Semaine n°7 du 16 novembre au 20 novembre

Informatique(Python) : cf exemples en annexe

- Fonctions : `def`, `return`.
- Instructions conditionnelles `if`, `else`, `elif`. (pas de fonction récursive)
- Module maths et random et variable global/local.
- Boucle `while` + Compteur.

Dérivées

- Nombre dérivé en a : définition, interprétation graphique (coefficient directeur de la tangente), fonction dérivée.
- Définition : fonction de classe $\mathcal{C}^1$.
- Dérivées usuelles.
- Opérations sur les dérivées, compositions, formules de compositions usuelles $(\ln(u), e^u, \sqrt{u}, u^n)$.
- Fonctions de deux variables : définition, ensemble de définition, méthode de calcul des dérivées partielles.

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

Remarques aux colleurs

— Merci aussi de poser une petite question d'informatique (cf Annexe).
— Aucun exercice n'a été réalisé sur la formule du binôme de Newton.

Exercice 1

Réaliser une fonction `maximum` prenant en paramètre deux nombres a et b et renvoyant le maximum de ces deux nombres (sans utiliser la fonction max) :

```
def maximum(a,b):
    if a>b:
        return a
    else:
         return b
```

Exercice 2

Créer une fonction `parite` qui prend en paramètre un entier n et renvoie `True` si cet entier est pair et `False` sinon.

```
def parite(n):
    if(n%2==0):
        return True
    else:
        return False
```

Exercice 3

Créer une fonction `entier` qui prend en entrée un entier et qui renvoie True si le nombre est un entier et False sinon

```
from math import floor
def entier(x):
   if (x==floor(x)) :
          return True
    else :
          return False
```

Exercice 4

Créer une fonction `jeu` qui prend en entrée un nombre entier `b` entre 0 et 10 et génère un nombre a au hasard entre 0 et 10. Si `b` et a sont égaux, on renvoie " gagné ", sinon on affiche " perdu ".

```
from random import randint
def jeu(b):
   a=randint(0,10)
   if (a==b) :
        return "gagné"
    else :
        return "perdu"
```

Exercice 5

Créer une fonction python `piecedesequilibre` qui simule l'expérience suivante : On lance une pièce de monnaie qui tombe sur pile avec probabilité 1/4 et qui tombe sur face avec probabilité 3/4.

Première solution :

```
from random import randint
def piecedesequilibre():
     a=randint(1,4)
     if (a==1) :  # 1 chance sur 4 d'obtenir le chiffre 1 quand on choisit un entier
         return "pile"
      else :
         return "face"
```

Deuxième solution :

```
from random import random
def piecedesequilibre():
    a=random() # nombre réel quelconque choisi entre 0 et 1
    if (a<=0.25) : # la proportion de nombres réels a<=0.25 est 0.25
          return "pile"
    else :
          return "face"
```

Exercice 6

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 7

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 8

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
```