Les lignes suivantes reprennent les points essentiels quant à l'organisation de cet enseignement.

1. Cours, TD, support de cours
   (a) Il y a 13 séances de TDs dont une notée, pas de cours magistraux
   (b) Le poly du cours est disponible sur Internet : http://www.xavierdupre.fr/mywiki/InitiationPython.
   (c) Sa version papier est disponible à la bibliothèque.
   (d) Les TDs sont accessibles ici : http://www.xavierdupre.fr/enseignement/td_python/python_td_minute/index.html.
2. évaluation :
   – un tutoriel obligatoire après les vacances de Toussaint
   – un TD noté (aux alentours de début décembre) (les annales des précédents examens : http://www.xavierdupre.fr/enseignement/td_python/python_td_simple/index.html)
   – un tutoriel facultatif plus facile avant Toussaint pour les élèves qui débutent et qui souhaitent avoir une note supplémentaire qui aura le même poids que les deux autres
3. Concernant le langage *Python*
   (a) le site officiel http://www.python.org/
   (b) Pourquoi *Python* ?
      – utilisable pour beaucoup d'usages (Web, calcul, interface graphique)
      – nombreuses librairies disponibles sur Internet (calcul scientifique, visualisation, internet, base de données, ...)
      – gratuit, facilement installable (librairie de même)
      – usage croissant

Plan prévisionnel :

Séance	**Date**	**Notions informatiques**	**Notions algorithmiques ou sujet abordé**
1		variables, boucles, tests	recherche dichotomique (programme à trou)
2		fonctions	tri
3		fichiers texte, listes	cryptage, décryptage (Vigenère)
4		dictionnaire	histogramme (déviner la langue d'un texte)
5		classes	carré magique
6		classes	début des graphes : parcours, successeur, ancêtres
7		modules, `numpy`	calcul matriciel, valeurs propres, inversion, utilisation des valeurs pour obtenir des variables normales non-corrélées
8		`numpy`	régression linéaire
9		`re`	discours des présidents [1]
10		séance de préparation aux TDs noté	
11		**TD noté**	**TD noté**
12		graphes, matrices, classes	plus court chemin dans un graphe, connexions avec l'algorithme de Viterbi (chaînes de Markov)
13		interfaces graphiques + `matplotlib`	visualisation d'un fichier texte X,Y,labels

Premier plan prévisionnel.

1. http://freakonometrics.blog.free.fr/index.php?post/2011/05/12/De-quoi-parle-un-president-francais-le-31-decembre, http://freakonometrics.blog.free.fr/public/data/voeuxpresidents.tar

1 TD 1 : Recherche dichotomique minutée

(correction page ??)

Abordé lors de cette séance	
programmation	variables, opérations, boucles, tests
algorithme	recherche dichotomique

Au cours de cette première séance, le chargé de TD vous fera découvrir les bases du langage *Python* et la façon d'écrire des programmes à l'ENSAE, notamment l'usage de l'éditeur de texte *SciTe* utilisé à l'ENSAE et de ses deux parties d'écran, l'une pour le programme, l'autre pour son exécution.

L'objectif de cette séance est aussi de programme la recherche dichotomique [2] qui est un algorithme simple mais qu'il est souvent utile de connaître pour comprendre pourquoi certaines façons de faire sont plus efficaces que d'autres.

Première demi-heure : premiers pas

Pour ce premier TD et cette première demi-heure, l'objectif est de réussir à exécuter le simple programme suivant :

```
x = 3
y = x + 5
print x
print x,y
# commentaire
```

Parvenir à la réalisation de cet objectif nécessite l'ouverture d'un éditeur de texte, l'écriture du programme, la conservation de ce programme sous forme d'un fichier texte, le nom de ce fichier texte devant se terminer par `.py`, l'interprétation de l'instruction `x = 3`, l'interprétation de l'instruction `print`, la correction des erreurs éventuels, l'exécution du programme, l'utilisation de commentaires...

