TP Informatique n° 1

Un bon début

Q.1 Suivre les instructions suivantes :

1. Se connecter au pc avec vos identifiants et mot de passe, si vous étiez déjà à Dessaignes, ils n'ont pas changé (en théorie)
2. Si ça ne fonctionne pas, on réessaye en vérifiant qu'on ne tape pas avec des moufles ou que le clavier numérique est activé.
3. Si ça ne fonctionne toujours pas, appeler au secours. Attention si j'arrive à me connecter à votre place le risque de chuter accidentellement par la fenêtre est de 37%.
4. Bravo, vous êtes prêt à commencer.

1 Introduction à Pyzo

Nous utiliserons pyzo pour programmer en python. Ouvrir Pyzo et repérer les différentes fenêtres.
Une instruction sur le TP qui commence par >>> doit être taper dans la console de Python puis excécuter avec la touche entrée.
Ce qui suit les instructions avec >>> est ce que doit vous afficher Pyzo en réponse.

Les entiers : integer, int

Tester les instructions suivantes en console.

L'addition + :
```
>>> 2+4
6
```

La soustraction - :
```
>>> 2-4
-2
```

La multiplication * :
```
>>> 2*4
8
```

Le quotient par la division euclidienne // :
```
>>> 17//5
3
```

Le reste par la division euclidienne % :
```
>>> 17%5
2
```

L'exponentiation ** :
```
>>> 2**10
1024
```

Les réels : flottants, float

On utilise le point "." comme séparateur : c'est anglo-saxon.

Les opérations possibles sur les flottants s'écrivent de la même manière que sur les entiers : les tester. On dispose de plus de la division en virgule flottante `/` :
`>>> 2.5 / 0.2 .`

Les opérations entre un entier et un flottant sont possibles : l'entier est converti 'à la volée' en flottant.
On peut convertir un flottant en entier via `int` c'est une troncature : `>>> int(5.79)`

Un autre type est important, les booléens, dont nous reparlerons bientôt

Les chaînes de caractères, str

Une chaine de caractères permet de représenter les textes. Les caractères les composants sont saisis entre guillemets. Tester les instructions :

L'affectation = :
La concaténation + :

```
>>> texte1 = 'Bonjour'
>>> texte2 = ' les taupins'
>>> texte1 +  texte2
'Bonjour les taupins'
```

La longueur `len` :

```
>>> len('bill')
```

Les tableaux : list

Les tableaux (ou listes) sont formés de plusieurs variables de types simples (ou composés !). Un tableau est un ensemble de valeurs ordonnées dont on peut changer les valeurs. Il y a des subtilités entre listes et tableaux qu'on verra en temps voulu. Sous Python on a des tableaux redimensionables.
Les valeurs sont entre crochets, séparées par des virgules.
Tester :

```
>>> t1 = [2,6,8]
>>> t2 = [ [4,2,3.1] , [3,1,6]]
>>> t3 = [ 'robert', 'bill', 'stan']
```

On peut accéder aux éléments. Attention **ON PART DE ZERO**. Tester et noter les résultats.

```
>>> t1[0]
>>> t1[2]
>>> t1[3]
>>> t2[0]
>>> t2[0][1]
>>> t3[2]
```

On dispose de la longueur aussi, tester `len(t1)`.
Une commande très utile `append` permet d'ajouter une valeur en fin de tableau. Tester

```
>>> T = [ 'Jon','Henry']
>>> T.append('Ike')
>>> T
```

Les n-uplets ou tuples

Un tuple est un ensemble de valeurs ordonnées dont on ne peut pas changer les valeurs. Les valeurs sont entre parenthèses, séparées par des virgules. Les opérations possibles sont :

L'affectation = :

```
>>> t=(6,8,2)
```

L'accès au k-ième élément :
On part de 0 !

```
>>> t[2]  # C'est bien des crochets, attention
2
```

On ne peut pas affecter :

```
>>> t[2]=5
TypeError: 'tuple' object does
not support item assignment
```

La concaténation + :

```
>>> t+(1,6)
(6, 8, 2, 1, 6)
```

La longueur `len` :

```
>>> len(t)
3
```

L'appartenance `in` :

```
>>> 8 in t
True
```

petits exercices

Q.2 Quel est le type adapté pour représenter chacune des données suivantes :

