DUT INFO 1 / 2015-2016

Mis à jour le 27 septembre 2015 à 21:13

Programmation et Mathématique: TD et TP

Guillaume CONNAN

UNIVERSITÉ DE NANTES

IUT NANTES Informatique

InforMATHique

Ensembles, logique, arithmétique

Recherche 0 - 1 Ensembles égaux

Parmi les ensembles suivants, quels sont ceux qui sont égaux ?

1. $A = \{x \mid x \in \mathbb{R} \text{ et } x^2 - 4x + 3 = 0\}$
2. $B = \{x \mid x \in \mathbb{R} \text{ et } x^2 - 3x + 2 = 0\}$
3. $C = \{x \mid x \in \mathbb{N} \text{ et } x < 3\}$
4. $D = \{x \mid x \in \mathbb{N} \text{ et } x < 5 \text{ et } x \text{ est impair }\}$
5. $E = \{1, 2\}$
6. $F = \{1, 2, 1\}$
7. $G = \{3, 1\}$
8. $H = \{1, 1, 3\}$

Recherche 0 - 2 Ensembles définis en extension

$E = \{0, 1, 2, 3, 4, 5, 6\}$. Définir en extension (c'est-à-dire lister tous les éléments) les ensembles suivants :

1. $A_1 = \{x \in \mathbb{N} \mid x^2 \in E\}$
2. $A_2 = \{x \in \mathbb{R} \mid x^2 \in E\}$
3. $A_3 = \{x \in E \mid x^2 \in E\}$
4. $A_4 = \{x \in E \mid \sqrt{x} \in E\}$
5. $A_5 = \{x \in E \mid 2x \in E\}$
6. $A_6 = \{x \in E \mid \frac{x}{2} \in E\}$

Recherche 0 - 3 Ensemble défini par compréhension

Écrivez en compréhension l'ensemble $\mathbb{D}$ des décimaux, l'ensemble $\mathbb{Q}$ des rationnels, l'ensemble $\mathbb{C}$ des complexes. L'ensemble des réels $\mathbb{R}$ est un peu plus délicat à définir malgré son nom...

Recherche 0 - 4 Négation des quantificateurs

Quelle peut-être la négation de $(\forall x)(P(x))$? De $(\exists x)(P(x))$?
L'ensemble universel est $S = \{1, 2, 3, 4, 5\}$. Pour chacune des propositions suivantes, donnez-en la négation puis la valeur de vérité.

1. $(\exists x)(x + 3 = 10)$
2. $(\forall x)(x + 3 < 10)$
3. $(\exists x)(x + 3 < 5)$
4. $(\forall x)(x + 3 \leqslant 7)$

Recherche 0 - 5 Pas d'erreur de type !

$E = \{0, 1, 2, 3, 4\}$. Compléter, lorsque c'est possible, par un des symboles (il peut y avoir plusieurs solutions, mais on s'obligera à choisir celle qui donne le plus « de renseignements ») :

$$\in, \ni, \subseteq, \supseteq, =, \neq, \subsetneq, \notin, \cdots$$

1. $2 \cdots E$,
2. $\{2, 3\} \cdots E$,
3. $\{2\} \cdots E$,
4. $\{2, 3, 4\} \cdots \{4, 3, 2\}$,
5. $\{2, 3, 4\} \cdots \{4, 3, 0\}$,
6. $\{\} \cdots E$,
7. $E \cdots E$,
8. $E \cdots \{E\}$,
9. $\{\} \cdots \{E\}$,
10. $E \cdots \{0, 1, 2, 3, \ldots, 10\}$,
11. $\{0, 1, 2, 3, 4, 5\} \cdots E$

Recherche 0 - 6 Erreur de type

Essayez d'expliquer ce qui se passe :

```
In [8]: M = [1,2,3]

In [9]: 1 in M
Out[9]: True

In [10]: [1,2] in M
Out[10]: False
```

Recherche 0 - 7

Créez des fonctions Python qui :

— calcule la somme de 2 entiers ;

— calcule la somme d'une collection d'entiers ;

— calcule la longueur d'une collection ;
— calcule la moyenne d'une collection d'entiers ;
— calcule la somme des carrés d'une collection d'entiers ;
— concatène une liste de chaînes de caractères ;
— met en majuscules une chaîne de caractères.

Recherche 0 - 8 multi-ensemble

On dispose d'un fichier relevant le nombre de lignes de code python écrites par jour par un étudiant sur un an.

1. **i.** On veut tracer la courbe représentant l'évolution du nombre de lignes en fonction du temps : quelle est la structure adaptée pour modéliser le fichier de données ?

ii. Et pour calculer le nombre moyen de lignes écrites ?

2. Remarque : un *multi-ensemble* est un ensemble dont les éléments peuvent être répétés : on dit qu'ils ont un *ordre de multiplicité*.

Par exemple, si $M = \{a, a, b, b, b, f\}$, l'ordre de multiplicité de a est 2, celui de b est 4, celui de f est 1, celui de e est 0.

C'est en fait la donnée d'un *ensemble* d'éléments E et d'une application m de cet ensemble vers $\mathbb{N}$.

Dans l'exemple précédent, E peut être par exemple $\{a, b, c, d, e, f, g\}$ et m est définie sur E par $m(a) = 2$, $m(b) = 4$, $m(c) = m(d) = m(e) = m(g) = 0$ et $m(f) = 1$.

i. Expliquez la différence entre un multi-ensemble et un ensemble, entre un multi-ensemble et une suite.

ii. Proposez au moins quatre fonctions Python qui calculent la multiplicité d'un élément `x` dans un multi-ensemble `M` qui sera représenté par une *liste* d'éléments entre crochets. On pourra comparer les temps d'exécution avec `%timeit` et utiliser des boucles ou des définitions par compréhension.

```
In [33]: M = [ 1,1,1,2,2,3,8,9 ] * 10000

In [34]: %timeit multi4(2, M)
100 loops, best of 3: 2.77 ms per loop
```

iii. Prposez des idées de test d'égalité de deux multi-ensembles.

iv. Déterminez une fonction (au moins) qui calcule la « somme » de deux multi-ensembles : c'est le multi-ensemble obtenu en additionnant les multiplicités de chaque élément.

Par exemple, $\{1, 1, 2, 2\} + \{1, 2, 2, 2, 3, 3\} = \{1, 1, 2, 2, 1, 2, 2, 2, 3, 3\} = \{1, 1, 1, 2, 2, 2, 2, 3, 3\}$.

v. Déterminez une fonction (au moins) qui calcule la « réunion » de deux multi-ensembles : c'est le multi-ensemble obtenu en affectant aux éléments des deux multi-ensembles leur plus grande multiplicité.

Par exemple, $\{1, 1, 2, 2\} \cup \{1, 2, 2, 2, 3, 3\} = \{1, 1, 2, 2, 2, 3, 3\}$.

vi. Déterminez une fonction (au moins) qui calcule l' « intersection » de deux multi-ensembles : c'est le multi-ensemble obtenu en affectant aux éléments communs aux deux multi-ensembles leur plus petite multiplicité.

Par exemple, $\{1, 1, 2, 2\} \cap \{1, 2, 2, 2, 3, 3\} = \{1, 2, 2\}$.

vii. Déterminez une fonction (au moins) qui calcule la « différence » de deux multi-ensembles : c'est le multi-ensemble obtenu en affectant aux éléments du premier le maximum entre 0 et sa multiplicité dans le deuxième.

Par exemple, $\{1, 1, 2, 2\} - \{1, 2, 2, 2, 3, 3\} = \{1\}$.

viii. Est-ce que ces opérations sont compatibles avec les opérations homonymes sur les ensembles ?

Recherche 0 - 9 Paire ordonnée - Produit de deux ensembles

Comment créer de l'ordre à partir du désordre...

Definition 0 - 1 (Paire ordonnée) *On note* $\langle a, b\rangle = \{\{a\}, \{a, b\}\}$

On remarque tout de suite que les rôles de a et b ne sont pas symétriques.
L'égalité des ensembles va nous permettre de définir une égalité des paires : ce n'est pas la même ! Un(e) informathicien(ne) doit bien s'en rendre compte...
Démontrez tout d'abord que :

$$(\{a,b\} =_{\text{ens}} \{x,y\}) \equiv ((a = x \wedge b = y) \vee (a = y \wedge b = x))$$

puis que

$$(\langle a,b\rangle =_{\text{paire}} \langle x,y\rangle) \equiv (a = x \wedge b = y)$$

On peut alors définir le produit (cartésien) de deux ensembles comme étant l'ensemble des paires formées d'éléments de ces ensembles...

