Sciences Numériques et Technologie

Alain Busser
Dominique Gluck
Christophe Mieszczak
Claire Savinas
Luc Vincent

Table des matières

Préface

A la rentrée prochaine, l'enseignement de Sciences Numériques et Technologie (SNT) va faire partie des enseignements communs et obligatoires de la classe de Seconde. La formation des élèves au numérique représente une part essentielle de nos convictions et nous souhaitons nous engager avec vous dans l'enseignement de cette nouvelle matière.

L'enseignement de SNT a pour visée de donner à l'élève des éléments qui lui permettent d'appréhender les principaux concepts numériques et de mieux comprendre la technologie qui nous entoure.

Nous partageons l'idée qu'il s'agit d'un enseignement très propice à la mise en pratique et nous avons donc voulu vous proposer ce recueil d'activités afin de vous accompagner dans votre enseignement.

Les activités présentes dans ce recueil ont été proposées par des formateurs et futurs professeurs de SNT et permettent d'aborder six thèmes du programme :

— Internet
— Le Web
— Les Réseaux Sociaux
— Les données structurées et leur traitement
— Localisation, cartographie et mobilité
— La photographie numérique

Toutes ces activités sont facilement réalisables en classe avec vos élèves. Elles ne nécessitent pas l'usage d'un ordinateur et peuvent simplement être traitées à l'aide d'une calculatrice.

L'ambition de ce recueil d'activités n'est évidemment pas de couvrir de manière exhaustive la totalité du programme de SNT mais de vous fournir un support sur lequel nous espérons que vous pourrez construire vos séances de classe !

Nous remercions sincèrement les auteurs pour leur précieuse collaboration à ce projet : Alain Busser, Dominique Gluck, Christophe Mieszczak, Claire Savinas et Luc Vincent.

Activité 1

Transport de données par voie routière

Cette activité s'inscrit dans la thématique **Internet** du programme de SNT. Elle est proposée par **Claire Savinas**, actuellement professeure de mathématiques et d'ICN au lycée Jean Vilar à Villeneuve-lès-Avignon. Claire Savinas assure également les formations académiques de SNT.

— **Contenu** : Indépendance d'internet par rapport au réseau physique.
— **Capacités attendues** : Caractériser quelques types de ré-

seaux physiques : obsolètes ou actuels, rapides ou lents, filaires ou non. Caractériser l'ordre de grandeur du trafic de données sur internet et son évolution.

1.1 Présentation de la méthode

Une société propose une méthode de transfert de données par voie routière. Un camion de 14 mètres de long pouvant stocker jusqu'à 100 Po (pétaoctets) de données assure le transit des données depuis un point de départ jusqu'à un point d'arrivée.

Le camion se rend au point de départ et les techniciens de la société vous aident à effectuer le transfert des données depuis vos serveurs vers le camion. Le transfert de 100 Po dure dix jours.

A l'arrivée du camion à destination, les techniciens effectuent le transfert des données depuis le camion jusqu'aux serveurs d'arrivée. Le transfert de 100 Po dure également dix jours.

On va chercher à calculer le débit de cette méthode et à le comparer avec un débit ADSL ou fibre.

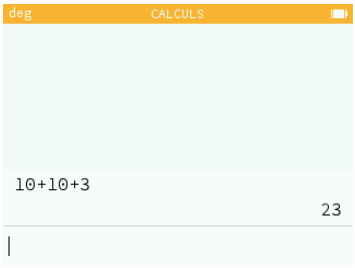
Indication : On prendra 10^{15} octets pour valeur approximative de 1 Po.

1.2 Calcul du débit

Un client de cette société demande un transfert de 100 Po entre deux points. Le transit de ces données du point de départ au point d'arrivée dure 3 jours.

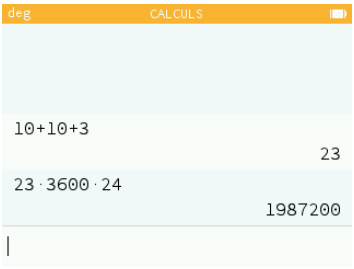
Question 1 Calculer le débit de ces 100 Po de données en Go/s.

On commence par calculer le temps total de transfert des données. Le chargement et le déchargement des données durent respectivement 10 jours. Le transit, quant à lui, dure 3 jours.

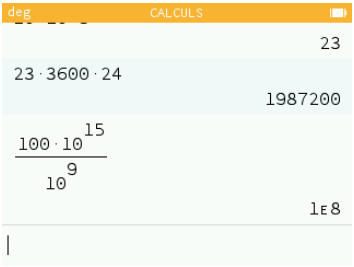


Le temps total de tranfert est donc $t_{\text{transfert}} = 23$ jours. Le débit est demandé en Go/s, il faut donc convertir le temps

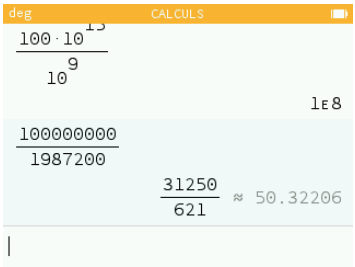
de transfert en seconde. Sachant qu'il y a 3600 secondes dans 1h et 24h dans une journée, on réalise le calcul suivant.



On a donc $t_{\text{transfert}} = 1987200$ s.
Il faut maintenant convertir 100 Po en Go. Sachant qu'un pétaoctet équivaut approximativement à 10^{15} octets et qu'un gigaoctet équivaut à 10^{9} octets :



La quantité de données transférée est donc 10^8 Go soit 100 millions de Go !
Il ne reste donc plus qu'à calculer le débit.



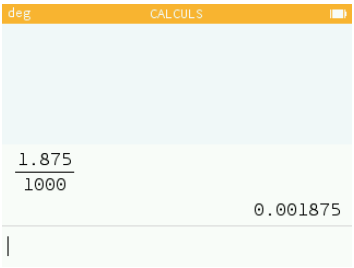
Le débit est donc de 50,3 Go/s.