1. la taille d'un individu en mètres
2. l'âge d'un individu en années
3. le nom d'une personne
4. les coordonnées d'un pixel
5. un numéro de téléphone
6. une adresse IP
7. un code pin
8. une adresse postale
9. une date et une heure

Q.3 On peut aussi convertir les nombres en chaînes de caractères, et parfois l'inverse. Tester.

```
>>> str(32)
>>> int('658')
>>> float('35.68')
>>> int('bill')
```

Q.4 On considère la liste suivante :

```
semaine=['Lundi','Mardi','Mercredi','Le 4e jour','Vendredi','Samedi']
```

Sans modifier directement dans la liste :
- Remplacer `'Le 4e jour'` par `'Jeudi'`
- Ajouter `'Dimanche'` en fin de liste

L'éditeur

On clique : File, New, File, Save As : `Info_01.py` sur votre dossier personnel.
Vous êtes dans l'éditeur : c'est nettement plus simple pour taper de longs programmes. On compile avec F5.

Q.5 Écrire et exécuter le programme suivant. Notons les deux façons d'écrire les accents.

```
print('Hello World!')
print("J'adore l'Info")
print('J\'aime Python')
```

Q.6 Taper à la suite :

```
a=5
b=3
c=a+b
print('La somme de ',a,' et ',b,' est de ',c)
```

Là normalement vous avez encore le "Hello world" qui s'affiche : groumpf.
La technique magique secrète pour ne pas avoir à ouvrir une nouvelle feuille à chaque programme est la suivante : ajouter avant et juste après le premier programme """ (trois guillemets), ça le passe en commentaire et donc ne passe pas en machine.

Q.7 Taper les instructions suivantes dans l'éditeur, et exécuter :

```
u = input('Entrez votre nom ')
a = input('Entrez votre année de naissance ')
m = 2019 - a
print('Vous avez ',m,' ans ', u)
```

Que ce passe-t-il ?
Retenir : **Par défaut lors d'un `input` on rentre une chaine de caractère**

Essayer la version suivante :

```
u = input('Entrez votre nom ')
a = int(input('Entrez votre année de naissance ')) # on convertit en entier
m = 2019 - a
print('Vous avez ',m,' ans ', u)
```

Q.8 Créer un petit script qui demande les informations de base sur une personne (comme si vous faisiez votre inscription au lycée). Attention à bien anticiper le type de variable pour chaque information.

Q.9 En reprenant la question précédente, on veut avoir à portée de main toutes ces informations pour tous les élèves de la classe, pour toutes les classes. Une idée d'organisation ?

2 Algorithmique

Instructions booléennes

Q.10 Valeur d'un booléen

Quelle est la valeur des expressions booléennes suivantes

```
3*3.5>10
3.*7==21
3-1>=5
0<10**-20==100**-10
not(2==1==4-3)
not (True and False)
not (not (True))
(5.5*2==1. or 1/2 !=0.5) and (3 % 2 ==0)
```

Vérifier en TP.

Q.11 Travail géométrique

Écrire des expressions booléennes traduisant les conditions suivantes, les nombres étant tous des flottants.

1. Le point de coordonnées (x, y) est à l'intérieur du cercle de centre (z, t) de rayon r.
2. Les points de coordonnées (x, y) et (z, t) sont situés sur une même droite passant par l'origine.
3. il existe un triangle dont les côtés mesurent a, b et c.

Q.12 Travail arithmétique

Écrire des expressions booléennes traduisant les conditions suivantes, les nombres étant tous des entiers.

1. n est divisible par 5.
2. m et n sont tels que l'un est multiple de l'autre
3. m et n sont de même signe
4. m, n et p sont de même signe
5. n est le plus petit multiple de 7 supérieur à 10^{100}
6. m, n et p sont distincts deux à deux

Écrire correctement au concours

La syntaxe Python doit être totalement respectée au concours : indentation, majuscules, : , etc,

Q.13 Corriger le code suivant :

```
x = input( Entrer un nombre)
a = x^2
if a > x
then print( |x| >1 )
else print( |x|<=1 )
```

Q.14 Corriger celui là :

```
x = 1568
a = 14**x
S = 0
While a > 10**5 :
    If a%7 = 0 :
        S = S +1
    a = a // 49
```

