Prise en main.

1. Pour commencer, on tape en ligne de commande:

   `sage -notebook`

   Lancée ainsi, la session Sage se passe dans un navigateur web qui affiche une page appelée `bloc-notes` (ou *worksheet* en anglais). Il se compose de *cellules* (sous la forme de cadres rectangulaires) ou on saisie des commandes et des formules.

   N.b. Au premier démarrage, Sage demande de s'identifier : on pourra choisir l'identifiant `admin` et le mot de passe `sage`.

2. Sage possède un système d'aide en ligne. Pour obtenir d'aide concernant une commande donnée on peut aussi taper directement `commande?` avec un point d'interrogation à la fin. On peut aussi cliquer sur le lien `Help` en haut de la page.

3. Pour lancer un calcul, on valide la saisie avec Shift+Entrée.

4. La commande d'affectation `"="` permet d'assigner une valeur à une variable, par exemple `a=10`. Attention, Sage distingue majuscules et minuscules.

   Pour effacer toutes les variables, on choisi `Restart worksheet` dans le menu `Action` en haut de la page.

5. On peut évaluer plusieurs expressions à la fois en les mettant dans le même cadre. On passe à la ligne avec la touche Entrée. Ceci est utile pour créer des boucles ou des petits programmes. On peut aussi mettre deux expressions dans une même ligne en les séparant avec un point-virgule (`";"`) ou un virgule. Attention, seuls les résultats de calculs dans la dernière ligne sont affichés.

6. Expressions mathématiques. L'encodage des expressions mathématiques se fait d'une manière habituelle.

   (a) Exemple. Le produit algébrique $(2x+3)(3x+4)$ se traduit comme `(2*x+3)*(3*x+4)`. On remarque que l'opérateur de multiplication `"*"` est obligatoire.

   (b) Exemple. La fonction puissance x^2 s'écrit `x^2`. Sur certaines machines il est difficile d'obtenir le symbole de l'accent circonflexe `^`. On peut alors utiliser l'écriture alternative : `x**2`.

7. On dispose de certaines constantes prédéfinies, par exemple `pi`, `i` (l'unité imaginaire) et `e` ($= exp(1)$).

   Attention : `pi`, comme les autres constantes et variables libres, est un symbole qui est traité *formellement*. Par exemple, `pi^2` renvoie juste `pi^2`, une autre constante formelle. Pour obtenir une valeur numérique approchée d'une telle constante (ou d'une expression quelconque) on utilise la commande `N` ou `n`, par exemple `N(pi)`, `N(log(9) + sqrt(3))`. Pour plus de chiffres significatifs, on utilise le paramètre `digits`, par exemple : `N(pi,digits=30)`.

   Attention : Techniquement, Sage ne protège pas les constantes prédéfinies et, par exemple, `pi = 10` sera accepté. On fera donc attention de ne pas changer `pi`, surtout si on veut calculer une circonférence ! On s'abstiendra aussi d'utiliser la variable `i` dans une boucle si on veut faire un calcul avec des nombres complexes.

   Attention : De même, il est possible d'affecter `N=10` et alors `N(pi)` ne marche plus !
   On s'interdit donc d'utiliser `N` comme une variable !

8. Un commentaire se met après le symbole dièse `#`. Tout ce qui est après ce symbole sur la ligne n'est pas interprété par le moteur mathématique.

9. Pour insérer une cellule, on positionne la souris juste au dessus d'une cellule existante. Une ligne horizontale apparaît. Si on clique là-dessus, une nouvelle cellule de calcul est créée. Si on clique tout en appuyant la touche Shift, on insère une nouvelle cellule de type texte, entièrement consacrée aux commentaires (un titre, des explications...).

   Pour effacer une cellule, on efface d'abord son contenu, puis on appuie la touche ← (retour arrière)

10. En cochant la case "Typeset" en haut de la page on impose mise en forme traditionnelle des formules mathématiques.

11. L'interruption d'un programme se fait par le menu `Action` puis `Interrupt`.

Exercice 1.

La fonction `log` avec l'option `base` permet de calculer le logarithme en base donnée, par exemple `log(123,base=10)`). Pour calculer la partie entière d'un nombre, on utilise la fonction `floor`.

a- Calculer le nombre de chiffres de factorielle 2012 (`factorial`). [1]

b- Calculer la longueur de $2^{43112609} - 1$. [2]

N.b. C'est le plus grand nombre premier connu à ce jour, sa découverte en 2008 valait un prix de 100 000 dollars.

Exercice 2.

Evaluer π à une précision de 1 000 000 chiffres. Mesurer le temps de calcul (cf. l'antisèche).

Exercice 3. *Boucle* `while`

Si un nombre est pair, on le divise par 2. S'il est impair, on le multiplie par 3, on lui ajoute 1 et on divise le résultat par 2. Ainsi, en commençant avec n=10, on obtient la suite suivante : $10 \to 5 \to 8 \to 4 \to 2 \to 1$. Une fois tombée sur 1, la suite tourne en rond : $1 \to 2 \to 1 \to ...$

Selon la *conjecture de Syracuse*, quelle que soit la valeur initiale n choisie, la suite finira toujours dans cette boucle. Plusieurs projets visent de vérifier cette conjecture à l'aide d'un ordinateur (voir d'un réseau d'ordinateurs). On se propose ici d'étudier quelques propriétés basiques.