Definition 0 - 2

$$A \otimes B = \{\langle a,b\rangle \mid (a \in A) \wedge (b \in B)\}$$

Pour se représenter le produit cartésien $E \otimes F$ avec $E = \{a,b,c\}$ et $F = \{1,2,3,4,5\}$, on peut évidemment l'écrire en extension : il possède $3 \times 5 = 15$ éléments (voir à ce sujet le paragraphe suivant).

$$E \otimes F = \{\langle a,1\rangle, \langle a,2\rangle, \cdots, \langle c,5\rangle\}$$

mais il est souvent préférable de faire appel à un tableau du type suivant :

$\otimes$	**1**	**2**	**3**	**4**	**5**
a	$\langle a,1\rangle$	$\langle a,2\rangle$	$\langle a,3\rangle$	$\langle a,4\rangle$	$\langle a,5\rangle$
b	$\langle b,2\rangle$	$\langle b,2\rangle$	$\langle b,3\rangle$	$\langle b,4\rangle$	$\langle b,5\rangle$
c	$\langle c,1\rangle$	$\langle c,2\rangle$	$\langle c,3\rangle$	$\langle c,4\rangle$	$\langle c,5\rangle$

Avec Python :

```
In [1]: { (x, y) for x in {'a', 'b', 'c'} for y in {1, 2, 3, 4, 5 } }
Out[1]:
{('a', 1),
 ('a', 2),
 ('a', 3),
 ('a', 4),
 ('a', 5),
 ('b', 1),
 ('b', 2),
 ('b', 3),
 ('b', 4),
 ('b', 5),
 ('c', 1),
 ('c', 2),
 ('c', 3),
 ('c', 4),
 ('c', 5)}
```

Recherche 0 - 10 Produit cartésien

1. Soit $A = \{0,1\}$. Déterminer $\left(A^2 \setminus \{\langle 0,0\rangle\}\right) \otimes A$.
2. Soit $A = \{3,5,7\}$ et $B = \{a,b\}$. Déterminer A^2, B^2, $A \otimes B$, $B \otimes A$, $B^2 \otimes A$.

Recherche 0 - 11 CSV

Quelle structure pourriez-vous associer à un fichier CSV ?

Recherche 0 - 12 Ensembles par compréhension, enregistrements et bases de données

Afin de s'exercer à définir des ensembles par compréhension à l'aide de filtres, nous allons introduire des « classes » Python. Il s'agit normalement d'une notion de la Programmation Orientée Objet mais nous ne l'utiliserons en fait que comme un *enregistrement*, structure que l'on retrouve dans tous les types de programmation et que vous explorerez notamment avec M. Morin en C.
Il faudra juste, pour l'instant, prendre ça comme un moyen de créer des « fiches » de renseignements.

Prenons par exemple la manière d'enregistrer un alien :

```
class Alien:
    def __init__(self, Nom, Sexe, Planete, NoCabine):
        "Constructeur d'Aliens"
        self.nom       = Nom
        self.sexe      = Sexe
        self.planete   = Planete
        self.no_cabine = NoCabine
```

Cela indique qu'un alien sera créé en indiquant son nom, son sexe, sa planète d'origine et son numéro de cabine, aucune valeur par défaut n'étant donnée.
On peut donc enregistrer un alien ainsi :

```
In [88]: a = Alien ( "Zorglub", "M", "Trantor", 1 )
```

et avoir accès à ses *attributs* :

```
In [89]: a.nom
Out[89]: 'Zorglub'

In [90]: a.sexe
Out[90]: 'M'

In [91]: a.planete
Out[93]: 'Trantor'

In [94]: a.no_cabine
Out[96]: 1
```

On a créé des « bases » regroupant chaque objet créé selon sa classe :

```
baseAlien =  {
  Alien ( "Zorglub",    "M", "Trantor", 1 ),
  Alien ( "Blorx",      "M", "Euterpe", 2 ),
  Alien ( "Urxiz",      "F", "Aurora",  3 ),
  Alien ( "ZBleurdite", "F", "Trantor", 4 ),
  Alien ( "Darneurane","M", "Trantor", 5 ),
  Alien ( "Mulzo",      "M", "Helicon", 6 ),
  Alien ( "Zzzzz",      "F", "Aurora",  7 ),
  Alien ( "Arggh",      "M", "Nexon",   8 ),
  Alien ( "Joranum",    "F", "Euterpe", 9 )
  }
```

On peut ainsi obtenir l'ensemble des noms des aliens :

```
In [98]: { a.nom for a in baseAlien }
Out[100]:
{'Arggh',
 'Blorx',
 'Darneurane',
 'Joranum',
 'Mulzo',
 'Urxiz',
 'ZBleurdite',
 'Zorglub',
 'Zzzzz'}
```

ou la liste des aliments mangés par les aliens venant de Trantor :

```
In [101]: %paste
lesAlimTrantor = { m.aliment
```

```
                for m in baseMiam
                for a in baseAlien
                if m.nom_alien == a.nom
                if a.planete == "Trantor" }

## -- End pasted text --

In [102]: lesAlimTrantor
Out[106]: {'Bortsch', 'Schwanstucke'}
```

Récupérez le fichier source disponible sur http://informathix.tuxfamily.org/?q=node/153 et répondez aux questions posées.

Recherche 0 - 13 Langage formel / langage courant

Soit p la proposition « Je mâche du foin » et q la proposition « Je mâche du grain ». Traduire par une phrase en français :

1. $\neg p$
2. $p \wedge q$
3. $p \vee q$
4. $q \vee \neg p$
5. $\neg(p \wedge q)$
6. $(\neg p) \wedge (\neg q)$

Recherche 0 - 14 Logique d'Hélène et les garçons

On considère les propositions atomiques B, T, V, C représentant les assertions suivantes : « Hélène est belle », « Hélène joue au Tennis », « Hélène fait du vélo », « Chri-Chri aime Hélène ». Énoncez des phrases simples traduisant les propositions suivantes :

1. i. $\neg B$
   ii. $\neg B \wedge \neg T$
   iii. $\neg(B \vee T)$
   iv. $(B \wedge V) \Longrightarrow C$
   v. $C \Longrightarrow (B \Longleftrightarrow V)$
   vi. $((B \vee T) \wedge \neg V) \Longrightarrow \neg C$
2. Traduisez par une proposition simple les phrases
   i. « Chri-Chri aime Hélène seulement si elle fait du vélo ».
   ii. « Chri-Chri aime Hélène si elle fait du vélo ».
   iii. « Chri-Chri aime Hélène si, et seulement si, elle fait du vélo ».
   iv. « Si Chri-Chri aime Hélène alors elle fait du vélo ».
   v. « Si Hélène fait du vélo alors Chri-Chri l'aime ».
   vi. « Il est suffisant qu'Hélène fasse du vélo pour que Chri-Chri l'aime ».
   vii. « Il est nécessaire qu'Hélène fasse du vélo pour que Chri-Chri l'aime ».

Recherche 0 - 15 Contraposée, réciproque et négation d'une implication

1. i. La contraposée de $A \Longrightarrow B$ est $\neg B \Longrightarrow \neg A$: une implication et sa contraposée sont-elles équivalentes ?
   ii. La réciproque de $A \Longrightarrow B$ est $B \Longrightarrow A$.
   iii. Exprimez $\neg(A \Longrightarrow B)$ avec les opérateurs $\neg$, $\wedge$ et $\vee$ mais sans parenthèses ni $\Longrightarrow$.
2. Déterminez les contraposées, les réciproques puis les négations des propositions suivantes :
   i. Si Max n'étudie pas son cours de maths, les filles le fuient.
   ii. Bill embrassera Céleste seulement s'il écrit de bons programmes en Python
   iii. Paul marquera un but seulement s'il lit la vie de Carl Gauss en latin.

Recherche 0 - 16 Logique et politique

> « In summary, I would like to convey a little positive message to you...I do not have any...Would two negative messages suit you ? »
>
> Woody Allen

Reformulez la phrase suivante, prononcée lors du grand meeting du PUB, Parti Unique Bordure, afin de choisir l'unique candidat des futures élections dictatoriales, sans négation.
Dieter de Villenstein : « Il faut soutenir Josef Chrtzw. Et soutenir l'action de M. Chrtzw, ce n'est pas ne pas soutenir notre ami à tous : Nikalaï Schrpuntz ».

Recherche 0 - 17 Nouvel opérateur logique

Le connecteur$\oplus$ est défini par : $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 1 = 1$ et $0 \oplus 0 = 0$.

1. Simplifier :
   - **i.** $x \oplus 0$
   - **ii.** $x \oplus 1$
   - **iii.** $x \oplus x$
   - **iv.** $x \oplus \neg x$
2. Montrer que :
   - **i.** $x \oplus y \equiv (x \vee y) \wedge \neg (x \wedge y)$
   - **ii.** $x \oplus y \equiv (x \wedge \neg y) \vee (\neg x \wedge y)$
3. L'opération $\oplus$ est-elle commutative ?
4. Vrai ou faux ?
   - **i.** $x \oplus (y \oplus z) \equiv (x \oplus y) \oplus z$
   - **ii.** $x \vee (y \oplus z) \equiv (x \vee y) \oplus (x \vee z)$
   - **iii.** $x \oplus (y \vee z) \equiv (x \oplus y) \vee (x \oplus z)$

Recherche 0 - 18 Si ma tante en avait...