Pyzo : divers point

Utilisation du clic droit : hormis les copier/coller il y a des éléments utiles
- Commenter/Décommenter une sélection
- Indenter, désindenter
- Exécuter une seule partie

Mettre en forme : les ligne de commentaire `#` sont là aussi pour s'y retrouver

```
### Partie A
# Q1
if truc blabalbla

# Q2
while machin
```

Branchement conditionnel

Q.15 Nombre de racines

Compléter le programme suivant, qui donne le nombre de racines de $ax^2 + bx + c$ en fonction des coefficients.

```
print('Entrer les coefficients')
a,b,c=float(input()),float(input()),float(input())
delta= ???
if delta ??? :
    print('il y a deux solutions')
if delta ???:
    print('il y a une solution')
if delta ???:
    print('il n\'y a pas de solutions')
```

Le tester avec les polynômes $X^2 + 3X - 5$; $X^2 + 2x + 1$; $X^2 + 4$: on rentre les coefficients.

Q.16 Un test simple

Écrire un programme qui demande à l'utilisateur de saisir un entier. Si cet entier est pair, on renvoie sa moitié, sinon : si il est divisible par 3 on renvoie son tiers, sinon on renvoie son carré.

Boucles Itératives

Q.17 Comprendre le `range`

Tester les ranges suivant, les modifier a souhait pour comprendre.
- Afficher les entiers de 0 à 10 :
```
for k in range(11):
    print(k)
```
- Afficher 5 fois le même texte (on ne se sert donc pas de `k` dans l'instruction répétée) :
```
for k in range(5):
    print('Hello, World!')
```
- Compter à rebours :
```
for k in range(5,0,-1):
    print(k)
print('Partez!')
```

Q.18 Une séquence de caractères

La `sequence` contenant les éléments n'est pas nécessairement une liste de nombre. Taper, exécuter et commenter :

```
chaine = "Bonjour les taupins"
for lettre in chaine:
    print(lettre)
```

Q.19 Distinguer position et valeur

Taper, exécuter et comprendre le programme suivant.

```
a = ['MPSI', 'is', 'my', 'life']
for i in range(len(a)):
    print(i,a[i])
```

Remarque sur `if` et `for`
On peut bien sur placer un `if` dans une boucle `for` (on peut placer ce qu'on veut, pourquoi pas une autre boucle).

Q.20 Affichage

On veut afficher un rectangle de n lignes et p colonnes remplit de `*`.
Par exemple pour $n = 4$ et $p = 5$ on obtient :

```
*****
*****
*****
*****
```

Pour éviter un retour à la ligne à chaque instruction on écrira `print("*",end="")`, la commande `end` lui signifiant d'écrire le symbole proposé au lieu de sauter à la ligne.
Modifier ensuite le programme pour obtenir un triangle rectangle isocèle :

```
*
**
***
****
```

Boucles Conditionnelles

Q.21 Tant que

Déterminer ce que fait le programme suivant :

```
n=int(input('Entrez un entier positif '))
a = n
p = 0
while a !=1:
    a = a//2
    p+= 1
print(p)
```

Modifier le `print` pour expliciter.

Q.22 Boucle infinie

Des valeurs de `n` pour cette boucle posent-elles un petit problème ?

```
n = int(input('Entrez un entier positif '))
while n !=1:
    if n > 4 or n%2==0 :
        n = n//2
    else:
        n = 2*n-3
    print(n)
```

Q.23 Test de validité

Créer un algorithme qui demande l'âge d'une personne et la redemande tant que ce qui est saisit n'est pas un entier positif (et réaliste soyons fou).

3 Applications plus poussées

Les suites

On considère une suite récurrente de la forme $u_{n+1} = f(u_n)$ avec u_0 donné. Pour calculer le terme u_n d'une suite récurrente, on initialise une variable U à u_0 et on effectue le calcul successif des termes dans une boucle itérative.

```
U=valeur de u_0
for k in range(n): # attention k va de 0 à n-1 il faut adapter de temps en temps
    U=f(U)
```

Q.24 Suite récurrente

Utiliser ceci pour calculer des termes de la suite (u_n) avec $u_0 = 5$ et $u_{n+1} = 0,5u_n + 0,5n - 1,5$. Pour $n = 100$ on doit trouver $u_{100} \simeq 95$

Q.25 Liste de termes

Transformer ce programme pour qu'il créer la liste des termes $[u_0, u_1, ...,_n]$ au lieu de renvoyer uniquement u_n. **Technique à retenir pour la suite !**

Q.26 Recherche de seuil

On considère toujours la même suite, qui diverge vers $+\infty$ (c'est évident). Écrire un programme qui renvoi la première valeur de n tel que $u_n > A$ pour A saisi. Pour $A = 1000$ on trouve $n = 1005$.