1.3 Comparaison avec les débits ADSL et fibre

Un article trouvé sur internet indique que les débits maximaux ADSL et fibre sont respectivement de 15 Mbit/s et 1 Gbit/s.

Question 2 Convertir les débits ADSL et fibre en Go/s.

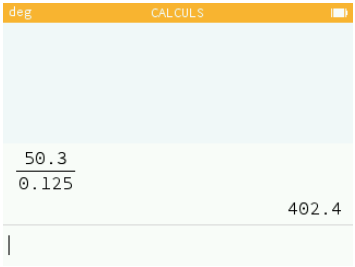
Attention ! Il y a 8 bits dans un octet. Ainsi 15 Mbit $= \frac{15}{8} = 1,875$ Mo et 1 Gbit $= \frac{1}{8} = 0,125$ Go !
On convertit ensuite les débits en Go/s.



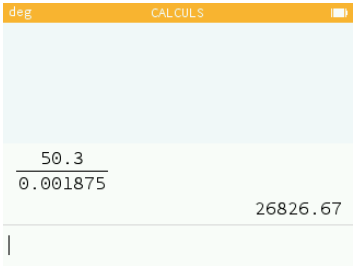
$debit_{\text{ADSL}} = 0,001875$ Go/s et $debit_{\text{fibre}} = 0,125$ Go/s.

Question 3 Comparer le débit camion avec les débits ADSL et fibre.

$debit_{\text{camion}} = 50,3$ Go/s et $debit_{\text{fibre}} = 0,125$ Go/s



Le débit camion est donc environ 400 fois plus important que le débit fibre.
$debit_{\text{camion}} = 50,3$ Go/s et $debit_{\text{ADSL}} = 0,001875$ Go/s



Le débit camion est environ 27 000 fois plus important que le débit ADSL !

Activité 2

Le PageRank de Google

Cette activité s'inscrit dans la thématique **Le Web** du programme de SNT. Elle est proposée par **Christophe Mieszczak**, actuellement professeur de mathématiques et d'ISN au lycée Léonard de Vinci à Calais.

- **Contenu** : Moteurs de recherche : principes et usages
- **Capacités attendues** : Mener une analyse critique des résultats fournis par un moteur de recherche.

2.1 Introduction

Le PageRank est l'algorithme d'analyse des liens hypertextes utilisé pour le classement des pages Web par le moteur de recherche Google. Le PageRank n'est qu'un indicateur parmi d'autres dans l'algorithme qui permet de classer les pages du Web dans les résultats de recherche. Ce système a été inventé par Larry Page, cofondateur de Google. Ce mot est une marque déposée.

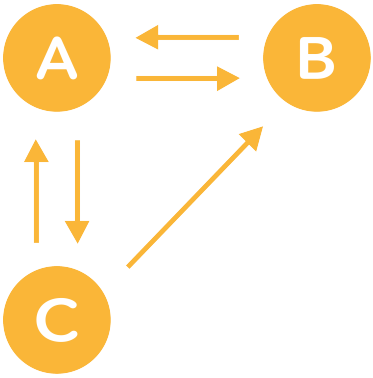
Le principe de base est d'attribuer à chaque page une valeur proportionnelle au nombre de fois que passerait par cette page un utilisateur parcourant le Web en cliquant aléatoirement, sur un des liens apparaissant sur chaque page.

A chaque instant, des "robots", appelés "spiders", parcourent la toile, de lien en lien, et mettent ainsi sans cesse à jour le PageRank des pages du web.

2.2 Avec une pièce de monnaie

On schématise une page web par un cercle et les liens entre les pages par des flèches. Trois pages A, B et C sont donc représentées de la façon suivante.

L'objectif est de déterminer la fréquence avec laquelle on visite chaque page lorsqu'on parcourt le graphe de façon aléatoire.



Trois pages dans un graphe

Au départ le nombre de visite de chaque page est nul.

On choisit une page de départ au hasard et on incrémente de 1 son nombre de visite (il passe donc à 1).

A l'aide d'une pièce de monaie équilibrée, nous allons nous promener sur ce graphe en respectant les règles ci-dessous :

- Si d'une page ne part qu'un seul lien alors on le suit.
- Si d'une page partent deux liens, on décide que l'un d'eux correspondra à face, l'autre à pile et on lance la pièce. Selon le résultat, on emprunte le lien correspondant.

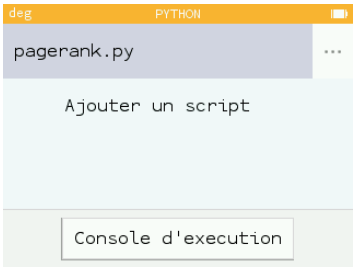
A chaque nouvelle page atteinte, on augmente son nombre de visite(s) et on se déplace à nouveau.

Question 1 Réalisez cette expérience 30 fois puis donnez un PageRank à nos trois pages, c'est à dire leurs fréquences de visite.

On réalise l'expérience en partant de la page C. Voici la liste des résultats obtenus : C, A, B, A, B, A, C, B, A, B, A, B, A, C, A, B, A, C, B, A, B, A, C, B, A, C, A, C, A, B. On compte le nombre d'apparitions de A, B et C : $n_A = 13$, $n_B = 10$ et $n_C = 7$.
Et donc les PageRank associés sont : $f_A = \frac{13}{30} = 0.43$, $f_B = \frac{10}{30} = 0.33$ et $f_C = \frac{7}{30} = 0.23$.