Commentez les phrases suivantes en fonction de la logique des propositions :

1. Si les poules ont des dents alors $2 + 2 = 5$;
2. Si $2 + 2 = 5$ alors les poules n'ont pas de dents ;
3. Si $2 + 2 = 5$ alors les poules ont des dents.

Recherche 0 - 19 Logique au pays des merveilles

« Reprenez donc un peu de thé » propose le Lièvre de Mars.
« Je n'ai rien pris du tout, je ne saurai donc reprendre de rien ! »
« Vous voulez dire que vous ne sauriez reprendre de quelque chose » répartit le Chapelier.
« Quand il n'y a rien, ce n'est pas facile d'en reprendre ».

— Alors comme ça, vous êtes étudiante ?
— Oui, en informatique par exemple.
— Alors que vaut cette fraction : un sur deux sur trois sur quatre ?
— Eh bien ...
— Elle vaut deux tiers, la devança le Loir.
— Ou trois huitièmes si vous préférez, ajouta le Lièvre de Mars.
— Ou encore un sur vingt-quatre, affirma le Chapelier.
— En fait, je crois que...
— Aucune importance ! Dîtes-nous plutôt combien vous voulez de sucre dans votre thé ?
— Deux ou trois, ça dépend de la taille de la tasse.
— Certainement pas, car de toute façon, deux ou trois c'est pareil.
— Parfaitement ! approuva le Loir en fixant Alice qui écarquillait les yeux.
— Ce n'est pourtant pas ce qu'on m'a appris, fit celle-ci.
— Pourtant, ce n'est pas compliqué à comprendre, en voici une démonstration des plus élémentaires
On sait que pour tout entier n on a successivement

$$(n+1)^2 = n^2 + 2n + 1$$

$$(n+1)^2 - 2n - 1 = n^2$$

Retranchons $n(2n+1)$ des deux côtés

$$(n+1)^2 - (n+1)(2n+1) = n^2 - n(2n+1)$$

Mézalor, en ajoutant $(2n+1)^2/4$, on obtient

$$(n+1)^2 - (n+1)(2n+1) + \frac{(2n+1)^2}{4} = n^2 - n(2n+1) + \frac{(2n+1)^2}{4}$$

Soit

$$\left((n+1) - \frac{2n+1}{2}\right)^2 = \left(n - \frac{2n+1}{2}\right)^2$$

En passant à la racine carrée, on obtient

$$(n+1) - \frac{2n+1}{2} = n - \frac{2n+1}{2}$$

d'où

$$n+1 = n$$

Et si je prends $n = 2$, j'ai aussitôt $3 = 2$

— Alors, qu'est-ce que vous en dites ?

— Je...commença Alice.

— D'ailleurs, cela prouve que tous les entiers sont égaux, la coupa le Lièvre de Mars.

— Pas mal du tout ! Qu'en dites-vous mademoiselle l'informaticienne ?

— Je vais vous dire tout de suite ce que j'en pense.

— Ah non ! Nous préférerions de loin que vous pensiez ce que vous allez nous dire.

— C'est pareil ! grinça Alice qui commençait à en avoir assez.

— Comment ça, c'est pareil ? Dire ce que l'on pense ce serait pareil que penser ce que l'on dit ? S'étrangla le Lièvre de Mars.

— Incroyable ! Et manger ce qu'on voit ce serait pareil que voir ce qu'on mange ?

— Mais...

— Et respirer quand on dort pareil que dormir quand on respire ?

— En logique, nous vous mettons 3 sur 5.

— Autant dire moins que un.

— C'est à dire zéro, puisque si 2=3 alors 1=0.

— Parce que chez vous, 3 c'est moins que 1 ? s'indigna Alice.

— On se demande ce qu'on vous apprend à l'école !Tiens, et je peux même vous donner une démonstartion beaucoup plus simple.

Prenez deux nombres non nuls et égaux :

$$a = b$$

Alors

$$a^2 = ab$$

et

$$a^2 - b^2 = ab - b^2$$

Un peu d'identité remarquable :

$$(a+b)(a-b) = b(a-b)$$

Je simplifie :

$$a + b = b$$

Mais comme $a = b$:

$$2b = b$$

et je simplifie :

$$2 = 1$$

— C'est de la folie pure, pensa Alice...

Merci à Pierre Osadtchy

Recherche 0 - 20 jeu logique

Les jeux virtuels sur ordinateur font souvent appel à des énigmes logiques régies par le calcul des propositions. Vous participez actuellement à une partie dont les règles sont les suivantes :
Les propositions composant une énigme sont alternativement vraies et fausses, c'est-à-dire que :

— *soit les propositions de numéro pair sont vraies et les propositions de numéro impair fausses ;*

— *soit les propositions de numéro pair sont fausses et les propositions de numéro impair vraies.*

Dans un labyrinthe, vous vous retrouvez bloqué dans une salle face à une porte sur laquelle se trouvent deux interrupteurs etiquetés A et B en position ouverte. Sur son seuil figure l'inscription suivante :
Pour ouvrir la porte :

P1 *Il faut fermer l'interrupteur A.*

P2 *Il faut fermer simultanément les interrupteurs A et B.*

P3 *Il ne faut pas fermer l'interrupteur B.*

Attention, en cas d'erreur la salle s'auto-détruira...

1. Exprimer P1 , P2 et P3 sous la forme de formules du calcul des propositions dépendant de A et de B.
2. Exprimer la règle du jeu dans le contexte des propositions P1 , P2 et P3.
3. En utilisant le calcul des propositions, déterminer l'action à effectuer pour ouvrir la porte.

Recherche 0 - 21 Complément à 2 puissance n

> *Le monde se sépare en 10 catégories : ceux qui comprennent cette phrase et les autres.*
>
> Zorglub in « Les aventures de Zorglub et Martine au Far-West »

Nous allons, pour favoriser la compréhension et simplifier les écritures, travailler sur 8 bits. Cela signifie que nos nombres entiers peuvent être codés avec 8 chiffres égaux à 0 ou 1 en base 2.

1. Écrivez les nombres de 0 à 17 en base 2. Quel est le plus grand nombre que l'on peut écrire avec 8 bits ?
2. En fait, on aimerait avoir des entiers signés, c'est-à-dire pouvoir représenter des entiers tant positifs que négatifs avec cette contrainte.
   On va donc réserver un bit (le plus à gauche) pour indiquer le signe du nombre : on mettra un 0 s'il est positif et un 1 s'il est négatif.
   Écrivez en compréhension l'ensemble des nombres que l'on peut ainsi représenter.
3. Bon, pas de problème...enfin si quand même un peu. Que pensez-vous de ces deux nombres :

   ```
   00000000 10000000
   ```

4. On voudrait additionner facilement ces nombres. Une idée serait d'utiliser l'addition posée comme à l'école primaire. Par exemple pour calculer $5 + 9$

   ```
       00000101
   +   00001001
   ------------
       00001110
   ```

   On trouve bien 14.
   et maintenant si on veut calculer $5 + (-9)$, qu'est-ce que cela donne avec ce système ?

5. Considérez maintenant le nombre `m = 11111111` et ajoutez-lui 1 (vous n'avez que 8 bits à disposition). Que vaut $m+1$? Que cela vous donne-t-il comme idée ?
Considérez n=11111110 et calculez $n+2$.
Considérez n=11111100 et calculez $n+4$.
Considérez n=11111000 et calculez $n+8$.
Considérez n=11111010 et calculez $n+6$.
Des idées ?

6. Si l'on considère que 2^8 vaut 0 sur 8 bits, que pensez-vous de x et de $2^8 - |x|$ si x est négatif ?
Comment s'écrit $2^8 - |x|$ sur 8 bits ?
Bon, vous commencez à voir ce qui se passe ?
Alors expliquez pourquoi ce code :

```c
#include <stdio.h>

int
main(void)
{
  char n = 52 ;
  char deux = 2 ;
  char centquatre = n * deux  ;
  char deuxcenthuit = centquatre * deux ;
  char quatrecentseize = deuxcenthuit * deux ;

  printf("104 = %d\n208 = %d\n416 = %d\n", centquatre,deuxcenthuit,quatrecentseize );
  return 0;
}
```

donne :

```
104 = 104
208 = -48
416 = -96
```

Généralisez, énoncez votre norme de représentation des entiers sur 8 bits, énoncez un « truc » pour trouver facilement de tête le complément à 2^8.

Recherche 0 - 22 8 bits sur Python

Un entier en base 16 peut être introduit en Python en le faisant précéder de `0x` :

```
In [25]: 0xa1
Out[25]: 161
```

Un entier en base 2 peut être introduit en Python en le faisant précéder de `0b` :

```
In [26]: 0b1101
Out[26]: 13
```

L'opérateur `&` entre deux entiers a et b crée un entier dont les bits sont positionnées à 1 si, et seulement si, ils sont positionnées à 1 dans a et b.

```
In [31]: 0b11 & 0b111000110101
Out[31]: 1

In [32]: 0b11 & 0b111000110111
Out[32]: 3

In [33]: 0b11 & 0b111000110111111011
Out[33]: 3

```

```
In [34]: 0b11 & 0b1110001101111101100
Out[34]: 0
```

Imaginez alors un moyen de calculer sur des entiers 8 bits non signés en Python.
Comment faire s'ils sont signés ?

