et surtout un exemple permettant de mettre clairement en valeur la syntaxe à utiliser

Créer un programme affichant 8 lignes des caractères du fichier « les_animaux_malades_de_la_peste.txt » 15 caractères à la fois.

Créer un programme copiant ligne à ligne le fichier « les_animaux_malades_de_la_peste.txt » dans un autre fichier texte au nom de votre choix

Séance 8

Tester les fonctions restantes de la fiche de cours et la compléter.

En fin de séance les élèves ont pu commencer à chercher à créer des fonctions :

1) Découper le texte en phrase.
2) Majuscules/Minuscules
    a. Mettre toutes les phrases du texte en minuscule.
    b. Remettre une majuscule au début de chaque ligne
    c. Mettre une lettre sur deux en majuscule, une lettre sur deux en minuscule.
    d. Mettre des majuscules uniquement au début de chaque phrase.

Pour le dernier quart d'heure : **Retour sur Turtle**
Créer des fonctions dessinant des figures géométriques
En utilisant ces fonctions , des boucles (à l'intérieur d'elles ou à l'extérieur), et des test, créez un dessin le plus beau possible (vous avez au maximum 100 lignes)

Séance 9

Partie I
1h de turtle pour présenter les projets personnels au professeur, pendant l'attente début de recherche :
Créer des fonctions :

1) Stylisé voir le fichier « decodage-cerveau.jpg »
    a. Objectif : Remplacer **dans un mot** systématiquement certaines lettres pas d'autres pour l'écrire de manière stylisée.
        i. Créer une fonction transformant tous les S d'une chaine de caractères en 5
        ii. Généraliser en créant Transform(chaine,initial,final) une fonction transformant dans une chaine de caractère nommé « chaine » un caractère « initial » en sa version transformée nommée « finale » . La question précédente correspondra à l'utilisation de la fonction avec la commande : Transform('S','5')
        iii. Après avoir créé deux listes : une des lettres à remplacer, une de leurs remplacements (par exemple [S,E,…] et [5,3,…]) créer une boucle qui transformera toutes les lettres de la première liste en lettre correspondant dans la seconde.
    b. Faire une fonction étant capable de transformer notre fable en un texte de la même forme que « decodage-cerveau.jpg »
2) Dyslexie. voir le fichier « dyslexie-test.jpg »

a. Permuter aléatoirement toutes les lettres d'un mot
b. Permuter aléatoirement les lettres d'un mot en conservant sa première et sa dernière.
c. Faire la même chose à l'échelle d'un texte.

Partie II

Créer un programme copiant les 5 premières lignes du fichier « les_animaux_malades_de_la_peste.txt » dans un fichier appelé « texte_court.txt »

On a comparé les sorties de deux variations du même programme (on a juste remplacer le paramètre 'r' par le paramètre 'rb' dans la fonction open.

```
fable = open('texte_court.txt','r')
truc=fable.read()
fable.close()
print(truc)
```

ça nous donne : Un mal qui répand la terreur,
Mal que le Ciel en sa fureur
Inventa pour punir les crimes de la terre,
La Peste (puisqu'il faut l'appeler par son nom),
Capable d'enrichir en un jour l'Achéron,
Faisait aux Animaux la guerre.

et

```
fable = open('texte_court.txt','rb')
texteBinaire=fable.read()
fable.close()
print(texteBinaire)
```

ça nous donne : b'Un mal qui r\xe9pand la terreur,\r\nMal que le Ciel en sa fureur\r\nInventa pour punir les crimes de la terre,\r\nLa Peste (puisqu\x92il faut l\x92appeler par son nom),\r\nCapable d\x92enrichir en un jour l\x92Ach\xe9ron,\r\nFaisait aux Animaux la guerre.'

On remarque que les caractères accentués on été remplacés par de drôles d'écritures , par exemple « é » a été remplacé par « \xe9 » , mais ce n'est pas tout, « ' » a été remplacé par \x92 et le saut à la ligne par « \r\n » par contre toutes les lettres non accentués (autrement dit pour les caractères de l'alphabet latin classique)et les espaces sont restés inchangés.
Si au départ le codage des caractère pouvait se faire facilement et occupait peu de mémoire (parce que les caractères anglais étaient la référence) on a commencé a avoir des problèmes quand on a du intégrer les caractères venant d'autres pays, russie, asie, mais aussi les caractères accentués français, espagnols, tchèques) Chaque pays a développé ses conventions, son codage propre, mais ça génère pas mal de problème de compatibilité. Pour permettre à tout le monde de coder de la même manière, on a développé une norme commune : unicode.