Seconde demi-heure : variable, type

On cherche à jouer avec les différents types de variables et certains opérations qu'on peut faire avec ou non.

```
i = 3                    # entier
r = 3.3                  # réel
s = "exemple"            # chaîne de caractères
c = (4,5)                # couple de valeurs (ou tuple)
l = [ 4, 5, 6.5]         # listes de valeurs ou tableaux
x = l [0]                # obtention du premier élément de la liste l
d = { "un":1, "deux":2 } # dictionnaire de valeurs
y = d ["un"]             # obtention de la valeur associée à l'élément "un"
couple = 4, 5            # collage de deux valeurs en un couple ou (tuple)
```

2. http://fr.wikipedia.org/wiki/Dichotomie

```
print type (l)           # on affiche le type de la variable l
mat = [ [0,1], [2,3] ]   # liste de listes
print mat [0][1]         # obtention du second élément de la première liste
n = None                 # None signifie que la variable existe mais qu'elle ne contient rien
                         # elle est souvent utilisé pour signifier qu'il n'y a pas de résultat
                         # car... une erreur s'est produite, une liste était vide...
```

Chaque lettre ou groupe de lettres désigne une variable, c'est une lettre qui permet de la manipuler quelque soit le contenu qui lui est affecté. Dans l'exemple précédent, `i` désigne 3 mais :

```
i = 3
i = i + 5
```

La variable `i` va d'abord correspondre à 3 puis 8 car on lui affecte une nouvelle valeur déduite de la première. L'ajout du mot-clé `print` permet de voir le résultat si celui-ci est syntaxiquement correct. Sans celui-ci, le programme peut marcher ou non mais rien n'apparaîtra dans la seconde partie d'écran de *SciTe*.

On cherche maintenant à voir comment se comporte les différents types de variables avec différentes opérations. `x` et `y` désigne des objets de même type.

```
print x + y          # addition
print x - y          # soustraction
print x / y          # division
print x * y          # multiplication
print x % y          # modulo
print x == y         # égalité
print x < y          # inférieur
print x <= y         # inférieur ou égal
print min (x,y)      # minimum
print max (x,y)      # maximum
print zip (x,y)      # zip ( [4,5], ["a", "b"] ) donne [ (4,"a"), (5,"b") ]
                     # de deux listes, on passe à une sorte de matrice
print True and False # et logique
print True or False  # ou logique
print (5 < 4) or (5 > 4) # condition
```

Ou encore de manipuler des variables :

```
print -x             # opposé
print len ( [ 4,5] ) # obtention du nombre d'éléments d'une liste
print not False      # négation
```

Les opérations marchent aussi parfois lorsque `x` et `y` ne sont pas de même nature, de type différent :

```
print [ 3,4 ] * 5
print "azerty" * 4
print 4 * "azerty"
print [ 3,4 ] + (3,4)
```

1) Le tableau suivant reprend tous les types standards du langage *Python*. Pourriez-vous remplir les deux cases correspondant à une opération entre les types `int` et `tuple` dans cet ordre puis dans l'autre. Est-ce la même liste ?

	None	bool	int	float	str	tuple	list	dict
None	==							
bool	==	and or						
int			+ - * / min max % == < <=	+ - * / min max % == < <=				
float								
str			*					
tuple								
list								
dict								

2) Le langage *Python* propose des conversions d'un type à un autre. Ainsi, il est possible de convertir un nombre en une chaîne de caractères. Quelques sont les lignes parmi les suivantes qui n'ont pas de sens selon le langage *Python* :

```
print int (3.4)
print list ( (4,5) )
print tuple ( [ 4,5] )
print dict ( [4,5] )
print str ( { "un":1, "deux":2 } )
```

3) Une chaîne de caractères (`str`) contient toujours du texte. Par exemple, si on veut afficher le message, quelle sont les lignes valides parmi les suivantes :

```
x = 1.2
y = 1.2 * 6.55
print "Le prix de la baguette est ", x, "francs."
print "Le prix de la baguette est " + x + "francs."
print "Le prix de la baguette est " + str(x) + "francs."
```

A chaque fois qu'on affiche un résultat numérique, il est implicitement converti en chaîne de caractères.