Recherche 0 - 23 Nombre de chiffres

Trouvez une fonction nb_chiffres qui renvoie le nombre de chiffres de l'écriture décimale de n.
On pourra utiliser le logarithme décimal :

```
from math import log10
```

Recherche 0 - 24 Décomposition en base quelconque

Trouvez une fonction base(b,n) qui renvoie la liste des chiffres de l'écriture normalisée de n en base b avec $1 < b < 10$.

```
In [1]: base(2,97)
[1, 1, 0, 0, 0, 0, 1]
```

Recherche 0 - 25 Écriture littérale

Déterminez une fonction qui renvoie l'écriture littérale d'un nombre dont l'écriture décimale est comprise entre 1 et 999.

Recherche 0 - 26 Persistance d'un entier

Un entier naturel n étant donné, on calcule le produit prod(n) de ses chiffres dans son écriture en base 10 puis le produit des chiffres de prod(n) et on recommence ainsi l'application de prod jusqu'à obtenir un chiffre entre 0 et 9. Le nombre minimal de fois où on applique prod pour transformer n en un chiffre entre 0 et 9 est appelé la *persistance* de n. Par exemple, la persistance de 9 est égale à 0, celle de 97 est égale à 3 car $\text{prod}(97) = 9 \cdot 7 = 63$, $\text{prod}(63) = 6 \cdot 3 = 18$, $\text{prod}(18) = 1 \cdot 8 = 8$. et celle de 9575 est égale à 5.
On voudrait connaître le plus petit entier naturel de persistance 5.

Recherche 0 - 27 Critères de divisibilité

Calculer x modulo n c'est déterminer le reste de la division euclidienne de x par n. On note par exemple :

$$7 \equiv 1[3] \qquad \text{ou} \qquad 7 \equiv_3 1$$

En travaillant modulo le bon entier, démontrer les critères de divisibilité usuels par 2, 3, 4, 5, 9, 10 et 11.
On rappelle que l'écriture en base 10 d'un nombre n est définie par la donnée de l'unique famille $(a_0, a_1, ..., a_k)$ vérifiant :

$$n = a_k 10^k + a_{k-1} 10^{k-1} + \cdots + a_2 10^2 + a_1 10 + a_0$$

avec $a_k \neq 0$ et $0 \leqslant a_1 < 10$ pour tout $i \in \{0, 1, .., k\}$.

Recherche 0 - 28 Liste des diviseurs

Trouver une fonction qui calcule la liste des diviseurs positifs d'un entier naturel donné.

Recherche 0 - 29 Bibinaire

Boby Lapointe, célèbre chanteur français, était aussi mathématicien à ses heures. Ayant trouvé le code binaire trop compliqué à utiliser, il inventa le code... bibinaire. Il suffit de remplacer les chiffres par des lettres. On commence par couper le nombre écrit en binaire en paquets de 2. S'il y a un nombre impair de chiffres, on rajoute un zéro à gauche, ce qui ne modifie pas la valeur de nombre. On commence par le premier groupe de deux chiffres le plus à droite. On remplace 00 par O, 01 par A, 10 par E, 11 par I.

Puis on prend le paquet de deux chiffres suivants en se déplaçant de droite à gauche. On remplace 00 par H, 01 par B, 10 par K, 11 par D.
Pour le paquet suivant, on recommence avec les voyelles. S'il y a encore un groupe, on remplace par une consonne, etc.
Écrivez les nombres de 0 à 25 en bibinaire. Pourquoi Boby a-t-il finement choisi le H ?
Écrivez la table de 3 en bibinaire.
Écrivez les nombres de 255 à 257 en bibinaire. Quel est la base du bibinaire ? Écrivez 355 en bibinaire.
Écrivez KEKIDI et HEHOBOBI en base 10.
Pourquoi est-il pratique d'écrire les nombres en base 2 avec un nombre de chiffres multiple de 4 avant de les traduire en bibinaire ?
Vous pouvez écrire un traducteur bibinaire en Python...

Fonctions, relations

Recherche 0 - 30 Toutes les Bool

Déterminez toutes les fonctions qui prennent deux booléens en argument et renvoient un booléen.

Recherche 0 - 31 operator

Python dispose d'une bibliothèque `operator` comprenant quelques fonctions usuelles.

```
In [1]: from operator import add,mul

In [2]: mul(add(2, mul(4, 6)), add(3, 5))
Out[2]: 208
```

Comment la machine peut évaluer cette expression ?
Il faut savoir comment Python procède.

```
In [9]: def fst(c): return c[0]

In [10]: fst((1,2))
Out[10]: 1

In [11]: fst((1,1/0))
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-11-707229f26363> in <module>()
----> 1 fst((1,1/0))

ZeroDivisionError: division by zero
```

Tout le monde ne fait pas pareil. En Haskell :

```
Prelude> let fst (a,b) = a
Prelude> fst (1,2)
1
Prelude> fst (1,1/0)
1
```

Dresser un arbre d'expression qui traduit l'évalutation de `mul(add(2, mul(4, 6)), add(3, 5))`.

Recherche 0 - 32 Signature mystérieuse

Voici une fonction mystérieuse :

```
def f(n) :
    def g(m) :
        return m*n
    return g
```

Analysez cette fonction, déterminez sa signature, comment l'utiliser, donnez des exemples.

Recherche 0 - 33 Une première fonction cryptographique

On considère la fonction définie sur `range(10)` par :

```
def f(n) :
    return n + (-1)**n
```

1. Calculez `f(f(n))` pour tout entier n de `range(10)`.
2. Pouvez-vous utiliser cette fonction pour crypter votre code de carte bleue ?

Recherche 0 - 34 Trouver x

Voici une petite fonction :

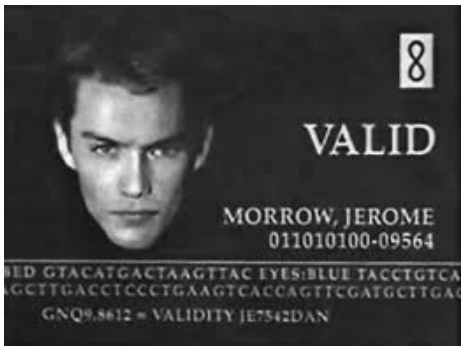
```
x = 0

def change_x(val) :
    global x
    print("valeur x avant  : " + str(x))
    print("adresse x avant : " + str(id(x)))
    x = val
    print("valeur x après  : " + str(x))
    print("adresse x après : " + str(id(x)))
```

Et quelques-unes de ses exécutions :

```
In [37]: change_x(2)
valeur x avant  : 0
adresse x avant : 10771488
valeur x après  : 2
adresse x après : 10771552

In [38]: change_x(2)
valeur x avant  : 2
adresse x avant : 10771552
valeur x après  : 2
adresse x après : 10771552

In [39]: change_x('a')
valeur x avant  : 2
adresse x avant : 10771552
valeur x après  : a
adresse x après : 139992609175120

In [40]: type(change_x)
Out[40]: function

In [41]: type(change_x(2))
valeur x avant  : a
adresse x avant : 139992609175120
valeur x après  : 2
adresse x après : 10771552
Out[41]: NoneType

In [42]: change_x(change_x(2))
valeur x avant  : 2
adresse x avant : 10771552
valeur x après  : 2
adresse x après : 10771552
valeur x avant  : 2
adresse x avant : 10771552
valeur x après  : None
adresse x après : 10288288

In [43]: print(change_x(2))
valeur x avant  : None
adresse x avant : 10288288
valeur x après  : 2
adresse x après : 10771552
None
```

Commentez...

Recherche 0 - 35 Traquez les impures



Nous avons déjà trouvé la fonction impure `print`. Traquez d'autres fonctions impures se cachant dans Python...

Recherche 0 - 36 Approximation affine d'une fonction numérique

Vous vous souvenez que $f'(x) = \lim_{dx \to 0} \dfrac{f(x+dx) - f(x)}{dx}$.
On peut donc considérer que pour dx « petit », $f'(x) \approx \frac{f(x+dx)-f(x)}{dx}$.
Déterminez une fonction qui a comme paramètres une fonction à dériver et un pas dx et qui renvoie l'approximation de la fonction dérivée.
Testez sur plusieurs exemples de fonctions que vous connaissez bien, faites varier aussi le pas. Commentaires ?

Recherche 0 - 37 Composée n fois

Déterminez une fonction qui renvoie la composée itérée d'une fonction donnée en argument.

Recherche 0 - 38 Moyenne olympique

Décrire une fonction qui étant donnés quatre flottants leur associe la moyenne des deux nombres parmi les quatre qui ne sont ni le plus grand ni le plus petit. Par exemple, la moyenne olympique de 10, 8, 12 et 14 est 11 et celle de 12, 12, 12 et 12 est 12. Généraliser.