Ecrire une procédure `syracuse(n)` qui étant donné un entier `n` retourne la longueur de la suite de Syracuse commençant par `n` (la suite se termine par la première occurrence de 1). Pour vérifier si un nombre est paire, on pourra utiliser l'opérateur *modulo* `"%"`, c'est-à-dire le reste de la division euclidienne; exemple : 12%2 renvoie 0.

Tester : `syracuse(10)` = 6, `syracuse(100)` = 19.

Exercice 4. *Boucle* `for`.

Trouver la longueur maximale de la suite pour la valeur initiale de `n` entre 10 et 10 000. Trouver la valeur de `n` qui produit la suite de cette longueur.

Exercice 5. *Listes*

Ecrire une procédure `syracuse_liste(n)` qui renvoie une liste contenant la suite de Syracuse.

Exercice 6. *Un peu de graphisme*

1. Visualiser l'évolution de la suite de Syracuse pour n = 700, 1500 et 10000. On pourra utiliser `list_plot` avec l'option `plotjoined` (cf. l'antisèche).

   Laquelle valeur de `n` donne lieu à la suite la plus longue?

2. Visualiser les longueurs de la suite en fonction de n allant de 10 à 10000. On pourra utiliser `list_plot` avec l'option `size=1`.

[1] rép.: 5776
[2] rép.: 12978189

Exercice 1. *Développement décimal*

Soit $x \in \mathbb{R}$. Posons $x_n = 10^{-n}E(10^n x)$ (la valeur décimale approchée à 10^{-n} près par défaut), puis $a_0 = E(x)$ et $a_n = E(10^n x) - 10E(10^{n-1}x)$, $n = 1, 2, \ldots$ (les coefficients du développement décimal). $E(x)$ représente ici la partie entière de x (cf. `floor(x)` en Sage).

1. Soit $x = 7/13$. Calculer sa valeur décimale approchée à 10^{-k} près pour `k` $= 3$ et 8; comparer avec la valeur approchée donnée par `N(7/13)`.

2. Soit $x = \pi$. Calculer x_{51} (pour l'affichage, on pourra utiliser l'option `digits=60` de la commande `N`). Comparer avec `N(pi, digits=52)` (pour la même longueur, il faut ici `digits=52` car la commande `N` compte aussi les chiffres avant la virgule).

   Est-ce la même chose? Faire d'autres approximations et expliquer (deviner) la différence entre x_n et `N(x,digits=n)`.

3. Ecrire une procédure `dec(x,n)` qui renvoie le coefficient a_n défini ci-dessus.

   Soit $x = \pi$. Calculer la liste de coefficients a_n, $n = 0, 1, \ldots, 51$. Comparer avec x_{51}.

   La formule $x_n = a_0 + \sum_{k=1}^{n} a_k 10^{-k}$ est-elle vérifiée?

Exercice 2. *Développement en base quelconque.*

1. En s'inspirant de la procédure `dec` de l'exercice précédent, écrire une procédure `binaire(x,n)` qui étant donné un réel x calcule le coefficient a_n de x en base en base 2.

2. Calculer la liste des coefficients $a_1, \ldots a_{53}$ du développement binaire de π (ce qui vaut $x_{53} - 3$).

   Afficher ensuite la valeur de cette approximation en base 10. Combien de chiffres après la virgule sont corrects dans le développement décimal ? On peut oublier ici la partie entière qui n'était pas prise en compte dans le développement binaire.

3. Ecrire une procédure `base(x,n,b)` qui calcule le coefficient a_n en base `b`. Dans toute base $b = 2, 3, \ldots, 9$, calculer et stocker dans une liste `L` le nombre de chiffres exacts dans le développement décimal étant donné x_{53} en base b.

   Visualiser la liste `L` ("l'efficacité relative" de bases de 2 à 10).

   Les résultat etait-il prévisible? Comment calculer 'directement' la liste `L` (c.a.d. sans calculer les coefficients a_n dans toutes les bases)? On mettra la réponse dans la feuille de calcul.

Exercice 3. *Face cachée des floats.*

Dans cet exercice on se propose de calculer 0.1 + 0.2 à l'aide d'une machine (et comparer avec un calcul traditionnel sur papier).

1. Effectuer un calcul direct : taper simplement 0.1 + 0.2.

   Le résultat rassurant n'est qu'une apparence quelque peu trompeuse. En effet, il suffit de taper N(0.1 + 0.2, digits=25) pour voir un autre résultat.

   Lequel est alors 'correct'? Sans doute le deuxième est plus précis. Mais d'où vient la différence? Puis quel principe permet à l'ordinateur (ou à une calculette) d'afficher alors un résultat simplifié pour qu'il soit 'mathématiquement correct'?

   Dans la suite, on répondra à ces questions en analysant les détails de ce calcul.