2.3 Simulation Python

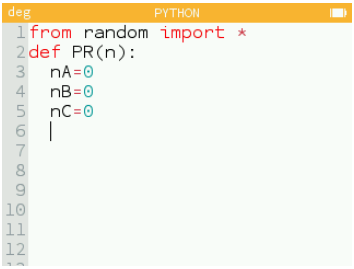
Question 2 Créez un nouveau programme Python et appelez le `pagerank.py`. Importez le module `random` qui sera nécessaire pour cette activité en tapant `from random import *` dans le script. Définissez ensuite une fonction nommée `PR(n)` dont l'objectif sera de réaliser n déplacements aléatoires sur le graphe précédent puis de renvoyer les résultats attendus.



On note nA, nB et nC les variables entières égales au nombre de visites des pages A, B et C.

Question 3 Commencez, dans notre fonction, par les définir en les initialisant à 0.

On définit les 3 variables.

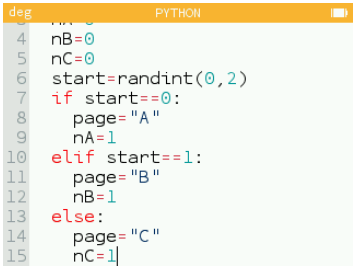


La page en cours de visite sera modélisée par la variable `page` qui pourra prendre les valeurs `"A"` ou `"B"` ou `"C"`. Pour démarrer il nous faut donc choisir au hasard entre A, B et C. Pour cela on va utiliser la fonction `randint(a,b)` du module `random`. Cette fonction génère un nombre aléatoire entre a et b.

Question 4 Définir la variable `start=randint(0,2)` qui choisit de façon équiprobable un nombre aléatoire entier dans {0,1,2} puis distinguer les cas suivants à l'aide d'un test "if" :

- Si `start` vaut 0 alors `page` sera égal à `"A"` et `nA` à 1.
- Sinon si `start` vaut 1 alors page sera égal à `"B"` et `nB` à 1.
- Sinon `page` sera égal à `"C"` et `nC` à 1.

On distingue les trois cas avec `if`, `elif` et `else`.



Il faut maintenant définir une boucle qui visitera n pages aléatoirement. A chaque “tour” de cette boucle :

- Si `page` est égal à `"A"` d’où partent deux liens, on définit `alea=randint(0,1)` :
  - si `alea` vaut 0 alors on passe à la page “B” et `nB` augmente de 1
  - sinon on passe à la page “C” et `nC` augmente de 1
- Sinon si `page` est égal à `"B"` d’où part un seul lien alors on passe à la page “A” et `nA` augmente de 1
- Sinon (on est forcément sur la page C d’où partent deux liens), on procède de la même façon que pour la page “A” afin de choisir entre A et B.

Question 5 Ecrire cette boucle.

> Il faut faire bien attention à l'indentation dans la disjonction des cas et des sous-cas.

```
for i in range(n):
    if page=="A":
        alea=randint(0,1)
        if alea==0:
            page="B"
            nB=nB+1
        else:
            page="C"
            nC=nC+1
    elif page=="B":
        page="A"
        nA=nA+1
    else:
        alea=randint(0,1)
        if alea==0:
            page="A"
            nA=nA+1
        else:
            page="B"
            nB=nB+1
```

Une fois la boucle terminée, il faut renvoyer les résultats attendus, c'est-à-dire les fréquences de visite de "A", "B" et "C" après n+1 parcours (en comptant le choix de départ).

Question 6 Définir ainsi les variables `fA`, `fB` et `fC` qui désignent les fréquences de visite des trois pages et terminer la

fonction par `return fA, fB, fC`.

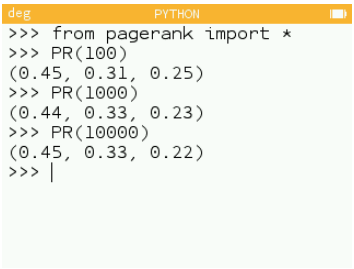
Les fréquences sont définies de la manière suivante :
$f = \frac{\text{nombre d'apparition de la page}}{\text{nombre d'expériences réalisées}}$

```
deg                    PYTHON
28      else:
29        alea=randint(0,1)
30        if alea==1:
31          page="A"
32          nA=nA+1
33        else:
34          page="B"
35          nB=nB+1
36    fA=nA/(n+1)
37    fB=nB/(n+1)
38    fC=nC/(n+1)
39    return fA,fB,fC
```

Question 7 Testez votre algorithme avec 100 puis 1000 puis 10000 déplacements. Classez A, B et C par popularité.

Rendez-vous dans la console Python en sélectionnant les trois points à gauche du nom du script et en choisissant **Exécuter le script**. Tapez ensuite `PR(100)`, `PR(1000)` et `PR(10000)` et observez les résultats. On a ici ajouté les instructions `round(fA,2)`, `round(fB,2)` et `round(fC,2)` dans le script pour arrondir les fréquences à deux chiffres

après la virgule.

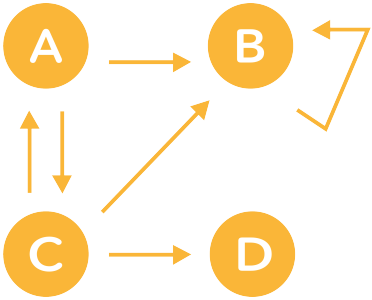


Les résultats de la question 1 étaient finalement assez proches de ce que l'on obtient ici !

2.4 Un premier cas particulier

Voici un nouveau graphe qui contient cette fois quatre pages : A, B, C et D.

Question 7 Observez le nouveau graphe représentant les liens entre les pages A, B, C et D et identifiez quel est le problème.



Graphe a quatre pages

> Si l'on se retrouve sur la page B, il n'y a aucun moyen de revenir sur les pages A, C ou D. De même, la page D ne mène à aucune autre page.

Pour pallier à ce problème, on décide que si la page visitée ne redirige pas vers une autre page, alors on choisit au hasard la page suivante parmi les autres.