Si on ajoute la ligne

```
for car in texteBinaire :
    print(car, end=' ')
```

on obtiendra : 85 110 32 109 97 108 32 113 117 105 32 114 233 112 97 110 100 32 108 97 32 116 101 114 114 101 117 114 44 13 10 77 97 108 32 113 117 101 32 108 101 32 67 105 101 108 32 101 110 32 115 97 32 102 117 114 101 117 114 13 10 73 110 118 101 110 116 97 32 112 111 117 114 32 112 117 110 105 114 32 108 101 115 32 99 114 105 109 101 115 32 100 101 32 108 97 32 116 101 114 114 101 44 13 10 76 97 32 80 101 115 116 101 32 40 112 117 105 115 113 117 146 105 108 32 102 97 117 116 32 108 146 97 112 112 101 108 101 114 32 112 97 114 32 115 111 110 32 110 111 109 41 44 13 10 67 97 112 97 98 108 101 32 100 146 101 110 114 105 99 104 105 114 32 101 110 32 117 110 32 106 111 117 114 32 108 146 65 99 104 233 114 111 110 44 13 10 70 97 105 115 97 105 116 32 97 117 120 32 65 110 105 109 97 117 120 32 108 97 32 103 117 101 114 114 101 46

ça nous permet de voir que même si tout à l'heur le print(texteBinaire) nous donnait une écriture presque lisible du texte, en fait en mémoire on a dans le fichier de type byte une suite de nombre, un par caractère du texte.
Ce que l'on voit sur l'écran et ce qu'on a stocké en mémoire sont deux choses différentes. Chaque caractère a un code.

On peut y acceder avec les fonctions chr et ord , inverses l'une de l'autre. chr(85) va me donner le caractère dont le code est 85 c'est-à-dire « U » alors que ord(« U ») me donnera 85.

Accéder au

chr(numéro) et ord(« caractère ») sont des fonctions inverses l'une de l'autre.

fin de séance (à terminer durant la séance 10)
dicté et commentaire du programme :

```
def Code(textebin,décalage) :
    texteavantcode=bytearray(textebin)
    texteaprescode=bytearray(b'')
    for i in range(len(texteavantcode)) :
        texteaprescode.append(i,texteavantcode[i]+ décalage)
    return texteaprescode
```

après reflexion j'ai effectué quelques changements dans le programme , ce sont les zones surlignées

Séance 10

Partie I
Reprise de la fonction « Code »

La tester avec un décalage puis avec le décalage contraire pour être sûr de bien revenir sur nos pieds

Puis créer un programme qui va prendre un fichier et le crypter en utilisant le même principe

Partie II

Programmation orientée objet

Partie III
Retour sur les exercices à faire pour cette séance (voir séance 9) en parallèle début de la recherche du **projet à rendre le 13 décembre** :
Adapter l'exemple de la fonction cryptfichier(entrée,sortie,décalage) (voir plus haut) du cours pour ne pas décaler toutes les lettres de la même manière, par exemple en utilisant un mot comme clé, (chaque lettre du mot correspond à un décalage, par exemple bca correspondrait aux décalages 2,3 et 1 , et lors du codage la première lettre sera décalée de 2 la suivante de 3 la suivante de 1 , puis la suivante de 2 , puis de 3 , puis de 1 puis …)
attention il faudra être capable de revenir en arrière (de décoder le message)

Séance 11

Correction de l'exercice sur les vecteurs

```
class Point(object):
   "cette classe définit les points du plan"
   def __init__(self, x, y):
      self.abscisse = x
      self.ordonnée = y
   def __str__(self) :
      return("point de coordonnées "+str(self.abscisse)+" et "+str(self.ordonnée))

class Vecteur(object):
   "cette classe définit les vecteurs du plan"
   def __init__(self, départ, arrivée):
      self.abscisse = arrivée.abscisse - départ.abscisse
      self.ordonnée = arrivée.ordonnée - départ.ordonnée

   def __str__(self) :
      return("vecteur d'abscisse "+str(self.abscisse)+" et d'ordonnée "+str(self.ordonnée))

   def __add__(self, autre): # addition vectorielle
      return Vecteur(Point(0,0),Point(self.abscisse + autre.abscisse, self.ordonnée + autre.ordonnée))

   def produit_scalaire(self,vecteur) :
      prosca = self.abscisse*vecteur.abscisse+self.ordonnée*vecteur.ordonnée
      return prosca

   def orthogonalité(self,vecteur) :
      verdict = (self.produit_scalaire(vecteur)==0)
      return verdict

A=Point(2,7)
B=Point(6,8)
C=Point(1,11)
O=Point(0,0)
AB=Vecteur(A,B)
AC=Vecteur(A,C)
BC=Vecteur(B,C)
```

```
class Point(object):
    "cette classe définit les points du plan"
    def __init__(self, x, y):
        self.abscisse = x
        self.ordonnée = y
    def __str__(self) :
        return("point de coordonnées "+str(self.abscisse)+" et "+str(self.ordonnée))

class Vecteur(object):
    "cette classe définit les vecteurs du plan"
    def __init__(self, départ, arrivée):
        self.abscisse = arrivée.abscisse - départ.abscisse
        self.ordonnée = arrivée.ordonnée - départ.ordonnée

    def __str__(self) :
        return("vecteur d'abscisse "+str(self.abscisse)+" et d'ordonnée "+str(self.ordonnée))

    def __add__(self, autre): # addition vectorielle
        return Vecteur(Point(0,0),Point(self.abscisse + autre.abscisse, self.ordonnée + autre.ordonnée))

    def produit_scalaire(self,vecteur) :
        prosca = self.abscisse*vecteur.abscisse+self.ordonnée*vecteur.ordonnée
        return prosca

    def orthogonalité(self,vecteur) :
        verdict = (self.produit_scalaire(vecteur)==0)
        return verdict

A=Point(2,7)
B=Point(6,8)
C=Point(1,11)
O=Point(0,0)
AB=Vecteur(A,B)
AC=Vecteur(A,C)
BC=Vecteur(B,C)
```