2. Stocker les valeurs 0.1 et 0.2 dans deux variables (`a` et `b`) en imposant le type de données `float` (`a = float(0.1)`). Ceci permet de faire calculs directs "à la calculette" (c.a.d. directement avec la machine, sans algorithmes supplémentaires, calculs symboliques ou encore sans conversion de données de Sage).

   Calculer `c = a + b`.

3. Afficher (directement et sans la commande `N`) les valeurs de `a,b` et `c`. Est-ce que `a=0.1` et `b=0.2`? Est-ce qu'on a l'égalité *mathématique* $a + b = c$?

   Dans la suite on se propose d'expliquer ces résultats.

4. Afficher maintenant 'vraies' valeurs de variables `a` et `b` à l'aide de la commande `N` avec le paramètre `digits` (trouver une valeur convenable pour ce dernier paramètre).

5. Développer 1/10 et 2/10 en base 2 à, par exemple, 50 chiffres (cf. l'exercice 1.3). Dans cette question, on utilisera la notation rationnelle exacte ("1/10" et non "0.1"; sinon tout au début on perd l'information sur la valeur exacte!).

   Le développement binaire est-il fini?

6. *Facultatif.* Dans cette question on retrouve les 'vraies' valeurs de a et de b. D'habitude, l'ordinateur ou logiciel dit "64 bits" (ou "double précision") représente un float dans (0,1) sous forme

$$\frac{J}{2^K} \qquad (*)$$

   où J et K sont deux entiers. Dans cette représentation J est appelé la *mantisse* et K l'*exposant*. Typiquement, la mantisse est stockée sur 53 bits.

   (a) Tronquer les développements binaires obtenus dans la question précédente pour qu'il contienne exactement 53 chiffres à partir de la première occurrences du "1". La longueur du développement ainsi obtenu correspond à l'exposant K dans la représentation (*). Attention, ce n'est pas forcément le même nombre pour 1/10 et 2/10.

   Ainsi on obtient un nombre float proche de la valeur exacte 1/10 (ou, respectivement, 2/10).

   (b) Calculer la valeur approchée de 1/10 (respectivement 2/10) en base 10 à partir de son développement binaire tronqué (cf. l'exercice précédent), stocker les résultats dans la variable `a_defaut` (respectivement `b_defaut`).

   (c) Calculer la différence `a_defaut-1/10` et `b_defaut-2/10`.

   (d) Calculer la valeur float approchée par excès `a_exces = a_defaut +` 2^{-56} et `a_exces = a_defaut +` 2^{-55} (pourquoi on a ici des exposantes différents)?

   (e) Choisir la valeur float approchée la plus proche à la valeur exacte 1/10 (respectivement 2/10).

   Comparer avec les valeurs affichées dans la question 4.

   N.b. Cela explique les 'anomalies' dans les valeurs de `a` et `b`. On voit que les erreurs d'arrondi se glissent dans le calcul bien avant qu'on commence même à additionner les deux nombres.

7. Additionner les 'vraies' valeurs de `a` et `b` et afficher leur somme en base 10 (c.a.d. à l'aide de `N`). Remarquer que cela n'explique *pas* entièrement la différence entre `c` et 3/10.

   On ne discutera pas ici les détails de l'addition. On remarque seulement qu'il s'agit d'une perte de précision supplémentaire due au fait que les 53 bits de mantisse ne sont pas dans la même position dans le développement binaire de 1/10 et 2/10.

8. On a identifié quelques inévitables difficultés qui perturbent le résultat d'une addition de deux nombres réels (ou floats).

   Quel procédé permet alors de cacher le dessous de calculs et afficher des résultats 'mathématiquement corrects'? Formuler une règle, étant donné que l'ordinateur dispose 53 bits pour faire les calculs (une réponse se trouve au verso).

9. Proposer des valeurs de floats `a` et `b` dans]0,1[pour que le calcul de `a+b` par ordinateur soit exact. Vérifier.

Exercice 4. *Surprises des floats.*

Dans l'exercice précédent on a vu que pour afficher des résultats 'mathématiquement corrects', l'ordinateur ou cache les 'vraies' valeurs dont il utilise dans ses calculs. C'est d'ailleurs une convention usuelle adoptée par des calculettes et des logiciels (matlab, Sage, xcas...). Une différence entre la valeur stockée dans une variable et la valeur affichée peut tout de même provoquer certaines surprises.

1. Ecrire une procédure `racine_cubique(n)` qui retourne la racine cubique de `n`.

   Attention : pour s'assurer que le logiciel effectue bien le calcul numérique et non *symbolique*, on utilisera la formule suivante : `n**(1.0/3)`.

   Tester avec quelques valeurs.

2. Calculer `racine_cubique(2097152)` et la partie entière (`floor`) de `racine_cubique(2097152)`.

   Expliquer en utilisant les 'vraies' valeurs des nombres en question.

