L'informatique en PTSI

I Introduction

Ce document est constitué de deux parties. La première partie porte sur le programme d'algorithmique et programmation au lycée.
Bien que diluées dans le cours de mathématiques des trois années, les notions de ce programme vont être une base essentielle pour l'informatique dans le supérieur. Il est essentiel de pratiquer pour progresser.
La seconde partie propose quelques exercices qui permettent de tester, de manière non exhaustive, les principales notions pré-requises pour la rentrée de septembre.

1 Programme d'algorithmique et programmation au lycée

Voici le programme des trois années de lycée. Parfois il se marie bien avec le programme de maths, parfois non. Selon votre formation, il vous sera nécessaire de maitriser ces notions dés les premières semaines en septembre.

2 Seconde

Utiliser les variables et les instructions élémentaires

Contenus
- Variables informatiques de type entier, booléen, flottant, chaîne de caractères.
- Affectation.
- Séquence d'instructions.
- Instruction conditionnelle.
- Boucle bornée (for), boucle non bornée (while).

Capacités attendues
- Choisir ou déterminer le type d'une variable (entier, flottant ou chaîne de caractères).
- Concevoir et écrire une instruction d'affectation, une séquence d'instructions, une instruction conditionnelle.
- Écrire une formule permettant un calcul combinant des variables.
- Programmer, dans des cas simples, une boucle bornée, une boucle non bornée.
- Dans des cas plus complexes : lire, comprendre, modifier ou compléter un algorithme ou un programme.

Notion de fonction

Contenus
- Fonctions à un ou plusieurs arguments.
- Fonction renvoyant un nombre aléatoire. Série statistique obtenue par la répétition de l'appel d'une telle fonction.

Capacités attendues
- Écrire des fonctions simples ; lire, comprendre, modifier, compléter des fonctions plus complexes. Appeler une fonction.
- Lire et comprendre une fonction renvoyant une moyenne, un écart type.
- Écrire des fonctions renvoyant le résultat numérique d'une expérience aléatoire, d'une répétition d'expériences aléatoires indépendantes.

3 Première et terminale générale : spécialité mathématique

Notion de liste

La génération des listes en compréhension et en extension est mise en lien avec la notion d'ensemble. Les conditions apparaissant dans les listes définies en compréhension permettent de travailler la logique. Afin d'éviter des confusions, on se limite aux listes sans présenter d'autres types de collections.

Capacités attendues

- Générer une liste (en extension, par ajouts successifs ou en compréhension).
- Manipuler des éléments d'une liste (ajouter, supprimer ...) et leurs indices.
- Parcourir une liste.
- Itérer sur les éléments d'une liste.

Pour pratiquer

Les IDE

Pour progresser en python, il est essentiel d'avoir un logiciel permettant d'écrire et exécuter du code python, cela s'appelle un IDE (Integrated Development Environment : environnement de développement en français). Vous en aurez besoin durant l'année pour les compte rendus de TP et les DM.

Deux IDE utilisées en PTSI sont : pyzo et spyder (que l'on installe via le logiciel Anaconda).

https ://pyzo.org/install.html

https ://www.anaconda.com/products/distribution

FRANCE IOI et autres

En plus des exercices corrigé de cette fiche, vous pouvez progresser en python à l'aide d'exercices en ligne sur le site France IOI.

Il y a d'autres moyens plus ou moins ludiques de progresser en python :

- Codecombat : jeu très simple au début façon jeu action-aventure.
- Codewars (appli et site) : d'un bon niveau.
- Project Euler : très mathématique.

Mémo Python

Variables

Type de variables :

— Chaines de caractères : des mots ou des phrases, notés entre guillemets.
— Réels
— Entiers (positifs ou négatifs)
— Booléen : peut prendre les valeurs VRAI ou FAUX.

Fonctions

En programmation, les fonctions sont très utiles pour réaliser plusieurs fois la même opération au sein d'un programme. Elles rendent également le code plus lisible et plus clair en le fractionnant en blocs logiques.

```
def nomdefonction(parametre1,parametre2,..):   #il peut y avoir 0,1,2,3 ..paramètres
    traitement
    return resultat  #optionnel
```

Elle peut ou non renvoyer des résultats !