4) Que vaut l'expression $(0,1) <= (0,2)$? Une idée de comment ce résultat est construit ?

Troisième demi-heure : boucles .

Le tri d'un tableau est une fonctionnalité indispensable et présente dans tous les langages de programmation. En *Python*, on écrira pour une liste :

```
l = [ 4, 6, 3, 4, 2, 9]  # n'importe quelle liste au hasard
l.sort ()
print l                  # le programme affiche la liste triée
```

5) En utilisant la documentation *Python* ou Internet ou un moteur de recherche, trouver comment classer des éléments en sens décroissant. L'instruction `help` dans le programme lui-même retourne l'aide associée à ce qu'il y a entre parenthèses comme `help(l.sort)`. Toute requête sur un moteur de recherche de type *python <élement recherché>* retourne souvent des résultats pertinents.

6) Le programme suivant affiche tous les éléments du tableau un par un grâce à une boucle `for` :

```
for i in xrange (0, len (l)) :
    print l [i]
```

Il utilise une boucle pour parcourir le tableau du début à la fin. Utilisez l'aide concernant la fonction [3] `xrange` pour parcourir le tableau en sens inverse.

7) On souhaite écrire un petit programme qui vérifie que le tableau est trié. Il suffit de compléter le programme suivant :

```
resultat = ...
for i in xrange (0, ...) :
    if ... > ... :
        resultat = False
        break
print "le tableau est-il trié : ", resultat
```

8) Il existe une autre boucle `while`. Complétez le programe suivant pour obtenir la même chose qu'à la question précédente :

```
resultat = ...
i = ...
while i < ...
    if ... > ... :
        resultat = False
        break
print "le tableau est-il trié : ", resultat
```

Quatrième demi-heure : recherche dichotomique .

La recherche dichotomique [4] consiste à chercher un élément `e` dans un tableau trié `l`. On cherche sa position :

- On commence par comparer `e` à l'élément placé au milieu du tableau d'indice `m`, s'ils sont égaux, on a trouvé,
- s'il est inférieur, on sait qu'il se trouve entre les indices 0 et $\texttt{m} - 1$,
- s'il est supérieur, on sait qu'il se trouve entre les indices $\texttt{m} + 1$ et la fin du tableau.

3. Une fonction est différente d'une variable, on la reconnaît grâce aux parenthèses qui suivent un nom. `min`, `max`, `len`, `xrange` sont des fonctions. Entre parenthèses, on trouve les variables dont la fonction a besoin pour fonctionner. Chaque fonction effectue des traitements simples ou complexes mais surtout qu'on souhaite répéter plusieurs fois simplement.

4. http://fr.wikipedia.org/wiki/Dichotomie

Avec une comparaison, on a déjà éliminé une moitié de tableau dans laquelle on sait que `p` ne se trouve pas. On applique le même raisonnement à l'autre moitié pour réduire la partie du tableau dans laquelle on doit chercher.

9) Il ne reste plus qu'à écrire le programme qui effectue cette recherche. On cherche à déterminer la position de l'élément `e` dans la liste `l`. On utilise les indications suivantes :
– il y a une boucle, de préférence `while`
– il y a deux tests
– la liste des variables pourrait être `e`, `l`, `a`, `b`, `m`

10) Que se passe-t-il lorsqu'on cherche un élément qui ne se trouve pas dans le tableau ?

Pour aller plus loin ou pour ceux qui ont fini plus tôt .

11) Si deux joueurs de tennis ont autant de chance l'un que l'autre de gagner un point, combien de points a en moyenne le vainqueur d'un tie-break ? Ecrire un programme qui permette de répondre à la question. Le module `random` permet de générer des nombres aléatoires.

12) Que se passe-t-il si chaque joueur a 75% de chance de gagner un point sur son service ?