Exercice 5. *Les floats dans la banque.*

Notre banquier, qui possède la licence de mathématiques financières, nous propose un plan d'épargne un peu plus sophistiqué que d'habitude. Le contrat de durée déterminée 20 ans prévoit que l'année k ($k = 1, 2, \ldots, 20$) on multiplie le capital par k et on prélève 1 euro pour frais de gestion (la première année on perd donc 1 euro). Le placement initial est $e - 1$ euros (ici et en Sage $e = \exp(1)$).

1. On suppose que la comptabilité de banque se fait en arrondissant à 2 chiffres après la virgule. Pour simuler cela on pourra définir le capital initial à l'aide de la commande `round(e-1,2)`. Etes-vous prêts à accepter le contrat (sachant que le client doit payer si le solde s'avère négatif)? Faites une simulation.

2. Le banquier insiste que le calcul soit fait avec un ordinateur ultramoderne qui tient compte de 12 chiffres (c.a.d. on commence par `round(e-1,12)`). Etes-vous toujours d'accord?

3. Essayer d'autres arrondis. Comparer avec une valeur approchée de $e - 1$ à 30 chiffres et formuler une règle qui permet de dire rapidement si le "contrar à n chiffres" après la virgule est intéressant pour la banque ou pour le client.

Exercice 6. *Valeur approchée de π*

On rappelle que $\sin(x) \sim x$ lorsque $x \to 0$. En particulier, $\lim_{n\to 0} n \sin(\pi/n) = \pi$.

On peut facilement démontrer la formule trigonométrique suivante

$$\sin(\alpha/2) = \frac{1}{2}\sqrt{2\left(1 - \sqrt{1 - \sin^2\alpha}\right)}, \quad \alpha \in (0, \frac{\pi}{2}). \qquad (✠)$$

1. En itérant la formule (✠), calculer $\sin(\pi/2^k)$ pour $k = 3, 5, 10, 20, 25, 28$ et 30.

2. Pour les mêmes valeurs de k, en déduire une valeur approché de π. Que dire de cette valeur pour $k = 28$ et 30 ? Comment expliquer ces résultats ? On notera la réponse dans la feuille de calcul.

Exercice 7. *Amplification d'erreurs.*

Soit

$$I_n = \int_0^1 \frac{x^n}{10+x}.$$

1. Calculer une valeur approchée de I_0 en utilisant la commande `numerical_integral` (exemple de la syntaxe : `numerical_integral(sin(x),0,pi)`; cette commande renvoie une valeur approchée de l'intégrale et une estimation de l'erreur).

2. On admet ici la relation de récurrence suivante :

$$I_n = \frac{1}{n} - 10I_{n-1}, \qquad n = 1, 2, \ldots.$$

   En partant d'une valeur approchée de I_0, calculer I_{35} à l'aide de cette formule.

3. Comparer avec la valeur de I_{35} obtenue directement à l'aide de `numerical_integral`. Quel valeur est 'correcte' ? Comment expliquer la différence?

Exercice 8. *Distributivité $a(b+c) \neq ab+ac$.*

1. Posons $u_0 = 0.23$ et $u_{n+1} = 4u_n(1-u_n)$ (cette suite est appelée la *suite logistique*). Calculer u_{80}.

2. Posons $v_0 = 0.23$ et $v_{n+1} = 4v_n - 4v_n^2$. Evidemment, il s'agit de la même suite que dans la question précédente. Calculer v_{80}.

3. Visualiser les deux suites (`list_plot`).

   Quelle valeur u_{80} ou v_{80} est correcte? Pour répondre, refaire ces calculs avec une meilleure précision. Mettre la réponse et une explication du phénomène observé dans la feuille de calcul.

Exercice 9. *Facultatif. Effet papillon.*

Tracer pas à pas 100 termes de la suite $d_n = u_n - u'_n$, où (u_n) est la suite logistique partant de $u_0 = 0.23$ et (u'_n) celle partant de $u'_0 = u_0 + 10^{-10}$. Pour éliminer les erreurs d'arrondi, on choisira une précision de calculs supérieure (p.ex. `digits=500`).

Les deux suites sont-elles proches l'une de l'autre?

Diminuer la perturbation initiale à 10^{-15}. Les valeurs u_{100} et u'_{100} se rapprochent-elles?

N.b. Par ailleurs, la suite logistique sert à construire des modèles du réel (par exemple, l'évolution d'une population). Pensez vous qu'une perturbation d'ordre 10^{-15} serait détectable? Peut-on donc faire des prévisions à long terme? Un simple battement d'ailes d'un papillon peut-il déclencher une tornade à l'autre bout du monde ?

Exercice 10. *Facultatif. Calcul exact et calcul approchée.*

Écrire deux fonctions calculant le n-ième terme de la suite définie par

$u_0 = 11/2$, $u_1 = 61/11$, $u_{n+1} = 111 - 1130/u_n + 3000/(u_n u_{n-1})$, pour $n \geq 1$.

La première par un calcul exact; la deuxième en calculant avec d chiffres significatifs.

Comparer les résultats obtenus avec $d = 10$, 20, 30 et pour n variant entre 5 et 40 par exemple.