Calculs de sommes et produits

Q.27 Calcul de sommes

Un calcul d'une somme $\sum_{k=0}^{n} u_k$ suit toujours le même codage :

```
S=0                       \\ On initialise la variable contenant le calcul de la somme à 0
for k in range(n+1):      \\ range(n+1) va bien de 0 à n
   S = S + valeur de u_k           \\ le calcul de la somme
```

On veut effectuer le calcul : $\sum_{k=0}^{n} k = 1 + 2 + 3 + ... + n$. La valeur théorique est de $\frac{n(n+1)}{2}$.

Écrire alors un programme permettant le calcul, et affichant la valeur théorique.

Modifier le programme afin de calculer $\sum_{k=0}^{n} k^2 = 1^2 + 2^2 + 3^2 + ... + n^2$ dont la valeur théorique est $\frac{n(n+1)(2n+1)}{6}$.

Q.28 Calcul de produits

Un calcul d'un produit $\prod_{k=0}^{n} u_k$ suit toujours le même codage :

```
P=1                       \\ On initialise la variable contenant le calcul du produit à 1
for k in range(n+1):      \\ range(n+1) va bien de 0 à n
   P = P * valeur de u_k           \\ le calcul de la somme
```

Appliquer cela au calcul de $a_n = \prod_{k=1}^{n} 2k$ et $a_n = \prod_{k=0}^{n-1}(2k+1)$ (Attention aux bornes !) on demandera à l'utilisateur de rentrer la valeur de n. Pour $n = 5$ on trouve $a_5 = 3840$ et $b_5 = 945$.

Q.29 Sommes doubles

Avec un peu d'imagination calculer les sommes :

$$A = \sum_{1 \leqslant i,j \leqslant 10} \frac{i^2}{j} \qquad B = \sum_{1 \leqslant i < j \leqslant 8} i + j \qquad C = \sum_{i+j=100} (i^2 + j)(i + j^2)$$

Conversions

Pour convertir un nombre X (écrit sous forme d'une chaîne de caractères) d'une base quelconque en base 10 (ou nombre entier), on utilise la fonction int(x [,base]) où base représente la base sous laquelle est écrit le nombre.

```
In [1]: int('AA',16)
Out[1]: 170
In [2]: int('1001',2)
Out[2]: 9
```

Pour convertir un nombre entier N (ou en base 10) en un nombre hexadécimal ou binaire, on utilise la fonction `hex(N)` ou `bin(N)`

```
In [3]: hex(25)
Out[3]: '0x19'
In [4]: bin(65)
Out[4]: '0b1000001'
```

Q.30 Binaire

Dans les cas désespérés il faut écrire soit même un convertisseur. Tester le code suivant :

```
n = # à saisir
q = -1
res = ''
while q != 0:
    q = n // 2
    r = n % 2
    res = 'r' + res
    n = q
print(res)
```

Divers dans le divers

Q.31 Un test aléatoire

Nous allons ici utiliser le hasard via la fonction `random.randint(1,6)` qui donne un nombre entier au hasard entre 1 et 6. Pour cela nous devons importer la bibliothèque `random`. Le programme suivant doit renvoyer le nombre de lancers qu'il faut pour obtenir un 6 sur un dé à 6 faces équilibré. Compléter le.

```
import random
n=1
while random.randint(1,6) ?? :
    n= ???
print('Nombre de lancers',n)
```

On commence à $n = 1$ car on fait au moins un lancer, alors que le `while` ne fait pas d'instruction si on ne rentre pas dans la boucle.
Modifier ce programme pour obtenir le nombre de lancers nécessaires pour obtenir deux (n) six (en tout), puis deux (n) six (de suite).

Q.32 Devinette

Écrire un programme qui prend au hasard un entier n entre 1 et 1000, et demande à l'utilisateur de rentrer un entier et répond "Trop haut" ou "Trop bas", en comptant le nombre d'étapes nécessaires pour gagner.