Structure conditionnelle : Si/Sinon

Une structure alternative (ou test) consiste, selon qu'une certaine condition est vérifiée ou non, à effectuer un certain traitement ou un autre. On la rédige de la manière suivante :

```
Si condition alors
   traitement 1                                  #indentation obligatoire
Sinon
   traitement 2                                  #le sinon peut être omis
Fin si
```

En python on écrira :

```
if condition :
   traitement 1
else:
   traitement 2
```

S'il y a plus que 2 cas, on pourra utiliser **elif** qui est une contraction de **else if** :

```
if condition1 :
   traitement 1
elif condition2 :
   traitement 2
else :
   traitement 3
```

Boucle pour

L'objectif de la boucle *Pour* est de faire répéter un nombre de fois connu à l'avance un traitement. On indique alors une valeur de départ, une valeur d'arrivée ou à ne pas dépasser et une valeur d'incrémentation (la manière de passer d'une valeur à la suivante.
On la rédige de la manière suivante :

```
Pour i  allant de (valeur initiale) à (valeur finale) par pas de (incrément) faire
   traitement                                          #indentation obligatoire
Fin Pour
```

Où i est appelé indice de boucle (on utilise parfois les lettres n, j ou k pour les indices.
Le pas d'incrémentation peut-être facultatif quand on avance de 1 en 1.
En python on écrira :

```
for i in range(debut, fin+1, pas):
   traitement
```

i prendra les valeurs de debut jusqu'à fin en allant de pas en pas.
Si le pas vaut 1 on peut ne pas l'écrire dans le range.

Boucle tant que

L'objectif de la boucle Tant que est de faire répéter un certain nombre de fois (non connu à l'avance) un traitement.
Tant qu'une certaine condition est réalisée on continue d'entrer dans la boucle.
On la rédige de la manière suivante :

```
Tant que condition faire
   traitement                                          #indentation obligatoire
Fin Tant que
```

En python on écrira :

```
while condition :
   traitement
```

Listes

Une liste est une collection d'éléments ordonnés, séparés par des virgules, l'ensemble étant enfermé dans des crochets.
Exemple : L=[3,51,-12,9,-12]
La numérotation commence à zéro : L[0] vaut 3. L[1] vaut 51.
len : taille de la liste. len(L) vaut 4 dans notre exemple.
append : ajoute un élément à la liste .
L.append(8) signifie que la liste L devient [3,51,-12,9,-12,8]
remove : enlève la première occurrence de l'élément entre parenthèse
L.remove(-12) signifie que la liste L initiale devient [3,51,9,-12]
On peut parcourir une liste avec une boucle for :

```
for element in L :
   traitement
```

Chaînes de caractères

Une donnée de type "chaîne de caractères" est une suite de caractères quelconques. Une constante chaîne de caractères s'indique en écrivant les caractères en question soit entre apostrophes, soit entre guillemets : 'Bonjour' et "Bonjour" sont deux écritures correctes de la même chaîne.

Exemple

ch1="Bonjour"
ch2="Monsieur"
Que fait ch1[0] ?
Que fait ch2[3] ?
Que fait ch1+ch2 ?
Que fait ch1[2 :] ?
Que fait ch2[:3] ?
On peut parcourir une chaine de caractères avec une boucle for :

```
for lettre in mot :
   traitement
```

II Quelques exercices

1 Des fonctions

Exercice 1 : On considère la fonction définie par $f(x,y) = \begin{cases} 1 \text{ si } x = y \\ 0 \text{ sinon} \end{cases}$

1. Écrire f en langage Python.
2. Que se passe-t-il si vous appliquez f à $x =$ `True` et $y =$ `False` ?

Exercice 2 : Que réalise la fonction `mystere` ci-dessous ?

```
import random as rd

def mystere(n,p):
    X=0
    for i in range(n):
        a=rd.random()
        if a<p:
            X=X+1
    return X
```

2 Des boucles

Exercice 3 : Que font les instructions Python suivantes :

```
for i in range(1, 12, 1):
    print(i)

for i in range(1, 12):
    print(i)

for i in range(0, 13, 3):
    print(i)
```

```
for i in range(6):
    print(i)

for i in range(16,6,-1):
    print(i)

for i in range(19,-7,-4):
    print(i)
```

Exercice 4 : On considère l'algorithme ci-dessous,

```
Algo
Variables : x,n  nombres entiers naturels
Traitement :
                Entrée x
                n prend la valeur 0
                Tant que x différent de 1 faire
                        Si x est un nombre pair alors
                                x prend la valeur x/2
                        Sinon:
                                x prend la valeur 3*x+1
                        n prend la valeur n+1
                Fin Tant que
                Renvoyer n
Fin
```