Question 8 Adapter la fonction `PR(n)` pour calculer le PageRank des pages A, B, C et D. Testez ensuite votre algorithme avec 100 puis 1000 puis 10000 déplacements. Classez A, B, C et D par popularité.

Voici les résultats obtenus après avoir modifié la fonction :

deg PYTHON

```
>>> from pagerank import *
>>> PR(100)
(0.19, 0.31, 0.34, 0.17)
>>> PR(1000)
(0.24, 0.29, 0.28, 0.18)
>>> PR(10000)
(0.25, 0.28, 0.28, 0.19)
>>>
```

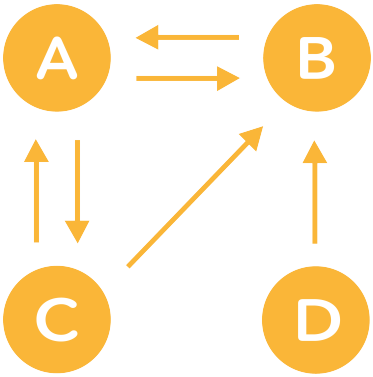
2.5 Un deuxième cas particulier

Voici un nouveau graphe.

Question 9 Observez le graphe et trouvez quel est le problème.

Aucune page ne mène vers la page D.

Pour pallier à ce problème on décide que, à chaque page visitée

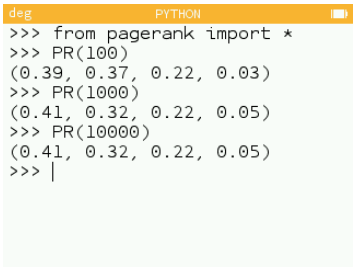


Graphe avec une page isolée

quelle qu'elle soit, on choisit une fois sur vingt la page suivante au hasard parmi toutes les pages.

Question 10 Adapter la fonction `PR(n)` pour calculer le PageRank des pages A, B, C et D. Testez ensuite votre algorithme avec 100 puis 1000 puis 10000 déplacements. Classez A, B, C et D par popularité.

Voici les résultats obtenus après avoir modifié la fonction :



On a ajouté une disjonction de cas en début de boucle pour tester si `nA`, `nB`, `nC` ou `nD` n'est pas multiple de 20 :

```
if nA%20==0 or nB%20==0 or nC%20==0 or nD%20==0:
    alea=randint(0,3)
```

```
    if alea==0:
        page="A"
        nA=nA+1
    elif alea==1:
        page="B"
        nB=nB+1
    elif alea==2:
        page="C"
        nC=nC+1
    else:
        page="D"
        nD=nD+1
```

Activité 3

Modélisation d'un réseau social

Cette activité s'inscrit dans la thématique **Les Réseaux Sociaux** du programme de SNT. Elle est proposée par **Claire Savinas**, actuellement professeure de mathématiques et d'ICN au lycée Jean Vilar à Villeneuve-lès-Avignon. Claire Savinas assure également les formations académiques de SNT.

— **Contenu** : Rayon, diamètre et centre d'un graphe.
— **Capacités attendues** : Déterminer ces caractéristiques sur des graphes simples.

3.1 Découverte du problème à résoudre

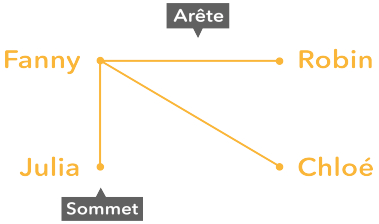
Fanny utilise avec ses amis un réseau social. Elle a fait la liste des liens d'amitiés dans le tableau suivant. Une croix dans le tableau signifie que les deux personnes concernées partagent un lien d'amitié. L'objectif de cette activité est de modéliser ces liens d'amitié pour pouvoir ensuite les étudier.

	Fanny	Chloé	Robin	Maéva	Angie	Matéi	Julia
Fanny		X	X				X
Chloé	X		X	X	X	X	X
Robin	X	X				X	
Maéva		X			X		X
Angie		X		X			X
Matéi		X	X				
Julia	X	X		X	X		

3.2 Découverte de la notion de graphe

Pour étudier les liens d'amitié, on va utiliser une représentation sous forme de graphe.

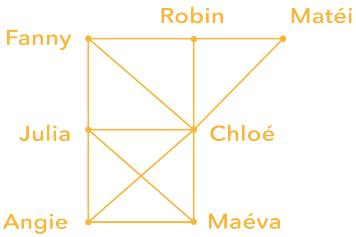
Question 1 Compléter le graphe ci-dessus où chaque arête signifie "... et .... sont amis".
Chaque personne est représentée par un sommet et chaque lien



Représentation sous forme de graphe

d'amitié est représenté par une arête.

Voici le graphe complété



3.3 Découverte de l'écartement d'un sommet d'un graphe

On considère que seulement deux personnes amies peuvent communiquer entre elles. Fanny peut parler avec ses amis qui sont Chloé, Robin et Julia. Pour communiquer avec Maéva, elle doit passer par exemple par Chloé, ce qui fait une distance de 2. Pour parler à Angie, elle peut également passer par Chloé, ce qui fait également une distance de 2. Pour parler à Matéi, elle peut passer par Robin, ce qui fait également une distance de 2. La distance maximale entre Fanny et les autres personnes du graphe est de 2.

Question 2 Compléter le tableau ci-dessous avec la distance maximale, c'est-à-dire le nombre de liens d'amitié entre un sommet avec les autres sommets du graphe.

Fanny	Chloé	Robin	Maéva	Angie	Matéi	Julia
2						

Voici le tableau rempli.

Fanny	Chloé	Robin	Maéva	Angie	Matéi	Julia
2	1	2	2	2	2	2

3.4 Déterminer le ou les centres d'un graphe

Dans un graphe donné, le centre est le sommet dont l'écartement est minimal. Un graphe peut avoir plusieurs centres. Les centres d'un graphe sont alors les éléments à partir desquels l'information se diffuse le plus vite dans un réseau.