Recherche 0 - 39 Curryfication

Vous n'avez travaillé le plus souvent qu'avec des fonctions d'une variable. Or la multiplication est en fait une fonction de deux variables :

```
1 def mul(x,y) :
2     return x * y
```

On peut fabriquer une fonction qui multiplie un nombre par 2 :

```
1 def mul_2(n) :
2     return 2 * n
```

On peut fabriquer une fonction qui multiplie un nombre par 3 :

```
1 def mul_3(n) :
2     return 3 * n
```

On peut fabriquer une fonction plus générale qui renvoie ces fonctions, pour 2, 3 ou n'importe quel autre nombre :

```
1 def mul_curry(k) :
2     return lambda n : k * n
```

Ainsi `mul(3) == mul_3`
Alors `mul(x,y) = mul_curry(x)(y)`
Dessinez des flèches pour illustrer cela.
Fabriquer une fonction `curry` telle que `curry(mul) == mul_curry`
Dessinez les flèches correspondantes. Pensez types.

Recherche 0 - 40 filtres

Créez une fonction **filtre** qui prend une liste en argument et un prédicat et renvoie les éléments de la liste qui vérifient le prédicat.
Créez une fonction **prend_tant_que** qui prend une liste en argument et un prédicat et renvoie les premiers éléments de la liste qui vérifient le prédicat et s'arrête au premier échec.
Créez une fonction **partition** qui prend une liste en argument et un prédicat et renvoie deux listes : l'une qui contient les éléments qui vérifient le prédicat et une autre qui contient les autres.

Recherche 0 - 41

On se propose de démontrer que tous les étudiants d'IUT ont le même âge et, pour cela, on note $P(n)$ l'affirmation « *si on choisit n étudiants ($n \in \mathbb{N}^*$), il est sûr qu'ils ont tous le même âge* »
Il est clair que $P(1)$ est vraie.
Démontrons que $P(n) \to P(n+1)$. Pour cela nous supposons que $P(n)$ est vraie (c'est l'hypothèse de récurrence) et nous choisissons un groupe quelconque de $n+1$ étudiants que nous ordonnons par ordre alphabétique (pourquoi pas). D'après l'hypothèse de récurrence, les n premiers de l'ordre alphabétique ont tous le même âge ainsi que les n derniers. Comme ces deux groupes de n étudiants ont au moins un étudiant en commun, on en déduit qu'ils ont tous le même âge.
Nous venons de démontrer que $P(n) \to P(n+1)$ pour tout $n \geq 1$ et, comme $P(1)$ est vrai, $P(n)$ est toujours vrai pour tout $n \geqslant 1$. Y a-t-il une erreur dans le raisonnement ?

Recherche 0 - 42 Javanais for dummies

On va simplifier le javanais habituel sans tenir compte de la découpe selon les syllabes. On introduit les lettres « av » devant la première voyelle rencontrée dans le mot en partant de la gauche. Créez une foncton `javanais(mot)` :

```
>>> javanais("chrome")
"chravome"

>>> javanais("air")
"avair"
```

Recherche 0 - 43 CCP MP 2015

Non, non, ce n'est pas un extrait d'un livre de seconde, c'est un sujet 0 proposé pour l'épreuve de Math du concours CCP...

1. Écrire une fonction factorielle qui prend en argument un entier naturel n et renvoie $n!$ (on n'accepetra pas bien sûr de réponse utilisant la propre fonction factorielle du module math de Python ou Scilab).
2. Écrire une fonction seuil qui prend en argument un entier M et renvoie le plus petit entier naturel n tel que $n! > M$.
3. Écrire une fonction booléenne nommée est_divisible, qui pend en argument un entier naturel n et renvoie True si $n!$ est divisible par $n+1$ et False sinon.
4. On considère la fonction suivante nommée mystere :

```
def mystere(n):
    s = 0
    for k in range(1,n+1):
        s = s + factorielle(k)
    return s
```

   i. Quelle valeur renvoie mystere(4) ?
   ii. Déterminer le nombre de multiplications qu'effectue mystere(n).
   iii. Proposer une amélioration du script de la fonction mystere afin d'obtenir une complexité linéaire.

Recherche 0 - 44 Ils ont osé : écrit Maths II CCP MP 2015

1. Donnez la décomposition en binaire (base 2) de l'entier 21.
On considère la fonction `mystere` suivante :

```
1 def mystere(n, b) :
2     """Données : n > 0 un entier et b > 0 un entier
3        Résultat : ......."""
4     t = [] # tableau vide
5     while n > 0 :
6         c = n % b
7         t.append(c)
8         n = n // b
9     return t
```

On rappelle que la méthode append rajoute un élément en fin de liste. Si l'on choisit par exemple `t = [4,5,6]`, alors, après avoir exécuté **`t.append(12)`**, la liste a pour valeur `[4,5,6,12]`.
Pour $k \in \mathbb{N}^*$, on note c_k, t_k et n_k les valeurs prises par les variables c, t et n à la sortie de la k-ème itération de la boucle « while ».

2. Quelle est la valeur renvoyée lorsqu'on exécute **`mystere(256,10)`** ?
On recopiera et complètera le tableau suivant en rajoutant les colonnes nécessaires pour tracer entièrement l'exécution :

k	1	1	...
c_k			...
t_k			...
n_k			...

3. Soit $n > 0$ un entier. On exécute **`mystere(n,10)`**. On pose $n_0 = n$.

i. Justifier la terminaison de la boucle `while`.

ii. On ote p le nombre d'itérations lors de l'exécution de **`mystere(n, 10)`**. Justifiez que pour tout $k \in [\![0, p]\!]$, on a $n_k \leqslant \frac{n}{10^k}$. Déduisez-en une majoration de p en fonction de n.

4. En vous aidant du script de la fonction **`mystere`**, écrivez une fonction **`mystere`** qui prend en argument un entier naturel et renvoie la somme de ses chiffres. Par exemple, **`somme_chiffres(256)`** devra renvoyer 13.

5. Écrivez une version récursive de la fonction **`somme_chiffres`** que vous nommerez **`somme_rec`**.

Recherche 0 - 45 lambda

Calculez, écrivez sans λ mais avec $\mapsto$ et traduisez en Python :

1. $(\lambda x \cdot (x+y))(2)$
2. $(\lambda x \cdot (x+y))(2y)$
3. $(\lambda x \cdot (\lambda y \cdot (x+y))(3))(2)$
4. $(\lambda x \cdot (\lambda y \cdot (x+y))(a))(b)$
5. $(\lambda x \cdot 4)(2)$

Recherche 0 - 46 Composition

Soit $f = \{(1,c),(2,a),(3,b),(4,c)\}$, $g = \{(a,2),(b,1),(c,4)\}$ et $h = \{(1,A),(2,B),(3,D),(4,D)\}$.
Calculez, si c'est possible, $g \circ f$, $f \circ g$, $h \circ (g \circ f)$, $h \circ g$ et $(h \circ g) \circ f$.
Déterminez

Recherche 0 - 47 fonction injective, surjective

Déterminez si les fonctions suivantes sont injectives, surjectives, bijectives :

1. $f : \mathbb{N} \to \mathbb{N}$, $x \mapsto x^2 + 2$
2. $f : \mathbb{N} \to \mathbb{N}$, $x \mapsto x \bmod 3$
3. $f : \mathbb{N} \to \mathbb{N}$, $x \mapsto \begin{cases} 1 \text{ si } x \text{ est pair} \\ 0 \text{ sinon} \end{cases}$
4. $f : \mathbb{N} \to \{0,1\}$, $x \mapsto \begin{cases} 1 \text{ si } x \text{ est pair} \\ 0 \text{ sinon} \end{cases}$
5. $f : \mathbb{N} \otimes \mathbb{N} \to \mathbb{N}$, $f(x,y) = x + y$

Recherche 0 - 48 Exponentiation

1. Voici une définition très classique de la puissance entière d'un entier : $x^0 = 1$ et $x^n = x \times x^{n-1}$
Cela donne :

```
def pow1(a, n) :
    assert n >= 0, "L'exposant doit être entier naturel"
    if n == 0 :
        return 1
    else :
        return a * pow1(a, n - 1)
```

Démontrez alors que pour tous entiers naturels x, n et m, `pow1(x, n + m) == pow1(x, n) * pow1(x,m)`
Attention, il y a une petite subtilité...

2. Démontrez ensuite, toujours par induction, que `pow1(x, n) * pow1(x, n) == pow1(x*x, n)`.

3. Mais il y a mieux à faire en remarquant que, quand n est pair, $x^n = (x \times x)^{n/2}$: quel est l'intérêt, algorithmiquement parlant ? Comment en tirer parti ?
Vous pourrez avoir besoin de l'opérateur // qui retourne le quotient de la division euclidienne de deux entiers.
Vous noterez par exemple pow2 cet nouvelle fonction.
Souvenez-vous de vos cours de 6ème. Effectuer la division euclidienne de deux entiers a et b, b étant non nul, c'est déterminer les entiers q et r tels que :

$$a = b \times q + r \qquad 0 \leqslant r < |b|$$

Comparez le nombre d'étapes nécessaires pour calculer 3^{25} avec l'une et l'autre méthode. Nous essaierons de généraliser tout à l'heure mais revenons d'abord à nos moutons.

