Mathématiques pratiques
Exploration par ordinateur

Dans cette séance, on va utiliser l'ordinateur pour explorer de manière *énumérative* certaines notions liées aux ensembles et au dénombrement.

Exercice 0 : rappels sur la récursivité

Soit (u_n) la suite d'éléments de $\mathbb{R}$ définie de manière *récurrente* par $u_0 = 1$ et $\forall n \in \mathbb{N}^*,\ u_n = nu_{n-1}$. Pour calculer les termes de la suite, on utilise une fonction *récursive*.

```python
def u(n):
  # input: an integer n

  # stop condition -> extremely important
  if n == 0:
    return 1

  # recursive call
  else:
    return n*u(n-1)

print(u(5)) # should print 120... but why?
```

Soit (v_n) la suite d'éléments de $\mathbb{R}$ définie de manière *récurrente* par $v_1 = 1$ et $\forall n \in \mathbb{N}^*,\ v_{n+1} = 2v_n + 1$. Donner une expression *explicite* de v_n grâce à l'outil informatique.

Exercice 1 : suites binaires

En mathématiques, un n-uple d'éléments est une *liste* ordonnée d'éléments où les répétitions sont autorisées. On note un n-uple entre parenthèses. Par exemple, $(\pi, \sqrt{2}, \ln 2, \pi)$ est un 4-uple d'éléments de $\mathbb{R}$. En python, on peut utiliser la structure de donnée appelée **tuple** pour représenter les n-uples. Observez par exemple la sortie du code suivant :

```python
t = (3,2,1,3,2) # a tuple
for i in t:
  print(i)
```

Le code suivant permet de créer une **liste** (autre structure de données python) et d'y ajouter des tuples.

```python
# create an empty list
l = list()

# create a tuple and add it to the list
t1 = (1,2)
l.append(t1) # the append function is important!

# create another tuple from t1
t2 = t1 + (3,) # concatenation; do not forget the comma

# add to the list
l.append(t2)

print(l[1]) # should print (1,2,3)
```

Soit $n \in \mathbb{N}$. L'objectif de cette question est de *dénombrer* l'ensemble des suites binaires de taille n. Par exemple $(1, 0, 1)$ est une suite binaire de taille 3. On note `suites(n)` la fonction qui prend en entrée un entier n et renvoie une liste de **toutes** les suites binaires de taille n. Par exemple, `suites(3)` *pourrait*[1] renvoyer :

```
[(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)]
```

Répondez aux questions suivantes :

1. On suppose que toutes les suites binaires de taille n soient connues. Proposer une méthode de construction des suites de taille $n+1$.
2. Écrire avec une approche récursive la fonction `suites(n)`.
3. Compter le nombre de suites binaires de taille n. Était-ce évident ?

Exercice 2 : ensemble des parties

En mathématiques, un ensemble est une *collection* non ordonnée d'éléments où la répétition n'est pas permise. On note un ensemble entre accolades. Par exemple, $\{\pi, \ln 2, \sqrt{3}\}$ est un ensemble de trois éléments. En python, on peut utliser la structure de donnée appelée **set** pour représenter les ensembles. Observez (et commentez) par exemple la sortie du code suivant :

```python
t = {3,2,1,3,2} # a set
for i in t:
  print(i)
```

Le code suivant vous donne quelques informations clés :

```python
# take a tuple and convert it to a set
set((1,2,3,4,3))

# remove redundant elements to create a "real" set
set({1,2,3,4,3})

# pop() removes the first element of the set
E = {1,2,3,4}
x = E.pop()
print(E)
print(x)
```

Pour un ensemble E donné, l'ensemble des parties de E noté $\mathcal{P}(E)$ correspond à l'ensemble de *tous* les sous ensembles possibles de E. Si on note `parties(E)` la fonction qui retourne l'ensemble des parties de E, `parties(1,2,3)` pourrait renvoyer :

```
[set(), {1}, {2}, {1, 2}, {3}, {1, 3}, {2, 3}, {1, 2, 3}]
```

Répondez aux questions suivantes :

1. On suppose qu'on a construit toutes les parties de l'ensemble $\{1, 2\}$. Proposer une méthode de construction des parties de l'ensemble $\{1, 2, 3\}$.
2. Écrire avec une approche récursive la fonction `parties(E)`.
3. Donner une formule pour le nombre de parties d'un ensemble de cardinal $n \in \mathbb{N}$ donné.
4. On note $E = \{x_1, x_2, \ldots, x_n\}$. Pour toute partie $A \subset E$, la fonction *indicatrice* de A notée $\mathbb{1}_A$ est définie par :

$$\mathbb{1}_A : \begin{cases} E \to \{0, 1\} \\ x \mapsto \begin{cases} 1 \text{ si } x \in A \\ 0 \text{ sinon} \end{cases} \end{cases}$$

1. L'ordre des éléments de la liste n'importe pas, c'est le compte qui est important !

La partie A est alors entièrement déterminée par le mot binaire $(\mathbb{1}_A(x_1), \cdots, \mathbb{1}_A(x_n))$. En termes mathématiques, l'application :

$$\varphi : \begin{pmatrix} \mathcal{P}(E) \to \{0,1\}^E \\ A \mapsto \mathbb{1}_A \end{pmatrix}$$

est bijective. Décrire entièrement cette bijection dans le cas $E = \{1, 2\}$ pour récréer l'intuition. Exploiter ensuite φ pour proposer une autre implémentation de `parties(E)` réinvestissant le travail de la question 1. Quel est l'avantage de cette nouvelle implémentation par rapport à la précédente ?

Exercice 3 : parties de taille donnée

Si on reprend l'ensemble renvoyé par `parties(E)` dans le cas $E = \{1, 2, 3\}$, on se rend compte qu'il y a :

1. Une partie de E à 0 élément qui est $\varnothing$
2. Trois parties de E à 1 élément qui sont $\{1\}, \{2\}$ et $\{3\}$
3. Trois parties de E à 2 éléments qui sont $\{1, 2\}, \{2, 3\}$ et $\{1, 3\}$
4. Une partie de E à 1 élément qui est E

Dans le cas où E est un ensemble à $n \in \mathbb{N}$ éléments, le nombre de parties de E à $k \in [\![0, n]\!]$ éléments est donnée par :

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Répondez aux questions suivantes.

1. Générer l'ensemble `parties_taille(k,E)` des parties de taille k d'un ensemble E en sélectionnant dans `parties(E)` les parties de taille requise. Vérifier que vous trouvez le bon nombre de parties avec la commande `binomial(n,k)`.
2. Pour construire les parties à k éléments d'un ensemble E à n éléments, on peut :
   — Distinguer un élément $a \in E$.
   — Construire les parties à k éléments de E ne contenant pas a. Cela revient à construire les parties à k éléments de $E \backslash \{a\}$.
   — Construire les parties à k éléments de E contenant a. Cela revient à prendre les parties à $k-1$ éléments de $E \backslash \{a\}$ puis à ajouter a pour avoir les parties à k éléments.

   En termes de coefficient binomial, on vient de montrer :

   $$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

   Utiliser cette astuce pour avoir une implémentation récursive de `parties_taille(k,E)`.
3. Comparer les approches a) et b). Mettre `%%time` en début de cellule d'exécution pourrait vous aider.