Question 3 Déterminer le centre de ce graphe.

> D'après le tableau précédent, la distance minimale est détenue par Chloé qui est donc le centre du graphe.

Question 4 Interpréter la réponse précédente dans le contexte de l'activité.

> Chloé est la personne qui devra passer par le moins d'intermédiaires pour joindre tout le groupe. Ici, elle est la seule qui puisse communiquer avec toutes les autres personnes directement.

3.5 Déterminer le rayon d'un graphe

Le rayon d'un graphe est l'écartement d'un centre du graphe.

Question 5 Déterminer le rayon de ce graphe.

> Chloé est le centre du graphe. Son écartement est de 1. Le rayon du graphe est donc 1.

Question 6 Interpréter la réponse précédente dans le contexte de l'activité.

> Le rayon est le nombre d'intermédiaires moins un pour joindre tout le groupe en partant du ou des centres du graphe.

3.6 Déterminer le diamètre d'un graphe

Dans un graphe donné, le diamètre est la plus longue distance entre deux sommets.

Question 7 Déterminer le diamètre de ce graphe.

> D'après le tableau des distances établi en début d'activité, la distance maximale entre deux sommets est de 2. Le diamètre de ce graphe est donc 2.

Question 8 Interpréter la réponse précédente dans le contexte de l'activité.

> Le diamètre du graphe est le nombre maximum d'intermédiaires moins un pour que deux personnes communiquent entre elles.

3.7 Déterminer le nombre de liens d'amitiés à utiliser pour contacter une personne.

Une chaîne d'un graphe est une liste ordonnée de sommets du graphe telle que chaque sommet de la liste soit adjacent au suivant. La longueur d'une chaîne est le nombre d'arêtes qui la composent. La distance entre deux sommets d'un graphe est la plus petite des longueurs des chaînes qui la relient.

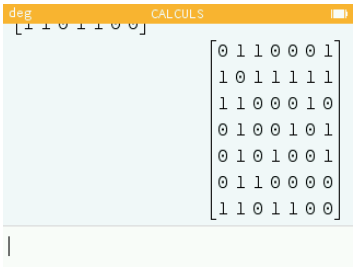
La matrice d'adjacence associée à un graphe non orienté d'ordre n dont les sommets sont numérotés de 1 à n est la matrice carrée d'ordre n, où le terme figurant en ligne i et colonne j est égal au nombre d'arêtes reliant i et j.

Question 9 Déterminer la matrice d'adjacence de ce graphe.

En conservant le même ordre de prénoms que dans le tableau, la matrice d'adjacence est la suivante :

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

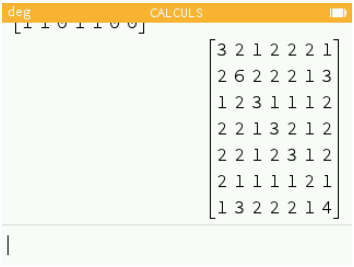
Sur la calculatrice :



Soit M la matrice d'adjacence associée à un graphe non orienté dont les sommets sont numérotés. p désigne un nombre entier naturel. M^p donne le nombre de chaînes de longueur p reliant i et j.

Question 10 Saisir la matrice d'adjacence de ce graphe dans la calculatrice et calculer le carré de cette matrice.

Appuyez sur la touche shift puis sur la touche e^{x} de la calculatrice pour créer une matrice. La remplir ensuite avec les coefficients et appuyer sur la touche x^2 pour calculer le carré.



Question 11 Interpréter le coefficient $a_{1,2}$ du carré de la matrice d'adjacence.

$a_{1,2} = 2$ donc il y a deux chaînes de longueur 2 qui relient Fanny et Chloé. Il s'agit de Fanny-Robin-Chloé et Fanny-Julia-Chloé.

Question 12 Interpréter le coefficient $a_{3,4}$ du carré de la matrice d'adjacence.

> $a_{3,4} = 1$ donc il n'y a qu'une seule chaîne de longueur 2 qui relie Robin et Maéva. Il s'agit de Robin-Chloé-Maéva.

3.8 Différence de fonctionnement entre Facebook et Twitter

Sur le réseau Facebook, pour être en relation, deux personnes inscrites doivent en effet s'accepter mutuellement comme "amis", alors qu'il est possible sur Twitter, de suivre une personne inscrite sans que cela ne soit réciproque.
On peut représenter ces relations unilatérales par des graphes et modéliser le sens de la relation par une orientation de l'arête.

Activité 4

Données structurées en Python

Cette activité s'inscrit dans la thématique **Les données structurées et leur traitement** du programme de SNT. Elle est proposée par **Alain Busser**, actuellement professeur de mathématiques au lycée Roland-Garros du Tampon. Alain Busser est aussi animateur à l'IREM de La Réunion et créateur du langage de programmation Sophus.

— **Contenu** : Données structurées.
— **Capacités attendues** : Identifier les différents descripteurs

d'un objet. Distinguer la valeur d'une donnée de son descripteur.

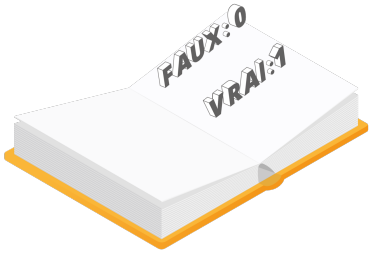
4.1 Exemple de données structurées

Il y a bien des siècles, Aristote (384 av. JC - 322 av. JC) constata que la phrase suivante était fausse : "2+2=5 et en plus je suis une fille". Qu'elle soit prononcée par un garçon ou par une fille.