Supposons que la suite converge. Quelle est la limite de la suite (u_n) ? Pour chercher des points fixes on pourra utiliser `solve`, p.ex `solve(2*x+5==1,x)`

Exercice 11. *Nombres normaux.*

Un nombre est appelé *normal en base b* si la fréquence d'apparition de tout chiffre dans son développement en base b est $1/b$, la fréquence de tout paire de chiffres est $1/b^2$ et, d'une manière générale, la fréquence de tout n-uplet (c.a.d. séquence de longueur n) est b^{-n}. Un nombre est *normal* s'il est normal en toute base b.

Remarques:

- Les nombres rationnels ne sont pas normaux en aucune base (pourquoi?).
- On sait[1] que presque tout nombre est normal (au sens que l'ensemble des exceptions est de longueur infiniment petite).
- Voici un exemple d'un nombre normal en base 10 : 0,12345678910111213141516171819202 1... (on ne sait pas s'il est normal dans les autres bases).
- D'après une conjecture de Borel, tout nombre algébrique[2] irrationnel est normal.
- On ne connait aucun exemple "naturel" d'un nombre normal ; en particulier on ne sait pas si π, e, $\sqrt{2}$ etc... sont normaux. On ne sait même pas si la fréquence d'un seul chiffre dans le développement décimal de π est $1/10$.

Cependant on croit bien que par exemple π est un nombre normal. Y contribuent des expériences comme la suivante.

On se propose ici de déterminer la fréquence de chaque chiffre dans le développement décimal de π à K chiffres après la virgule (on pourra prendre $K = 100$ puis 1000 et 100000).

1. Construire la liste de coefficients du développement décimal de π de longueur K.
2. Compter le nombre d'apparitions de chaque chiffre et stocker les résultats dans une liste `L` (`L[0]` représentera le nombre de 0, `L[1]` le nombre de 1 etc.).
3. Visualiser le résultat à l'aide de `bar_chart(L)`. Que voit-on lorsque K varie de 100 à 100000 ?
4. *Facultatif.* Déterminer et visualiser les fréquences de tout paire de chiffres. De même pour tout triplet.
5. *Facultatif.* Déterminer et visualiser les fréquences de tout chiffre en base b variant de 2 à 9.

Exercice 12. *Facultatif. π visualisé*

La commande `matrix(10)` crée une matrice carré de taille 10. Remplir une telle matrice avec des coefficients du développement de π en base 10 (ou en base 2). Visualiser la matrice à l'aide de `matrix_plot`. On pourra utiliser l'option `cmap='hot'`. De même pour une matrice d'une taille supérieure (50, 100,...).

[1] grace à un théorème d'Emile Borel, 1909

[2] nombre algébrique est un nombre réel qui est racine d'un polynôme à coefficients entiers

[2] Parce que $2^{-53} \approx 10^{-16}$, on peut penser que 15 chiffres de la représentation décimale sont corrects. Il suffit donc de couper l'affichage à 15 ou 16 chiffres après la virgule.

1 Congruences

Exercice 1. *Tables d'addition et de multiplication.*

Calculer et visualiser les table d'addition et de multiplication de $\mathbb{Z}/100\mathbb{Z}$. On pourra utiliser la commande `matrix(100)` qui crée une matrice de taille 100 puis `matrix_plot` avec l'option `cmap='hot'` pour couleurs.

Exercice 2. *Equations congruentielles.*

Ecrire des boucles qui "résolvent" les équations : $15x = 21 \mod 28$ puis $36x = 54 \mod 42$, c'est à dire qui délivrent les entiers entre 0 et 27, puis 0 et 41, satisfaisant à ces équations. Pour calculer $a \mod b$, pensez à l'opérateur `%`.

Exercice 3. *Inversion modulo p.*

Ecrire une petite procédure `inverse(n,p)` qui calcule l'inverse d'un entier n modulo un entier p. Ensuite vérifiez en calculant le produit `n*inverse(n,p)` modulo `p` avec quelques valeurs de `n` et de `p`.

Existe-t-il l'inverse de 18 modulo 100?

Trouver l'ensemble des nombres (≤ 1000) 'qui n'ont pas l'inverse modulo 1001. Expliquer (cf .`gcd`).

Exercice 4. *Pascal et Sierpinski.*

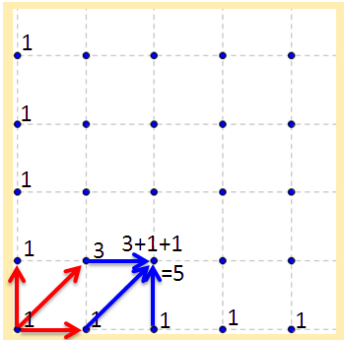
Les coefficients binomiaux $\binom{n}{k}$ sont donnés par `binomial(n,k)`.

1. Afficher 10 lignes du triangle de Pascal. On pourra utiliser une matrice `A=matrix(10)` et remplir les composantes a_{nk} pour $k = 0, ..., n$ (ce qui donne une matrice triangulaire inférieure).
2. Calculer et visualiser à l'aide de `matrix_plot` avec (`cmap='hot'`) 512 lignes du triangle de Pascal modulo 2 (on pourra inverser les couleurs en affichant la matrice `-A`).
3. De même modulo 3, 6, 30, 210, 211.