4. Il s'agit maintenant de démontrer que les fonctions pow1 et pow2 sont équivalentes.
Le cas n impair est assez direct. Pour le cas pair, il y a plus de travail...

5. Moralité ? Parmi ces deux codes, un est simple à lire et peu efficace, l'autre est compliqué à lire et efficace : faut-il malgré tout jeter le premier à la poubelle ?

6. Peut-on avoir un ordre de grandeur du nombre d'étapes nécessaires au calcul de x^n ? Cela peut avoir une importance capitale, notamment en cryptographie où on calcule régulièrement des entiers à la puissance des nombres d'une centaine de chiffres.
Faisons pour cela un petit rappel sur une fonction primordiale en informatique. Nous voudrions savoir combien de bits sont nécessaires à l'écriture d'un entier et combien de bits sont égaux à 1...Menez l'enquête. Peut-être aurez-vous besoin d'introduire le logarithme de base 2 :

$$\log_2(x) = \frac{\ln x}{\ln 2}$$

Recherche 0 - 49 Fabriquons la multiplication

On a défini l'exponentiation comme itération de la multiplication. On peut de même définir la multiplication comme itération de l'addition :

```
def mul1(a, b) :
    if b < 0 :
        return -mul1(a, -b)
    if b == 0 :
        return 0
    return a + mul1(a, b - 1)
```

Combien faut-il effectuer d'opérations pour calculer `mul1(a,b)` ?
On dispose de deux opérations élémentaires sur les chaînes de bits qui permettent de doubler ou de couper en 2 un nombre en décalant d'un bit vers la droite ou la gauche :

```
def double(n) :
    return n << 1
```

```
3
4  def demi(n) :
5      return n >> 1
```

Déterminez alors une multiplication plus efficace en vous inspirant de ce qui a été fait pour l'exponentiation.
Démontrez que votre nouvelle fonction calcule effectivement le produit de ses arguments.
Donnez des processus récursifs et itératifs.

Recherche 0 - 50 Nombres premiers

Définition

Quoi de plus simple qu'un nombre premier :

Définition 0.1 Un entier naturel est dit premier s'il est supérieur (i.e. supérieur ou égal) à 2 et n'est divisible que par 1 et lui-même.

et pourtant, ils renferment tant de mystères que les plus grands esprits depuis des siècles n'ont toujours pas réussi à en percer tous les secrets et ce malgré les énormes progrès technologiques et les investissements colossaux consentis par les pouvoirs tant civils que militaires pour assurer ou percer la confidentialité des transmissions de toutes natures qui continuent de dépendre d'une meilleure connaissance des nombres premiers, ce que nous verrons un peu plus loin. Qui sont-ils ? Combien sont-ils ? Où sont-ils ? À quoi servent-ils ? Nous essaierons de donner quelques éléments de réponses à ces questions.
Mais tout d'abord, pourquoi jouent-ils un rôle si important ? Fondamentalement, il existe deux manières d'engendrer $\mathbb{N}$:

- si on veut engendrer $\mathbb{N}$ en utilisant l'addition, on s'aperçoit que le nombre 1 nous suffit : on « fabrique » 2 en additionnant 1 avec lui-même ; 3 en additionnant 1 avec 2, etc.
- si on veut engendrer $\mathbb{N}$ en utilisant la multiplication, là, les choses se compliquent. Pour « fabriquer » 2, il faut le créer ; même problème pour 3. On fabrique 4 en multipliant 2 avec lui-même, mais il faut créer 5. On fabrique 6 en multipliant 3 avec 2. On crée 7. On fabrique 8 à partir de 2. On fabrique 9 à partir de 3. On fabrique 10 à partir de 2 et 5, etc.

  Les nombres que l'on est obligés de créer sont les briques nécessaires à fabriquer tous les autres. C'est bien plus compliqué que l'addition me direz-vous, mais la multiplication est plus « puissante » et nous permet d'aller bien plus vite et plus loin.

Les nombres premiers sont donc ces éléments qui nous permettent de fabriquer tous les autres. Un des premiers problèmes étudiés à été de savoir s'ils peuvent tenir dans une boîte. Euclide à répondu à cette question il y a vingt-trois siècles et la réponse est non.
Pour le prouver, nous aurons besoin d'un résultat intermédiaire :

Théorème 0 - 1 Tout entier naturel admet au moins un diviseur premier.

Si ce nombre, appelons-le n, est premier, tout va bien.
Sinon, l'ensemble de ses diviseurs étant non vide (n est dedans) et borné (par 1 et lui-même), il admet un plus petit élément $p \neq 1$. Ce nombre n'est pas divisible par un autre, sinon ce nombre plus petit que p divisant p diviserait n et alors p ne serait plus le plus petit diviseur de n. Le nombre p est donc premier et voilà.

Théorème 0 - 2 Il y a une infinité de nombres premiers.

Raisonnons par l'absurde et supposons qu'il existe exactement n nombres premiers qu'on nommera p_1, p_2,$\cdots$, p_n et appelons N le nombre

$$N = p_1 p_2 \cdots p_n + 1$$

Il est plus grand que tous les p_i, donc il n'est pas premier d'après notre hypothèse, donc il admet un diviseur premier p qui est donc un des p_i, puisqu'il n'y a qu'eux. Soit i_0 tel que $p = p_{i_0}$. Alors p divise $p_1 p_2 \cdots p_n$. Or il divise N, donc il divise leur différence $N - p_1 p_2 \cdots p_n$, c'est à dire 1, donc $p = 1$ ce qui est absurde puisqu'il est premier. Ainsi, il n'existe pas de plus grand nombre premier.

Il y en a donc une infinité et l'aventure ne fait que commencer.

Comment vérifier qu'un nombre est premier ?

Force brute

Créez une fonction divise k n qui teste si l'entier k divise n.

```
In [70]: divise(7,63)
Out[70]: True
```

Créez une fonction les_diviseurs :: Int -> [Int] qui renvoie l'ensemble des diviseurs de n. On pourra utiliser la méthode **extend** :

```
In [71]: les_diviseurs(100)
Out[71]: {1, 2, 4, 5, 20, 25, 50, 100}
```

Déduisez-en un test est_premier :: Int -> Bool

```
In [77]: est_premier(2**10 - 1)
Out[77]: False
```

On peut alors créer une fonction find :

```
find :: (a -> Bool) -> [a] -> a
The find function takes a predicate and a list and returns the first element in
 the list matching the predicate, or None if there is no such element.
```

pour déterminer une fonction prochainPremier qui calcule le nombre premier qui suit l'argument :

```
In [81]: prochain_premier(2**10 - 1)
Out[81]: 1031
```

Cependant, l'efficacité de cette fonction laisse à désirer...Comment peut-on améliorer les_diviseurs ?
Mais si on ne cherche que les nombres premiers, a-t-on besoin de tous les diviseurs ?

Étudions le problème pour être plus efficaces

Observons les diviseurs de 1321 par exemple : si aucune des 1319 divisions ne « tombe juste », alors on pourra dire que 1321 premier. Et puis d'abord, il est impair, il ne semble pas être divisible par 3, alors pourquoi pas...

diviseur	2	3	4	5	6	7	8	9	10	11	12	13
quotient	660,5	440,3	330,3	264,2	220,2	188,7	165,1	146,8	132,1	120,1	110,1	101,6
diviseur	14	15	16	17	18	19	20	21	22	23	24	25
quotient	94,36	88,07	82,56	77,71	73,39	69,53	66,05	62,90	60,05	57,43	55,04	52,84
diviseur	26	27	28	29	30	31	32	33	34	35	36	37
quotient	50,81	48,93	47,18	45,55	44,03	42,61	41,28	40,03	38,85	37,74	36,69	35,70

Le tableau est incomplet : il reste encore à essayer les quotients de 38 à 1320, mais cela aurait été mauvais pour la planète.
Tout d'abord, nous n'avons pas trouvé de quotient entier en ce début d'enquête. Nous observons de plus que si la suite des diviseurs est croissante, celle des quotients est décroissante. Vous avez sûrement remarqué qu'à partir de 37, les quotients sont inférieurs aux diviseurs donc nous pouvons nous arrêter là : si pour un diviseur supérieur à 37, on trouvait un quotient entier q, alors en divisant 1321 par q, qui est inférieur à 37, on devrait se trouver au début de notre liste de diviseurs et ce diviseur serait entier. Le problème, c'est qu'aucune division par un entier inférieur à 37 n'a donné de quotient entier donc on peut s'épargner les 1283 divisions restantes.