Vers 1850, George Boole (1815-1864) rapprocha cette phrase du calcul 0×a=0 sans avoir à tenir compte de la valeur de a. Boole a alors proposé de représenter le *faux* par le nombre 0 et le *vrai* par le nombre 1. Cela lui a notamment permis de ramener les "et" et les "ou" à des multiplications et des additions.

En cela, on peut dire que Boole a proposé une traduction vers des nombres qui permet de fabriquer un dictionnaire donnant la traduction des mots "Vrai" et "Faux" sous forme de nombres. Voici ce dictionnaire, où les mots sont dans l'ordre alphabétique et les définitions (0 et 1) données juste après les mots à définir, précédées de double-points comme dans le Larousse et le Robert.

Cela veut tout simplement dire que pour Boole, "Faux" veut dire "0" et "Vrai" veut dire "1". En Python, on écrit ces définitions séparées par une virgule, entre accolades. La variable donnant ceci s'appelle `loi` parce que le titre du livre de Boole est *Les lois*



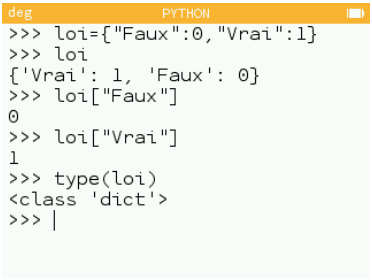
Dictionnaire de Boole

de la pensée.

Le type de la variable `loi` est `dict`, abréviation de “dictionnaire”. Et pour consulter le dictionnaire, par exemple pour savoir comment Boole traduit “Vrai”, on entre `loi["Vrai"]` dans la console :

4.2 Relations

Si on entre `loi.items()` dans la console de Python, on voit des couples du type (clé,valeur). Un dictionnaire Python est un ensemble d’objets comme (‘Faux’,0) qu’on appelle couple en mathématiques.



Utilisation d'un dicitonnaire en Python

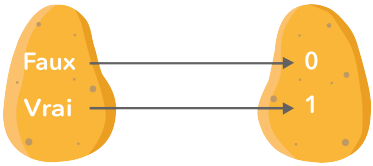
On dit qu'on a accouplé le nom "Faux" avec le nombre 0. La notion est dûe à René Descartes qui assimilait les points d'un plan à des couples (x,y) de coordonnées de ces points.

Vers le milieu du XIXe siècle (à l'époque de Boole), le mathématicien anglais Arthur Cayley (1821-1895) a proposé de dessiner un couple par une flèche. Par exemple pour le couple (a,b), Cayley dessine a→b.

L'ensemble de ces dessins de flèches entre divers éléments s'appelle un graphe. Cette notion, dûe à Cayley, est omniprésente en informatique.

Si on veut dessiner le graphe d'un dictionnaire, on constate qu'il est possible de regrouper les clés dans un ensemble dit de départ, les valeurs dans un autre ensemble dit d'arrivée.

Comme l'ensemble de départ contient deux éléments `Faux` et `Vrai`, on représente l'ensemble par une sorte de sac (ou de patate) contenant deux éléments, l'un représentant `Vrai`, l'autre représentant `Faux`.



Idem pour l'ensemble d'arrivée, dont les deux éléments représentent respectivement 0 et 1. On a deux flèches, allant de Vrai vers 1 pour représenter le couple (Vrai,1) et l'autre allant de Faux à 0, représentant le couple (Faux,0).

Un ensemble de couples s'appelle une relation. Cette définition est de Gottlob Frege (1848-1925) et son application aux bases de données date de 1970. Elle est dûe à Edgar Codd (1923-2003).

Dans une base de données, une relation n'est pas représentée par des flèches mais par une table comme celle-ci :

Clés	Valeurs
Faux	0
Vrai	1

Le dictionnaire de Boole est quand même une relation bien particulière : chaque clé n'est associée qu'à une valeur. Une telle relation s'appelle application ou fonction.

4.3 Boole et les fonctions

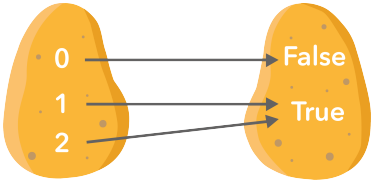
Python possède une fonction donnant la valeur entière correspondant à un booléen. Cette fonction s'appelle `int()` (comme "integer" en anglais qui veut dire "entier" en français). Mais Boole était anglais, et n'utilisait pas les mots français Vrai et Faux.

Question 1 Taper les instructions suivantes dans la console et noter le résultat.

- `int(False)`
- `int(True)`
- `int(2+2==4)`
- `int(2+2==5)`

Python a une autre fonction qui fait le contraire de `int` (l'annuaire inversé de tout-à-l'heure) et elle s'appelle `bool`, en hommage à George Boole.

La fonction `bool()` associe la valeur False au nombre 0 et la valeur True au nombre 1 (ou à tout autre nombre différent de 0, par exemple 2).



Représentation avec des patates de la fonction Python bool

Question 2 Taper les instructions suivantes dans la console et noter le résultat.

- `bool(0)`
- `bool(1)`
- `bool(0*0)`
- `bool(0*1)`
- `bool(1*1)`
- `bool(0+0)`
- `bool(0+1)`
- `bool(1+1)`

Question 3 Comparer avec le résultat des instructions suivantes.

- `False and False`
- `False and True`
- `True and True`

— `False or False`
— `False or True`
— `True or True`

La ressemblance entre les deux résultats est à l'origine de l'expression "algèbre de Boole" pour désigner le calcul logique. À l'inverse, les ordinateurs calculent par combinaison d'opérations logiques, en binaire (les opérandes valent tous 0 ou 1).

