Semaine n°9 du 25 novembre au 29 Novembre

Informatique(Python) : cf exemples en annexe

- Boucle `while` + Compteur.
- Fonction récursive

Méthodes de calcul : sommes et produits

- Notation $\sum$: définition, linéarité, Chasles, changement d'indice.
- Sommes téléscopiques.
- Sommes classiques à connaître (démonstration exigible) :
  - $\sum_{k=1}^{n} 1 = n, \quad \sum_{k=0}^{n} 1 = n+1, \quad \sum_{k=i}^{j} 1 = j-i+1, \quad \sum_{k=i}^{j} a = (j-i+1)a$ où $a \in \mathbb{C}$
  - $\sum_{k=0}^{n} k = \frac{n(n+1)}{2}, \quad \sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$
  - $\forall q \in C, q \neq 1, \sum_{k=0}^{n} q^k = \frac{1-q^{n+1}}{1-q}$ et de manière générale $\sum_{k=i}^{j} q^k = q^i \frac{1-q^{j-i+1}}{1-q}$
- Sommes doubles du type $\sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} a_{ij}$ et $\sum_{1 \leq i \leq j \leq n} a_{ij}$.
- Produits : définition, notation $\prod$, propriétés, $\prod_{k=i}^{j} c$ où $c \in \mathbb{C}$
- Coefficients binomiaux :
  - Factorielle
  - Définition $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ pour $k \in [\![0, n]\!]$
  - Formules : $\binom{n}{0}, \binom{n}{1}$, symétrie $\binom{n}{k} = \binom{n}{n-k}$, formule "sans nom" $\binom{n+1}{k+1} = \frac{n+1}{k+1}\binom{n}{k}$ (démonstration exigible)
  - Triangle de Pascal (démonstration exigible)
  - Binôme de Newton

Nombres complexes

- Forme algébrique d'un nombre complexe, somme, produit, partie réelle et imaginaire, représentation géométrique.
- Conjugué :définition, interprétation géométrique, propriétés, caractérisation des nombres réels et des nombres imaginaires purs avec le conjugué.
- Module ($|z| = \sqrt{x^2+y^2}$), expression avec le conjugué ($|z| = \sqrt{z\overline{z}}$), interprétation géométrique, propriétés
- Inégalités triangulaires :
  $\forall (z, z') \in \mathbb{C}^2, \quad |z+z'| \leq |z| + |z'|$
  $\forall (z, z') \in \mathbb{C}^2, \quad ||z| - |z'|| \leq |z - z'|$

Remarques aux colleurs

— Merci aussi de poser une petite question d'informatique (cf Annexe).
— La notion de nombre complexe est nouvelle pour beaucoup de nos élèves. Il n' y a pas encore eu de séance de TD sur les complexes.

Exemples de programmes informatiques

Exercice 1

Réaliser une fonction `DepasseValeur` prenant en paramètre un entier naturel M et renvoyant le plus petit entier naturel n tel que $2^n > M$.

```
def DepasseValeur(M):
    n=0  #initialisation du compteur
    while (2**n <=M):
        n=n+1    #incrementation du compteur
    return n
```

Exercice 2

On veut créer le programme suivant : l'ordinateur choisit un nombre entier au hasard entre 1 et 100 puis demande à l'utilisateur de rentrer des nombres entiers jusqu'à deviner le nombre choisi par l'ordinateur. A chaque tentative ratée de l'utilisateur, l'ordinateur indique si le nombre proposé est trop grand ou trop petit par rapport au nombre cherché.

```
from random import *
nb=randint(1,100)
tentative=0   #valeur fausse pour pouvoir rentrer dans la boucle
while (tentative != nb):
    tentative=eval(input("donner un nombre entier entre 1 et 100 : "))
    if tentative <nb:
        print("le nombre proposé est trop petit.")
    elif tentative > nb:
        print("le nombre proposé est trop grand.")
    else:
        print("Gagné!")
```

Exercice 3

Creer un script qui demande à l'utilisateur de donner le mot de passe du labo de bio (”cellule”). L'utilisateur a le droit à trois tentatives maximum.

```
mdp="cellule"
proposition=''    #valeur fausse pour pouvoir rentrer dans la boucle
nbessai = 0      #initialisation du compteur
while (proposition != mdp) and (nbessai <3):
    proposition=input("donner le mot de passe : ")
    nbessai = nbessai + 1    #incrementation du compteur
if (proposition ==mdp):
    print("bienvenu au labo de bio")
else:
    print("nombre de tentatives dépassé")
```

Exercice 4

Créer une fonction récursive `factorielle` qui prend en entrée un entier naturel n et renvoie le résultat de $n!$.

```
def factorielle(n):
    if n==0:
        return 1
    else :
        return n*factorielle(n-1)
```

Exercice 5

Créer une fonction récursive `suite` qui prend en entrée un entier naturel `n` et renvoie la valeur de u_n où (u_n) est la suite définie par :

$$\begin{cases} u_0 = \frac{1}{2} \\ \forall n \in \mathbb{N}, u_{n+1} = 3u_n + 2 \end{cases}$$

```
def suite(n):
    if n==0:
        return 1/2
    else :
        return 3*suite(n-1) +2
```