Mine de rien, nous venons de franchir un grand pas dans la théorie des nombres : pour tester si l'entier 1321 est premier, il nous a suffit d'effectuer 36 divisions [a]. Deux problèmes se posent maintenant : pourquoi 36 et pouvons-nous généraliser ce résultat aux autres entiers ?
Un petit Joker : $\sqrt{1321} \approx 36,3...$

a. Et encore, nous aurions pu faire mieux comme nous allons le voir tout de suite après.

Reprenons la propriété 0 - 1 page 19. Tout naturel n admet un plus petit diviseur p qui est premier. Écartons le cas où n est premier et donnons un nom aux nombres qui restent :

Definition 0 - 3 *Un entier naturel autre que 1 qui n'est pas premier est dit* **composé**.

L'entier n étant composé, il s'écrit donc $n = pq$, avec q un entier supérieur à p (car p est le plus petit diviseur). Ainsi

$$p \leqslant q \quad \text{et donc} \quad p^2 \leqslant pq = n \quad \text{c'est à dire} \quad p \leqslant \sqrt{n}$$

Nous pouvons donc généraliser l'observation précédente

Théorème 0 - 3 Si un entier est composé, alors il admet un diviseur premier inférieur à sa racine carrée.

Inspirez-vous de ce résultat pour encore améliorer les fonctions précédentes.
Vous pourrez fabriquer par exemple une fonction qui renvoie le plus petit diviseur d'un nombre (autre que 1 bien sûr) :

```
In [100]: ppd(2**59-1)
Out[100]: 179951
```

Par exemple, il faut 3 secondes pour déterminer le nombre premier qui suit 2^{50} :

```
In [92]: %timeit prochain_premier(2**50)
1 loops, best of 3: 3.45 s per loop
```

Vous pouvez ensuite vous lancer dans l'écriture d'une fonction qui renvoie la décomposition d'un entier en produit de facteurs premiers sachant que nous admettrons le théorème suivant (mais rien ne vous empêche de vous lancer dans une démonstration...) :

Théorème 0 - 4 Tout entier n supérieur à 2 admet une et une seule (à l'ordre près des termes) décomposition en produit fini de nombres premiers

Recherche 0 - 51 Test de Fermat

Pour nos calculs en cryptographie, nous aurons besoin de calculer $a^{p-1} (\mathrm{mod}\, p)$ avec a ou p des « grands » nombres entiers.
Par exemple, que vaut $97^{p-1} (\mathrm{mod}\, p)$ avec $p = 2^{400}$?...
Cela risque de prendre pas mal de temps si nous calculons cette puissance naïvement.
Déterminez une fonction **pow_mod** qui calcule $a^p \mod n$.

```
In [123]: pow_mod(97, 2**400-1, 2**400-1)
Out[123]: 2148126494837635514917182107347665353849377629644858667779989767348429080159664570727704067023680946885559654463293277443
```

Nous avons mis au point une méthode permettant de tester si un entier est premier ou non. Cela marche assez bien pour des petits nombres, mais cette méthode devient impraticable s'il s'agit de tester un entier d'une centaine de chiffres.
Nous avons besoin d'un dernier résultat : le fameux « petit théorème de Fermat ».S'il est qualifié de petit, c'est qu'une conjecture célèbre du même Fermat, restée indémontrée pendant des siècles, s'est accaparée le titre de *Grand Théorème de* Fermat.

$$x^n + y^n = z^n \quad \text{n'a pas de solution dans } \mathbb{N}^3 \quad \text{pour } n > 2$$

Dans la marge d'un manuscrit, Fermat prétendait en avoir trouvé la démonstration mais manquer de place pour l'écrire. Il a pourtant fallu attendre 1994 pour qu'Andrew Wiles le démontre en utilisant des outils surpuissants : l'entêtement d'une multitude de chercheurs a abouti à la démonstration de ce théorème, mais surtout a permis de développer d'importants outils en cryptographie donc en sécurité informatique.
Voici celui qui nous sera utile qui n'est en fait qu'une conséquence du théorème d'Euler...

Théorème 0 - 5 **Petit théorème de Fermat**
Si p est premier, $a^p \equiv a (\mathrm{mod}\, p)$ (ie $a^p - a$ est un multiple de p).

Comment interpréter alors le résultat donné par python juste au-dessus concernant $2^{400}1$?
Créez une fonction **test_f** qui teste si un nombre a vérifie $a^p = a \mod p$.
Que peut-on dire d'un nombre p qui ne passe pas ce test ? Et s'il le passe ?
On va donc faire des tests aléatoires en utilisant **randint** qui renvoie un entier aléatoirement choisi entre deux entiers selon une distribution uniforme.
Pourquoi avoir précisé **randint(11,min(n, 2**63 - 1))** ?

```
from numpy.random import randint

def test_fermat_alea(n) :
    a = randint(11,min(n, 2**63 - 1))
    return test_f(a,n)
```

Créez alors une fonction qui teste si un nombre est probablement premier.

```
In [145]: proba_premier(2**3217 - 1, 100)
Out[145]: True

In [146]: proba_premier(2**4423 - 1, 100)
Out[146]: True

In [147]: proba_premier((2**3217 - 1) * (2**4423 - 1), 10)
Out[147]: False
```

Recherche 0 - 52 Recherches séquentielles et par saut

Un fichier séquentiel informatisé est schématiquement constitué ainsi, chaque fiche contient

- *L'adresse S de la fiche suivante si elle existe sinon* ***nil*** *pour indiquer la fin du fichier.*
- *L'adresse P de la fiche précédente si elle existe sinon* ***nil*** *pour indiquer qu'il n'y a rien en deçà.*
- *Une donnée D_i alphanumérique*
- *Une première adresse* ***début*** *qui est l'adresse de la première fiche.*

Nous supposerons que le fichier est trié suivant l'ordre croissant et pour simplifier l'étude nous poserons $D_i = i$ (la donnée est égale au numéro de la fiche). Le fichier contient N fiches. Nous appellerons temps d'accès à la fiche numéro i le nombre de fiches lues en partant de *début* pour arriver à la fiche i y compris la fiche i. Ainsi le temps d'accès à la fiche 1 est 1.

Recherche séquentielle.

Pour trouver la fiche i ($1 \leq i \leq N$) on part de *début*, on lit les fiches dans l'ordre des numéros jusqu'à obtenir la fiche i. Si on cherche successivement les fiches 1,2,3,$\cdots$, N en repartant de *début* à chaque fois, quel est le temps moyen d'accès à une fiche ?

Recherche par saut.

- *Nous supposerons ici que N est le carré d'un entier, $N = a^2$ ($a \in N^*$). Soit k un entier divisant N et supérieur à 1, $N = kj$.*
- *Nous créons un autre fichier dit de nœuds $N_1, N_2, \ldots, N_j$.*
- *Le départ, noté* **dép**, *contient l'adresse du premier nœud.*
- *Le premier nœud contient l'adresse de la fiche numéro k, la donnée de la fiche numéro k l'adresse du nœud suivant et l'adresse de la fiche $k-1$.*
- *Le deuxième nœud contient l'adresse de la fiche numéro $2k$, la donnée de la fiche numéro $2k$ l'adresse du nœud suivant et de la fiche numéro $2k-1$.*
- .....
- *Le dernier nœud (le $j^{ème}$) contient l'adresse de la fiche numéro N*, et celle de $N-1$, *la donnée de la fiche numéro N et* ***nil***.

Pour chercher une fiche i on part de **dép**, on passe au premier nœud et on regarde la donnée α portée par le nœud. Si $\alpha < i$ on passe au nœud suivant, sinon on passe à la fiche indiquée par le nœud et, si nécessaire, on revient en arrière dans le fichier jusqu'à atteindre la fiche i. Un temps d'accès à une fiche sera le nombre de nœuds parcourus + le nombre de fiches parcourues y compris celle cherchée.

Si on cherche successivement les fiches 1,2,3,$\cdots$,N en repartant de **dép** à chaque fois, on se propose de déterminer le temps moyen d'accès à une fiche et trouver la valeur de k qui minimise ce temps moyen.

1. Relier la recherche par saut avec la ?? page ??
2. Déterminer le temps d'accès à la fiche 1.
3. Déterminer le temps d'accès à la fiche 2 (si $k > 2$).
4. Déterminer le temps d'accès à la fiche k.
5. Déterminer le temps d'accès à la fiche $k-1$.
6. Déterminer le temps d'accès à la fiche $k+1$.
7. Déterminer le temps d'accès à la fiche ik.
8. Déterminer le temps d'accès à la fiche N.
9. Déterminer le temps d'accès à la fiche j.
10. Déterminer le temps total d'accès aux fiches du $i^{ème}$ paquet.
11. Déterminer le temps total d'accès à toutes les fiches.
12. Déterminer le temps moyen d'accès à une fiche en fonction de k et de N.
13. Déterminer la valeur de k qui minimise ce temps moyen.
14. Comparer le temps moyen d'accès à une fiche par une recherche séquentielle et une recherche par saut en prenant N = 10 000 et une unité de temps égale à $10^{-2}s$.

Recherche 0 - 53 Schéma de Horner-Ruffini-Holdred-Newton-Al-Tusi-Liu-Hui...