Pour en savoir plus, écoutez la conférence "Quoi de neuf sur le triangle de Pascal" :
`http://videocampus.univ-bpclermont.fr/?v=QbDmUsT2VsSf`

Exercice 5. *Facultatif. Nombres de Delannoy.*

Imaginons une grille de points de $\mathbb{N} \times \mathbb{N}$. A chaque point (a, b) on associe le nombre, noté $D(a, b)$, de trajectoires de $(0, 0)$ à (a, b) composées de trois mouvements *croissants* comme indiqué sur le dessin ci-contre. On voit facilement que $D(a, b)$ vérifie les relations suivantes :



- $D(a, 0) = 1$ pour tout $a \in \mathbb{N}$
- $D(0, b) = 1$ pour tout $b \in \mathbb{N}$
- Si $a, b > 0$, alors il y a trois façons d'attendre le point (a,b) à partir de ces voisins (cf. l'image ci-contre). Cela donne la relation de récurrence

$$D(a, b) = D(a - 1, b) + D(a, b - 1) + D(a - 1, b - 1).$$

1. En utilisant ces relations, calculer et afficher une matrice de nombres $D(i,j)$, $i,j = 0,1,\ldots,9$.
2. Calculer une matrice de nombre $D(i,j)$ modulo 3, $i,j = 0,\ldots,300$. Visualiser (`matrix_plot`).

Exercice 6. *Cartes bancaires.*

On rappelle la formule de Luhn pour calculer la clef pour un numéro d'une carte bancaire : si $a_1, a_2, \ldots, a_{15}$ sont les chiffres significatifs, on calcule

$$s = \sum_{i=1}^{7} a_{2i} + \sum_{j=0}^{7} b_{2j+1},$$

où b_{2j+1} est l'entier à un chiffre égal à la somme des chiffres dans l'écriture décimale de $2 \times a_{2j+1}$; la clef a_{16} est alors le complément à 10 du reste de la division euclidienne de s par 10 (si le reste est nul, alors la clef $a_{16} = 0$).

1. Ecrire une procédure `clef(L)` qui étant donnée une *liste* de 15 chiffres renvoie la clef de Luhn.
2. Etant donné le mot-clé "carte bancaire", un moteur de recherche d'images revoie quelques réponses où on peut reconnaitre le numéro de la carte. Vérifier si les numéros suivants sont plausibles :
   5131 7512 3456 7890 4973 0123 4567 8901 4556 5198 4996 8739
   5358 4101 8163 9970 2973 8162 9273 9723 4275 3156 0372 5493
   5076 1923 7003 6150.
3. Dans un magasin, un voleur a remarqué le numéro de la carte de crédit d'un client qui passait aux caisses. Malheureusement pour le voleur, le client à caché les 3 premiers chiffres avec son doit. Les 13 derniers chiffres étant 9 7139 0441 1103, combien de possibilités doit essayer le voleur pour faire un achat sur un site internet à l'aide du vrai numéro de cette carte? On suppose ici que le voleur ne connait pas la structure de ce numéro, ce qui pourrait donner plus d'informations, mais il connaît bien la formule de Luhn.
4. *Facultatif.* Même question avec les trois *derniers* chiffres cachés (donc la clef !), les treize premiers chiffres étant 4024 0071 5066 9.

2 Codage

En Sage, une lettre ou un mot se représente par une chaine de caractères entre apostrophes (`'a'`,`'un mot'`, `'ceci est une phrase'`, etc.).

Etant donnée une lettre, la commande `ord` renvoie son code ASCII (p.ex. ord('A') renvoie 65). Etant donné un nombre entre 65 et 90, on retrouve la lettre correspondant à l'aide de `chr` (p.ex. `chr(67)` renvoie `'C'`). Voici quelques opérations concernant des lettres et des mots. Pour parcourir des lettres d'un mot (ou d'une phrase) on peut écrire :

```
for k in 'unmot':
   print k
```

Pour réunir deux lettres (ou deux chaînes de caractères) on utilise l'opérateur `"+"`, par exemple :

`a = 'mot1'; b = 'mot2'; c = a+b`

Alors `c` vaut `'mot1mot2'`.

Exercice 7. *Codage basique.*

1. Ecrire une procédure `code_une(lt)` qui étant donnée une lettre (majuscule) `lt` renvoie son code entre 0 et 25 (A correspond à 0, B à 1 etc...). Tester.
2. Ecrire une procédure réciproque `decode_une(c)` qui étant donnée un nombre entre 0 et 25 renvoie la lettre correspondante. Tester.
3. Ecrire une procédure `code_mot(m)` qui étant donné un mot `m` renvoie une liste de codes. Ecrire une procédure `decode_mot(L)`. Tester si les deux procédures sont bien réciproques.