1. Exécuter l'algorithme à la main et faire un tableau de valeurs pour $x = 5$
2. Même question pour $x = 6$
3. Même question pour $x = 9$
4. Écrire la fonction correspondante en Python de paramètre x et la tester avec les valeurs de x ci-dessus ainsi que pour $x = 127$

Exercice 5 :

1. Écrire le fonction Python ayant pour paramètre n entier strictement positif et qui renvoie $S_n = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$
2. Écrire un programme qui prend en entrée un nombre réel positif A, et qui renvoie la plus petite valeur de n pour laquelle $S_n > A$

3 Des listes et des chaînes de caractères

Exercice 6 : Écrire une fonction en Python qui prend deux paramètres entiers naturels non nuls p et N, et qui renvoie la liste de tous les multiples positifs de p inférieurs ou égaux à N.

Exercice 7 : Écrire pour chacun des cas suivants une fonction qui prend en argument une liste de nombres réels et renvoie :

1. son plus grand élément,
2. la position de son plus petit élément,
3. sa moyenne et sa variance,
4. deux listes composées des éléments pairs d'un côté et impairs de l'autre.

On veillera à ne pas utiliser les fonctions `max` *,* `mean`*, etc.*

Exercice 8 :

1. Écrire une fonction qui détermine si une chaîne contient ou non le caractère "n".
2. Écrire une fonction qui compte le nombre d'apparition du caractère "n" dans une chaîne.
3. Écrire une fonction qui renvoie une chaîne en l'inversant. Ainsi par exemple, "zorglub" deviendra "bulgroz".
4. Écrire une fonction qui détermine si une chaîne de caractères donnée est un palindrome (c'est-à-dire une chaîne qui peut se lire indifféremment dans les deux sens), comme par exemple "radar" ou "S.O.S".

III Corrigés

Corrigé 1 :

```
def f(x,y):
   if x==y:
      return True
   else:
      return False
```

Corrigé 2 : Ce code simule une variable aléatoire suivant une loi binomiale de paramètres n, p

Corrigé 3 : A tester vous-même, pour comprendre la structure du for : valeur de départ, valeur de fin, et le pas.

Corrigé 4 :

```
def syracuse(x):
    n=0
    while x!=1:
        print(x)
        if x%2==0:
            x=x//2
        else:
            x=3*x+1
        n=n+1
    return n
```

syracuse(127) renvoie 46

Corrigé 5 :

```
def somme(n):
    S=0
    for i in range(1,n+1):
       S=S+1/i
    return S

def seuil(A):
    n=0
    S=0
    while S<=A:
       n=n+1
       S=somme(n)
    return n
```

Corrigé 6 :

```
def multipos(p,N):
    L=[]
    x=0
    while x*p<=N:
        L.append(x*p)
        x=x+1
    return L
```

Corrigé 7 :

```
 def plusgrand(L):
    M=L[0]
    taille=len(L)
    for i in range(1,taille):
        if M<L[i]:
           M=L[i]
    return M

def position(L):
    m=L[0]
    pos=0
    taille=len(L)
    for i in range(1,taille):
        if m>L[i]:
           m=L[i]
           pos=i
    return pos

def moyenne(L):
    somme=0
    for i in range(len(L)):
        somme=somme+L[i]
    return somme/len(L)

def variance(L):
    mu=moyenne(L)
    somme=0
    for i in range(len(L)):
        somme=somme+(L[i]-mu)**2
    return somme/len(L)
```

```
def partage(L):
    M=[]
    N=[]
    for i in range(len(L)):
        if L[i]%2==0:
            M.append(L[i])
        else:
            N.append(L[i])
    return M,N
```

Corrigé 8 :

```
def cherche(ch):
    taille=len(ch)
    for i in range(taille):
       if ch[i]=="n":
          return True
    else:
        return False

def nombren(ch):
    compte=0
    taille=len(ch)
    for i in range(taille):
       if ch[i]=="n"
           compte=compte+1
    return compte

def inverse(mot):
   tom=""
   n=len(mot)
   for i in range(n-1,-1,-1):
        tom=tom+mot[i]
   return tom

def palindrome(mot):
    if mot==inverse(mot):
        return True
    else:
        return False
```

Malek Ait Kaci,
malek.aitkaci@gmail.com,