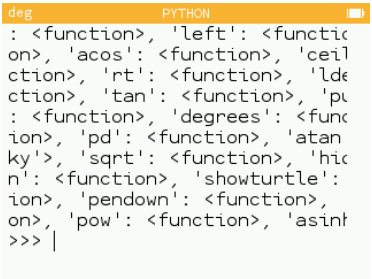
4.4 Un dictionnaire variable : le dictionnaire des variables

Outre le fait qu'avoir seulement deux entrées dans un dictionnaire soit relativement faible pour des Big Data, le dictionnaire de Boole omet un aspect important des Big Data : le fait qu'elles soient évolutives. Par exemple, lorsqu'un individu demande à placer son numéro de téléphone sur liste rouge, on le retire de l'annuaire.

Chaque exécution d'une instruction a pour effet de modifier au moins une variable, donc l'état de la machine (ordinateur). Ce modèle selon lequel un ordinateur est un dictionnaire donnant à chaque nom de variable une valeur provient de Christopher Strachey (1916-1975) et Dana Scott (1932-).

Il est possible de trouver ce dictionnaire en Python à l'aide de la fonction `globals()` qui donne la liste de toutes les variables

(globales) de la session Python courante.



Fonction globals en Python

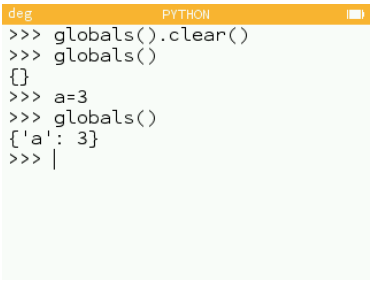
Pour y voir plus clair, on va tout d'abord nettoyer le dictionnaire en le réinitialisant.

```
>>> globals().clear()
>>> globals()
{}
```

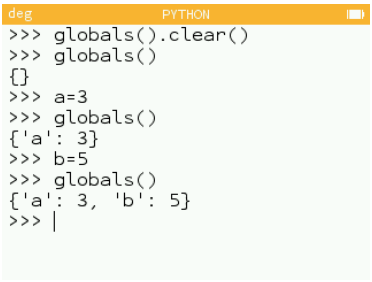
Ceci étant fait, l'état de la machine devrait donc être vide, c'est-à-dire qu'il n'y a plus aucune variable globale.

Après l'affectation de la valeur 3 à la variable a (obtenue par `a=3` en Python) le nouvel état de la machine est :

Si on affecte ensuite la valeur 5 à la variable b, le dictionnaire globals() s'enrichit d'une nouvelle variable.



Effacement et remplissage de la fonction globals en Python



Ajout d'une variable globale en Python

Ce nouvel état est représenté par le dictionnaire `{'a': 3, 'b': 5}` que renvoie `globals()`.

Représentation visuelle de globals

Ce dictionnaire sert à calculer des expressions algébriques :

- Si on écrit `3*8` dans la console, on demande implicitement à Python de calculer 3×8 et d'afficher le résultat : la console est une sorte de calculatrice.
- Mais si on écrit `a*8`, Python ne peut pas effectuer la multiplication avant de savoir quelle est la valeur de a (par quel nombre il doit remplacer la lettre "a"). Pour ce faire, Python va consulter le dictionnaire et trouver que "a" signifie 3, ce qui lui permet d'évaluer `a*8` en 3×8 puis en 24.

Si on veut effectuer quelque chose de répétitif, on utilise une variable `n` servant de compteur (puisqu'on va compter les étapes avec n). Pour cela on fera usage de l'expression n+1 qui donne l'entier suivant n (2 pour n=1, 3 pour n=2, etc).

Question 4 Compléter l'affichage de la session ci-dessous (en console) :

```
>>> n = 1
>>> globals()

>>> n+1
2
>>> globals()

>>> n<5
```

Question 5 Dans le script précédent, quelle est la valeur finale de n (affichée par globals() par exemple) ?

Comme on le voit, lorsque Python veut savoir si n est inférieur à 5, il va là encore lire dans le dictionnaire la valeur n, puis comparer ce nombre avec 5.

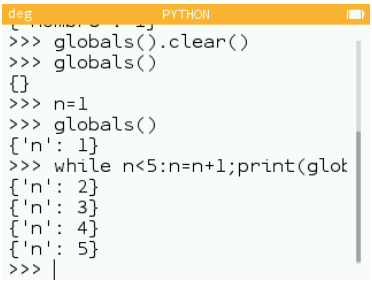
Pour que n varie vraiment, il ne faut pas se contenter de calculer n+1, mais il faut ensuite injecter le résultat dans n. Ce qui se fait en écrivant n=n+1

Question 6 Compléter les affichages des états successifs de la machine :

```
>>> n = n+1
>>> globals()

>>> n = n+1
>>> globals()
```

```
>>> n = n+1
>>> globals()

>>> n = n+1
>>> globals()

>>> n<5
```

Pour automatiser le comptage 1, 2, 3, 4, on peut demander à Python de modifier n par n=n+1 tant que n est inférieur à 5. Cela fait bien 4 itérations :



Modification des globals avec une boucle

En résumé, pour compter jusqu'à 4 avec Python, il faut

— initialiser la variable n à 1
— passer de n à l'entier suivant n+1 tant que n est inférieur à 5

En langage Python, il est d'usage de commencer à compter à partir de 0, et non de 1. On procède de cette manière :

```
>>> globals().clear()
>>> globals()
{}
>>> n = 0
>>> globals()

>>> while n<4: n=n+1; print(globals())
```

Question 7 Compléter les étapes suivantes qui donnent les états successifs de la machine :

- `{"n":...........}`
- `{"n":...........}`
- `{"n":...........}`
- `{"n":...........}`

Python peut faire de manière automatique le passage de n à n+1 ainsi que l'initialisation de n et le test de fin de boucle, avec l'instruction for qui traduit "pour n allant de 0 à 4 (non compris)" :

```
>>> globals().clear()
>>> globals()
{}
>>> for n in range(0,4):print(globals()
```

Question 8 Compléter les étapes suivantes qui donnent les états successifs de la machine :

- `{"n":..........}`
- `{"n":..........}`
- `{"n":..........}`
- `{"n":..........}`

Activité 5

Transmission des coordonnées GPS : trame NMEA

Cette activité s'inscrit dans la thématique **Localisation, cartographie et mobilité** du programme de SNT. Elle est proposée par **Luc Vincent**, actuellement professeur de Physique-Chimie et d'ISN/ICN au lycée des Graves à Bordeaux.