La méthode que nous allons voir porte le nom du britannique William George Horner (1786 - 1837) mais en fait elle fut publiée presque 10 ans auparavant par un horloger londonien, Theophilius Holdred et simultanément par l'italien Paolo Ruffini (1765 - 1822) mais fut déjà utilisée par Newton 150 ans auparavant et par le chinois Zhu Shije cinq siècles plus tôt (vers 1300) et avant lui par le Persan Sharaf al-Din al-Muzaffar ibn Muhammad ibn al-Muzaffar al-Tusi vers (1100) et avant lui par le Chinois Liu Hui (vers 200) révisant un des résultats présent dans *Les Neuf Chapitres sur l'art mathématique* publié avant la naissance de JC...
Il faut cependant noter que Ruffini l'avait employée en fait comme un moyen de calculer rapidement le quotient et le reste d'un polynôme par $(X-\alpha)$.
Dans toute la suite, un polynôme de dégré n sera représenté par le vecteur de ses coefficients. Par exemple, [1 2 3] correspond au polynôme $1+2x+3x^2$.
Prenons l'exemple de $P(x) = 3x^5-2x^4+7x^3+2x^2+5x-3$. Le calcul classique nécessite 5 additions et 15 multiplications. On peut faire pas mal d'économies de calcul en suivant le schéma suivant :

$$\begin{aligned}
P(x) &= \underbrace{a_n x^n + \cdots + a_2 x^2 + a_1 x}_{\text{on met } x \text{ en facteur}} + a_0 \\
&= \left(\underbrace{a_n x^{n-1} + \cdots + a_2 x}_{\text{on met } x \text{ en facteur}} + a_1 \right) x + a_0 \\
&= \cdots \\
&= (\cdots(((a_n x + a_{n-1})x + a_{n-2})x + a_{n-3})x + \cdots)\, x + a_0
\end{aligned}$$

Ici cela donne $P(x) = (((((3x)-2)x+7)x+2)x+5)x-3$ c'est-à-dire 5 multiplications et 5 additions.
Comparez les complexités au pire du calcul de $P(t)$ pour $t \in \mathbb{K}$. Vous prendrez comme « unité de complexité » les opérations arithmétiques de base : + - *.
Déterminez une fonction `horner(P,t)` qui évalue le polynôme `P` en `t` selon le schéma de Horner.

Recherche 0 - 54

On travaille dans $\mathbb{R}$. Calculer les sommes :

1. $\sum_{i=3}^{6} ij$, $\sum_{i=0}^{4} i^2$, $\sum_{k=0}^{4} k^2$, $\sum_{j=0}^{4} i^2$, $\sum_{i=1}^{50} a_j$
2. $\sum_{i=1}^{n\geq 1} a$, $\sum_{i=h}^{n\geq h} a$, $\sum_{i=1}^{n\geq 1} 3$
3. $\sum_{1\leq i<j\leq 5} ij$, $\sum_{1\leq i<j\leq 5} (i+2j)$

Recherche 0 - 55

On travaille dans $\mathbb{R}$, calculer les sommes $\sum_{i=1}^{n} i$, $\sum_{i=50}^{100} j$, $\sum_{i=h}^{k} i$, $\sum_{i=h}^{k} (i+j)$, $\sum_{i=h}^{k} ij$, $\sum_{i=h}^{k} (\alpha i + j)$

Recherche 0 - 56

On considère un tableau de nombres ayant n lignes et p colonnes. Les lignes sont numérotées du haut en bas et les colonnes de la gauche vers la droite. Le nombre se trouvant sur la $i^{ème}$ ligne et $j^{ème}$ colonne est noté $a_{i,j}$ ou a_{ij}. Par exemple, si $n = 4$ et $p = 3$, le tableau se présente par

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,1}$	$a_{3,2}$	$a_{3,3}$
$a_{4,1}$	$a_{4,2}$	$a_{4,3}$

On note L_i la somme des nombres de la ligne numéro i, C_j la somme des nombres de la colonne j et T la somme de tous les nombres du tableau.

1. Exprimer L_i et C_j à l'aide d'un symbole $\sum$.
2. Exprimer T de deux façons.

Recherche 0 - 57

On reprend les notations de l'exercice précédent avec $n = p = 5$. Que calculent les sommes suivantes ? On pourra donner la solution en grisant les cases du tableau qui correspondent aux nombres intervenant dans les sommes.

1. $\sum_{i=1}^{5} \sum_{j=1}^{5} a_{ij}$
2. $\sum_{j=1}^{5} \sum_{i=1}^{5} a_{ij}$
3. $\sum_{i=1}^{5} \sum_{j=1}^{i} a_{ij}$
4. $\sum_{j=1}^{5} \sum_{i=1}^{j} a_{ij}$
5. $\sum_{i=1}^{5} \sum_{j\geq i} a_{ij}$
6. $\sum_{i+j=6} a_{ij}$
7. $\sum_{1\leq i<j\leq 5} a_{ij}$

Recherche 0 - 58 Tapis de Sierpinski

Monsieur Sierpinski avait ramené d'un voyage en Orient un tapis carré de 1 mètre de côté dont il était très content. Jusqu'au jour où les mites s'introduisirent chez lui.
En 24 heures, elles dévorèrent dans le tapis un carré de côté trois fois plus petit, situé exactement au centre du tapis. En constatant les dégats, Monsieur Sierpinski entra dans une colère noire ! Puis il se consola en se disant qu'il lui restait huit petits carrés de tapis, chacun de la taille du carré disparu. Malheureusement, dans les 12 heures qui suivirent, les mites avaient attaqué les huit petits carrés restants : dans chacun, elles avaient mangé un carré central encore trois fois plus petit. Et dans les 6 heures suivantes elles grignotèrent encore le carré central de chacun des tout petits carrés restants. Et l'histoire se répéta, encore et encore ; à chaque étape, qui se déroulait dans un intervalle de temps deux fois plus petit que l'étape précédente, les mites faisaient des trous de taille trois fois plus petite...

1. Faire des dessins pour bien comprendre la géométrie du tapis troué. Calculer le nombre total de trous dans le tapis de Monsieur Sierpinski après n étapes. Calculer la surface S_n de tapis qui n'a pas encore été mangée après n étapes. Trouver la limite de la suite $(S_n)_{n\geq 0}$. Que reste-t-il du tapis à la fin de l'histoire ?
2. Calculer la durée totale du festin « mitique »...

Recherche 0 - 59

$|x| < 1$ et on note $S_n(x) = \sum_{k=0}^{n} x^k$, $\sigma_n(x) = \sum_{k=0}^{n} kx^k = \sum_{k=1}^{n} kx^k$.

1. Calculer $S_n'(x)$.
2. Calculer $\sum_{k=0}^{+\infty} x^k$.
3. Calculer la limite de $S_n'(x)$ lorsque n tend vers $+\infty$.
4. Exprimer $\sigma_n(x)$ à l'aide de $S_n(x)$.
5. Calculer $\sum_{k=0}^{+\infty} kx^k$.

Recherche 0 - 60

On considère que op est une opération à temps constant.
Combien d'op sont réalisées dans les boucles suivantes :

```
Pour i de 1 à n Faire
| op
FinPour
```

```
Pour i de 1 à n*n Faire
| Pour j de 1 à n*3 Faire
| | Pour k de 1 à n Faire
| | | op
| | FinPour
| FinPour
FinPour
```

```
Pour i de 1 à n Faire
| Pour j de 1 à n Faire
| | op
| FinPour
| Pour k de 1 à n Faire
| | op
| FinPour
FinPour
```

```
Pour i de 1 à n Faire
| Pour j de 1 à n Faire
| | op
| FinPour
FinPour
```

```
Pour i de 1 à n Faire
| Pour j de 1 à n Faire
| | Pour k de 1 à n Faire
| | | op
| | FinPour
| FinPour
FinPour
```

```
Pour i de 1 à n Faire
| Pour j de 1 à 90n Faire
| | op
| FinPour
| Pour k de 40n à 1 Faire
| | op
| FinPour
FinPour
```

```
Pour i de 1 à n Faire
| Pour j de 1 à i Faire
| | op
| FinPour
FinPour
```

```
Pour i de 1 à n Faire
| Pour j de 1 à i Faire
| | Pour k de 1 à j Faire
| | | op
| | FinPour
| FinPour
FinPour
```

```
s ← 0
i ← n
TantQue i > 0 Faire
| Pour j de 0 à i − 1 Faire
| | s ← s + 1
| FinPour
| i ← i // 2
FinTantQue
Retourner s
```

```
s ← 0
i ← 1
TantQue i < n Faire
| Pour j de 0 à i − 1 Faire
| | s ← s + 1
| FinPour
| i ← i * 2
FinTantQue
Retourner s
```

```
s ← 0
i ← 1
TantQue i < n Faire
| Pour j de 0 à n − 1 Faire
| | s ← s + 1
| FinPour
| i ← i * 2
FinTantQue
Retourner s
```