Ewen Le Bihan, 2020

programmationmaths

Rapprocher les deux

https://ewen.works/programath

1 = vs ==

En maths, on note indifféremment le = de la **déclaration** et le = de l'**hypothèse**.

- Ce que j'appelle le = *de la déclaration*, c'est celui qu'on utilise pour poser une variable: "Soit $a = 45$".
- Ce que j'appelle le = *de l'hypothèse*, c'est le = dont on n'est pas sûr, celui que l'on veut prouver ou réfuter: "Supposons $b = c + d$".

En programmation, ce que l'on dit est interprété par une machine, qui ne peut pas déduire cette différence cruciale toute seule. On est donc obligé de noter les "deux =" différement:

Python	Math
`a = 45`	Soit $a = 45$
`a == 45`	$a = 45$
`P = a == 45`	Soit P la propriété "$a = 45$"

Pour que ça ressemble comme deux gouttes d'eau aux propriétés dans une récurrence:

`def P(n):` `  return n == 2*n`	Soit $P(n) = (n = 2n)$

Comme en programmation on pose (ie on déclare) plus que l'on ne teste, le "=" tout simple sert de "= de la déclaration".

D'ailleurs, les propriétés c'est comme les relations, c'est des fonctions à valeurs dans $\mathbb{B}$ mais shhhh...

2 Syntaxe de base

+, × et tout ça.

`(1 + 2) * 3 - 4**6`	$(1 + 2) \cdot 3 - 4^6$
`a//b + a/b`	$\left\lfloor \frac{a}{b} \right\rfloor + \frac{a}{c}$
`r = a%b`	Posons r tel que $a = \left\lfloor \frac{a}{b} \right\rfloor b + r$
`from math import sqrt` `sqrt(5)`	$\sqrt{5}$

3 Ensembles, intervalles

Attention, je parle dans cette partie des *ensembles*, et pas des listes en Python. Les ensemble ça existe aussi en Python, et c'est très similaire aux ensembles en maths:

- Ça se note {`1, 2, 3`} (au lieu de `[1, 2, 3]`)
- Il n'y a aucun doublon dans l'ensemble (chaque élément est différent)
- Il n'y a pas de notion d'ordre des éléments (on a {`2, 4`} `=` {`4, 2`})

Bien sûr, tout s'adapte aux listes, c'est juste qu'avec des ensembles c'est plus simple pour faire des parallèles avec les maths. Pour les spés, on peut considérer la liste en Python comme une matrice-ligne (avec une seule ligne et n colonnes). Mais j'ai pas fait spé donc je vais éviter de dire des conneries, je reste sur des parallèles Python/maths avec des ensembles.

`range(a, b)`	$[\![a, b[\![$
`len(A)`	$\#A$
`{ 2*a for a in A }`	$\{2a,\ a \in A\}$
`{ a for a in A if a**2 == a/2 }`	$\{a \in A,\ a^2 = \frac{a}{2}\}$

Une version qui mélange les deux

`{ f(a) for a in A if P(a) }`	$\{f(a),\ a \in A,\ P(a)\}$ *(pas vraiment légal mais on l'a utilisé une fois)*

4 Opérateurs sur les ensembles

Là, pas d'équivalents pour les listes

`A | B`	$A \cup B$
`A & B`	$A \cap B$
`A ^ B`	$A \Delta B$
`A - B`	$A \setminus B$
`A < B <= C > D >= E`	$A \subsetneq B \subset C \supsetneq D \supset E$
`b in B # Marche aussi avec les listes`	$b \in B$

5 Fonctions

`def f(x):`
`  return 2 * x**2 + 5`
`def f(x, y):`
`  return 2 * x**2 + 5/y`
	Soit $f = x \mapsto 2x^2 + 5$
	Soit $f = (x, y) \mapsto 2x^2 + \frac{5}{y}$

Un petit bonus

`def f(x: int, y: float) -> float:` `  return 2 * x**2 + 5/y`	Soit $f = \begin{cases} \mathbb{Z} \times \mathbb{R} & \to \mathbb{R} \\ (x, y) & \mapsto 2x^2 + \frac{5}{y} \end{cases}$ (Techniquement il faudrait dire "Soit $\mathbb{F}$ l'ensemble des nombre flottants" et remplacer $\mathbb{R}$ par $\mathbb{F}$)

6 Variables liées et libres

Vous vous rappelez le truc chelou de la marmite, et le fait qu'on est pas accès aux variables déclarées dans des fonctions en dehors de celles-ci? Et bah y'a tout pareil en maths enfait.

```
a = 5
def f(x, y):
        return a + x*y
```

Ici, en dehors de `f`, impossible d'accéder à `x` ou `y`, ils n'existent pas. Par contre, on peut accéder à `a` dans `f`, ou en dehors bien sûr.

Et bah en maths aussi c'est pareil:

Soit $a = 5$. Notons $f = (x, y) \mapsto a + xy$.
Ainsi $x = 666$

Dans la définition de f, on utilise a sans problème, par contre, dans la ligne d'après...

WHAT?!
Mais *qui* est x ?

Pas de raison que ce soit différent en programmation, `x` et `y` sont sont des variables *liées* par "`def f(x, y):`" de la même manière que x et y sont liées par "$(\underline{x}, \underline{y}) \mapsto$"