Exercice 8. *Code de César.*

1. Ecrire une procédure `cesar(mot, d)` de codage de César qui code un mot `m` avec un décalage `d`. La fonction `map` pourra être utile.
2. Ecrire une procédure de décodage `decesar`.
3. Coder (mot par mot) "CE MESSAGE EST CONFIDENTIEL" avec un décalage par 10. Décoder le résultat. On pourra créer une liste de mots et utiliser la fonction `map`.
4. Trouver (par une recherche exhaustive) le décalage utilisé pour code la phrase suivante: "OH FUBSWDJH HVW ODUV GH FRGHU XQ PHVVDJH".
5. Decoder "BG FTMAXFTMBVL RHN WHGM NGWXKLMTGW MABGZL RHN CNLM ZXM NLXW MH MAXF" (John von Neumann).

Exercice 9. *Codage affine.*

1. Ecrire une procédure `affine(mot, a,b)` de codage affine $f(n) = an + b \pmod{26}$.
2. Ecrire une procédure réciproque `de_affine`. Pour résoudre une équation congruentielle $ax + b = m$, on pourra utiliser la commande `xgcd(a,26)` qui renvoie le triplet `(g,u,v)` tel que $\text{pgcd}(a, 26) = g = u \cdot a + 26v$. On en déduit l'inverse de a mod 26 et, par conséquent, la solution de notre équation.
3. Coder "CE MESSAGE EST CONFIDENTIEL" avec $f(n) = 5n + 7$. Décoder. La fonction `map` pourra être utile.
4. Coder 'PROBLEME' avec $a = b = 2$. Décoder. D'où vient le problème?
5. Décoder le mot "KYBIX" étant donné qu'il a été codé avec un codage affine avec $a < 10$ et $b < 10$. Utiliser une contrainte concernant une de ces variables pour limiter la recherche exhaustive.
6. Etant donné $a = 15$, décoder "LP NVP UJVR YCJAVXRJUR L PRG AP QH LJFYCPIPURXJU" (William Thurston, medaille Fields, à propos de la recherche en mathématiques).

Exercice 1. *Crible d'Erathostène.*

Construire la liste P des nombres premiers dans $[2, n]$ de deux manières différentes :

1. en utilisant `is_prime`
2. en criblants la suite des entiers x, $2 \leq x \leq n$: soit k le premier entier non criblé (initialement $k = 2$); alors on rajoute k à la liste et on supprime dans la suite tous les multiples de k.

   Pour realiser cet algorithme en Sage, on pourra créer une liste auxiliaire `L` de taille $n+1$ remplie de 1; alors la "suppression" d'un nombre m se fait par l'affectation `L[m] = 0`.

   Astuce : pour créer une liste de taille 20 remplie de 1, on peut "multiplier une liste unité" : `L = [1]*20`.

Comparer avec la méthode précédente. On pourra utiliser `time` pour chronométrer. Penser aux "grandes" valeurs de n.

Exercice 2. *Répartition des nombres premiers.*

1. *Comparaison de fonctions : deux fonctions équivalentes sont-elles toujours "proches"?.*

   Soient $f(x) = x^3$ et $g(x) = (x-1)^3$. Représentez graphiquement $f(x)/g(x)$ pour "grandes" valeurs de x.

   Calculer la limite de ce rapport (exemple : `limit(1/x,x=infinity)`). Est-ce que $f \sim g$ au voisinage de l'infini?

   Etudier de la même manière le comportement de la différence $f(x) - g(x)$ lorsque $x \to \infty$. Est-ce que $\lim_{x\to\infty}(f(x) - g(x)) = 0$?

2. En utilisant la procédure `crible`, compter les nombres premiers inférieurs à 1 million. Comparer avec la commande `prime_pi`.

   Dans la suite par $\pi(x)$ on note le nombre de nombres premiers inférieurs à x. Visualiser la fonction $\pi(x)$ pour `x` entre 1 et 100 puis entre 1000 et 2000. Attention : pour des raisons techniques, la commande `plot(prime_pi(x),x,1000,2000)` ne marche pas bien. On utilisera donc la syntaxe *sans variable explicite* : `plot(prime_pi,1000,2000)`.

3. Dans la suite on se propose de découvrir une formule mathématique qui donne une valeur approchée de $\pi(x)$. On va suivre la démarche de Gauss (sachant que celui dernier calculait "à la main" !).

   Soit $P(x)$ la proportion $P(x) = \pi(x)/x$. Posons

   $$R(x) = \frac{1}{P(10x)} - \frac{1}{P(x)}.$$

   Ecrire une procédure `R` qui renvoie une valeur numérique approchée (cf. `N`) de la fonction R.

   Calculer $R(x)$ pour quelques valeurs de x. Visualiser la courbe représentative de $R(x)$.

   Que se passe-t-il lorsque x devient grand, p.ex. $x = 10^4$, 10^5, 10^6, 10^7, ...?

4. Déterminez les mêmes écarts pour la fonction $\log(x)$, c.a.d. calculer $L(x) = \log(10x) - \log(x)$ (ici et en Sage, $\log x$ signifie le logarithme naturel). Dans un premier temps on pourra calculer les valeurs approchées de $L(x)$ pour quelques valeurs de x. Puis on retrouvera la valeur algébrique exacte sur papier (c.a.d. sans Sage).