Autre exemple utilisant les classes (polymorphisme)

```
class Atome:
  """atomes simplifiés, choisis parmi les 10 premiers éléments du TP"""
  table = [None, ('hydrogène',0), ('hélium',2), ('lithium',4),
       ('béryllium',5), ('bore',6), ('carbone',6), ('azote',7),
       ('oxygène',8), ('fluor',10), ('néon',10)]

  def __init__(self, nat):
    "le n° atomique détermine le n. de protons, d'électrons et de neutrons"
    self.np, self.ne = nat, nat      # nat = numéro atomique
    self.nn = Atome.table[nat][1]    # nb. de neutrons trouvés dans table

  def affiche(self):
    print
    print "Nom de l'élément :", Atome.table[self.np][0]
    print "%s protons, %s électrons, %s neutrons" % \
          (self.np, self.ne, self.nn)

class Ion(Atome):
  """les ions sont des atomes qui ont gagné ou perdu des électrons"""

  def __init__(self, nat, charge):
    "le n° atomique et la charge électrique déterminent l'ion"
    Atome.__init__(self, nat)
    self.ne = self.ne - charge
    self.charge = charge

  def affiche(self):
```

"cette méthode remplace celle héritée de la classe parente"
Atome.affiche(self) # ... tout en l'utilisant elle-même !
print "Particule électrisée. Charge =", self.charge

Programme principal :

a1 = Atome(5)
a2 = Ion(3, 1)
a3 = Ion(8, -2)
a1.affiche()
a2.affiche()
a3.affiche()

```
class Atome:
    """atomes simplifiés, choisis parmi les 10 premiers éléments du TP"""
    table = [None, ('hydrogène',0), ('hélium',2), ('lithium',4),
            ('béryllium',5), ('bore',6), ('carbone',6), ('azote',7),
            ('oxygène',8), ('fluor',10), ('néon',10)]

    def __init__(self, nat):
        "le n° atomique détermine le n. de protons, d'électrons et de neutrons"
        self.np, self.ne = nat, nat       # nat = numéro atomique
        self.nn = Atome.table[nat][1]     # nb. de neutrons trouvés dans table

    def affiche(self):
        print
        print "Nom de l'élément :", Atome.table[self.np][0]
        print "%s protons, %s électrons, %s neutrons" % \
                  (self.np, self.ne, self.nn)

class Ion(Atome):
    """les ions sont des atomes qui ont gagné ou perdu des électrons"""

    def __init__(self, nat, charge):
        "le n° atomique et la charge électrique déterminent l'ion"
        Atome.__init__(self, nat)
        self.ne = self.ne - charge
        self.charge = charge

    def affiche(self):
        "cette méthode remplace celle héritée de la classe parente"
        Atome.affiche(self)                    # ... tout en l'utilisant elle-même !
        print "Particule électrisée. Charge =", self.charge

### Programme principal : ###

a1 = Atome(5)
a2 = Ion(3, 1)
a3 = Ion(8, -2)
a1.affiche()
a2.affiche()
a3.affiche()
```

Puis recherche de l'exercice :

1. Définissez une classe `JeuDeCartes()` permettant d'instancier des objets « jeu de cartes » dont le comportement soit similaire à celui d'un vrai jeu de cartes. La classe devra comporter au moins les trois méthodes suivantes : - méthode constructeur : création et remplissage d'une liste de 52 éléments, qui sont eux-mêmes des tuples de 2 éléments contenant les caractéristiques de chacune des 52 cartes. Pour chacune d'elles, il faut en effet mémoriser séparément un nombre entier indiquant la valeur (2, 3,