   En déduire une approximation à $\pi(x)$ par une fonction (assez) élémentaire.

   Notons que Gauss a donné cette approximation en 1792, à l'age de 15 ans. Il n'avait pas de preuve de cette constatation (il n'a donc pas publié ses résultats).

5. Soit $f(x) = x/\log(x)$. Représentez graphiquement sur le même graphique $\pi(x)$ et $f(x)$. Les fonctions vous semblent-elles proches l'une à l'autre?

   Tracez les graphes de $\pi(x)/f(x)$ et $\pi(x) - f(x)$ (séparément). On pourra prendre l'intervalle $[10^5, 10^6]$. Les fonctions sont-elles équivalentes? La différence s'annule à l'infinie?

6. A l'age de 64 ans Gauss a conjecturé que $\pi(u)$ est bien approché par la surface sous la courbe $x \to \frac{1}{\log x}$ entre 2 et u. On rappelle que cette surface se calcule à l'aide de l'intégral; on notera

$$g(u) = \int_2^u \frac{1}{\log x} dx$$

   (cf. `numerical_integral(f,a,b)`). Calculez $g(10^4)$, $g(10^5)$, $g(10^6)$.

   Refaire la question précédente avec g au lieu de f. Superposer des graphiques correspondants.

   Quelle approximation est la meilleure ?

7. La fonction $\pi(x) - g(x)$ est-elle monotone ? bornée ? positive ?

   N.b. En regardant nos graphiques on pourra conjecturer que pour tout $x \in \mathbb{R}$, on a $\pi(x) < g(x)$. Cependant on sait que pour certain $x_0 \in \mathbb{R}$ on a $\pi(x_0) > g(x_0)$. La valeur de x_0 est aussi grande (d'ordre de grandeur 10^{300}), qu'il est pratiquement impossible de calculer explicitement $\pi(x_0)$. Ceci montre bien certaines limites de l'expérience mathématique sur l'ordinateur.

Exercice 3. *Nombres de Carmichael.* On rappelle le petit théorème de Fermat :

Si p est premier, alors pour tout $a \in \mathbb{Z}$, on a $a^p \equiv a \mod p$.

1. A l'aide de Sage vérifier que pour $p = 1105$ la réciproque du petit théorème de Fermat est fausse : pour tout $a \in [2, p-1]$ tel que $\text{pgcd}(a, p) = 1$ on a $a^{p-1} = 1 \mod p$ et pourtant p est, évidemment, composé. Pour trouver le $\text{pgcd}(a, p)$ on pourra utiliser `gcd`.

   Les nombres *composés* ayant cette propriété sont appelés *nombres de Carmichael*.

2. Ecrire une procédure `cm(n)` qui, étant donné un entier `n` retourne 1 si `n` est un nombre de Carmichael et retourne 0 sinon.

3. Calculer la liste des nombres de Carmichael inférieurs à 20 000. Combien y en a-t-il?

Exercice 4. *Conjecture de Goldbach.*[1]

Selon la conjecture de Goldbach, tout nombre entier pair (sauf 2) se représente comme la somme de deux nombres premiers. On s'autorise ici à utiliser le même nombre premier deux fois (par exemple 6=3+3). C'est l'un des plus vieux problèmes non résolus de la théorie des nombres et des mathématiques.

1. Ecrire une procédure `gold(n)` qui renvoie le *nombre de décompositions possibles* d'un entier pair `n` en somme de deux nombres premiers; par exemple, `gold(16)` renvoie 2 car 16=11+5=13+3.

   La conjecture de Goldbach revient à dire que pour tout n entier pair, `gold(n)`≥ 1.

2. Calculer une liste des valeurs de `gold(n)`, `n` pair variant de 4 à 100. Visualiser (`list_plot`).

   De même pour n variant de 4 à 1000 puis de 4 à 5000 (c'est ce qu'on appelle la *la comète de Goldbach*).

Exercice 5. *Nombre de taxicab. Cet exercice n'a rien à voir avec les nombres premiers mais il concerne deux théoriciens de nombres de tout premier plan !*

Retrouver le nombre qui a été efface dans l'anecdote suivante :

Je [G. H. Hardy[2]] *me rappelle qu'une fois en allant le voir* [Ramanujan[3]] *lorsqu'il était couché et malade à Putney, j'ai été conduit dans un taxi-cab portant le n°..., et remarquai que le nombre semblait plutôt ennuyeux, et j'espérai qu'il ne fût pas un présage défavorable. « Non », me répondit-il, « c'est un nombre très intéressant ; c'est le plus petit nombre exprimable comme une somme de deux cubes* [positifs] *de deux manières différentes ».*[4]

Indication : en espérant que le numéros de taxis à Putney ne sont pas trop grands, on pourra effectuer une recherche exhaustive.

[1] Christian Goldbach (1690 - 1764), mathématicien allemand.
[2] Godfrey Harold Hardy (1877 - 1947), mathématicien britannique
[3] Srinivâsa Aiyangâr Râmânujan (1887 - 1920), mathématicien indien
[4] d'après *La Symphonie des nombres premiers* de Marcus du Sautoy.