4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, les 4 dernières valeurs étant celles des valet, dame, roi et as), et un autre nombre entier indiquant la couleur de la carte (c'est-à-dire 0,1,2,3 pour Cœur, Carreau, Trèfle & Pique). Dans une telle liste, l'élément (11,2) désigne donc le valet de Trèfle, et la liste terminée doit être du type : `[(2, 0), (3,0), (3,0), (4,0), ... ... (12,3), (13,3), (14,3)]` - méthode `nom_carte()` : cette méthode renvoie sous la forme d'une chaîne l'identité d'une carte quelconque, dont on lui a fourni le tuple descripteur en argument. Par exemple, l'instruction : `print jeu.nom_carte((14, 3))` doit provoquer l'affichage de : `As de pique` - méthode `battre()` : comme chacun sait, battre les cartes consiste à les mélanger. Cette méthode sert donc à mélanger les éléments de la liste contenant les cartes, quel qu'en soit le nombre. - méthode `tirer()` : lorsque cette méthode est invoquée, une carte est retirée du jeu. Le tuple contenant sa valeur et sa couleur est renvoyé au programme appelant. On retire toujours la première carte de la liste. Si cette méthode est invoquée alors qu'il ne reste plus aucune carte dans la liste, il faut alors renvoyer l'objet spécial `None` au programme appelant. Exemple d'utilisation de la classe `JeuDeCartes()` :

```
jeu = JeuDeCartes()             # instanciation d'un objet
jeu.battre()                    # mélange des cartes
for n in range(53):             # tirage des 52 cartes :
   c = jeu.tirer()
   if c == None:                # il ne reste plus aucune carte
      print 'Terminé !'         # dans la liste
   else:
      print jeu.nom_carte(c)    # valeur et couleur de la carte
```

2. Complément de l'exercice précédent : Définir deux joueurs A et B. Instancier deux jeux de cartes (un pour chaque joueur) et les mélanger. Ensuite, à l'aide d'une boucle, tirer 52 fois une carte de chacun des deux jeux et comparer leurs valeurs. Si c'est la première des 2 qui a la valeur la plus élevée, on ajoute un point au joueur A. Si la situation contraire se présente, on ajoute un point au joueur B. Si les deux valeurs sont égales, on passe au tirage suivant. Au terme de la boucle, comparer les comptes de A et B pour déterminer le gagnant.

Puis retour sur codage, décodage

Ecriture stylisée

```
def StyliTexte(texte):
   nouveauTexte=texte
   nouveauTexte = nouveauTexte.upper()
   nouveauTexte=nouveauTexte.replace('E','3')
   nouveauTexte=nouveauTexte.replace('S','5')
   nouveauTexte=nouveauTexte.replace('T','7')
   nouveauTexte=nouveauTexte.replace('I','1')
   nouveauTexte=nouveauTexte.replace('A','4')
   nouveauTexte=nouveauTexte.replace('O','0')
   return nouveauTexte
```

C’est surtout l’occasion de découvrir la méthode replace, et de parler de la démarche de programmation qui sera fortement utile pour mener à bien les projets

Bonus : imaginez qu’on soit en train de construire python, plus particulièrement on s’occupe de la classe string(object)
Donner le détail de la méthode « replace » autrement dit qu’à dû écrire le créateur de Python pour créer la méthode telle que vous l’avez découverte.

Ecriture Dyslexique
Il peut être utile de décoller la ponctuation des mots sinon elle risque d'être considérée comme étant la dernière lettre du mot.

Final (pour le projet de fin de cycle) :
Recycler tout ce qu'on a vu pour le travail à rendre à la fin de la séance du 13 décembre
Pour rappel :
Adapter l'exemple de la fonction cryptfichier(entrée,sortie,décalage) (voir plus haut) du cours pour ne pas décaler toutes les lettres de la même manière, par exemple en utilisant un mot comme clé, (chaque lettre du mot correspond à un décalage, par exemple bca correspondrait aux décalages 2,3 et 1 , et lors du codage la première lettre sera décalée de 2 la suivante de 3 la suivante de 1 , puis la suivante de 2 , puis de 3 , puis de 1 puis …)
attention il faudra être capable de revenir en arrière (de décoder le message)

aide :
au final pour le stockage du fichier crypté j'ai choisi d'écrire une suite de nombres séparés par des espaces, chaque nombre correspondant à un caractère (décallé grace au cryptage). C'est un choix personnel, vous êtes libres de le faire autrement.

Ecueil possible : attention en bytearray, il n'y a que 256 caractères

Fonctions pouvant être utiles
Pour convertir de string vers binary : 'chaine de caractères string'.encode('utf-8')
Pour convertir de binary vers string : b'chaine de caractères binary'.decode('utf-8')
Pour convertir directement de string vers bytearray : bytearray('chaine string', 'utf-8')

2489%256 donne le reste de la division euclidienne de 2489 par 256