— **Contenu** : Protocole NMEA 0183.
— **Capacités attendues** : Décoder une trame NMEA pour trou-

ver des coordonnées géographiques.

5.1 Introduction

Les différents composants d'un appareil électronique (ex : un téléphone mobile) communiquent par des protocoles normalisés. Ainsi, les puces GPS qui effectuent les calculs de positionnement envoient leurs résultats présentés suivant une trame normalisée : la trame NMEA 0183. Le développeur d'un dispositif souhaitant interroger un GPS sait qu'il pourra exploiter cette trame pour obtenir des informations de date et de position.

5.2 La norme NMEA 0183

La norme NMEA est le protocole de transmission des données GPS. Ces données sont transmises sous la forme de trames. Chaque trame commence par le caractère $ et se compose de plusieurs éléments séparés par des virgules. Voici un exemple de trame :

```
$GPGGA, 064036.289, 4836.5375, N, 00740. 9373,
E, 1, 04, 3.2, 200.2, M, , , , 0000*0E
```

Les deux premiers caractères correspondent à l'identifiant du récepteur : ici GP pour Global Positioning System. Les trois lettres

suivantes correspondent à l'identifiant de la trame : GGA pour GPS Fix et Date. C'est la trame la plus courante.

5.3 Eléments de la trame GGA

Décomposons maintenant cette trame selon les premiers éléments qui la composent :

- **GPGGA** : type de la trame
- **064036.289** : heure d'envoi de la trame, ici 06h 40min 36,289s (UTC)
- **4836.5375, N** : latitude Nord, ici 48°36,5375' (en DM, degrés minutes)
- **00740.9373, E** : longitude Est, ici 7°40,9373' (en DM également)
- **1** : type de positionnement (1 pour le positionnement GPS)
- **04** : nombre de satellites utilisés
- **3.2** : précision horizontale
- **200.2, M** : altitude, ici 200 mètres

5.4 Les notations DMS, DM et DD

Généralement, on exprime les coordonnées géographiques dans le système sexagésimal, noté DMS pour degrés, minutes, secondes. Par exemple 49°30'30" pour 49 degrés, 30 minutes et 30

secondes. Une minute d'angle vaut 1/60 degrés tandis qu'une seconde d'angle vaut 1/3600 degrés.

Il est également possible d'utiliser les unités DM (Degré Minute) ou DD (Degré décimal) :

— En DMS : 49°30'30"
— En DM : 49°30,5'
— En DD : 49,5083° (généralement avec quatre décimales)

Question 1 Vérifier par un calcul que la latitude 48°36.5375' (DM) de la trame NMEA donnée en exemple en début d'activité correspond à 48°36'32.25" (DMS).

Sachant que $1' = 60''$ alors $0.5375' = 60 \times 0.5375 = 32.25''$.

5.5 Extractions des données contenues dans une trame avec Python

Voici une vue des résultats de quelques instructions Python obtenues depuis la console.

```
>>> ligne="nom,prenom,age,17"
>>> element=ligne.split(",")
>>> element
['nom','prenom','age','17']
```

```
>>> type(element)
<class 'list'>
>>> element[1]
'prenom'
>>> prenom=element[1]
>>> prenom[2:4]
'en'
>>> type(element[3])
<class 'str'>
>>> int(element[3])
17
>>> float(element[3])
17.0
>>>
```

Question 2 D'après ces résultats, quelle instruction en python permet d'obtenir une liste nommée `attribut` à partir d'une chaîne de caractères nommée `trame`?

```
trame = "$GPGGA, 064036.289, 4836.5375, N, 00740.
9373, E, 1, 04, 3.2, 200.2, M, , , , 0000*0E"
```

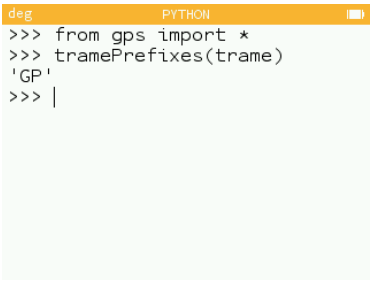
> Il faut saisir l'instruction : `attribut=trame.split(",")`.

Question 3 Ecrire la fonction `tramePrefixes(trame)` qui reçoit une trame complète et renvoie l'identifiant du récepteur, c'est-à-dire les deux premières lettres du type de la trame (pre-

mier élément après le caractère $. Sur la trame d'exemple, la fonction doit renvoyer "GP".

```
def tramePrefixes(trame):
     talkerId=trame[1:3]
    return talkerId
```

On teste ensuite la fonction avec la trame d'exemple.



Question 4 Modifier cette fonction pour qu'elle renvoie le nom de l'équipement qui a émis la trame. On utilisera les correspondances suivantes :

- BD ou GB : Beidou
- GA : Galileo

— GP : GPS
— GL : GLONASS

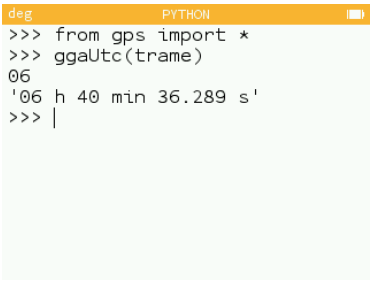
```
def tramePrefixes(trame):
    talkerId=trame[1:3]
    if talkerId=="GD" or talkerId=="GB":
        tramePrefixesValue="Beidou"
    elif talkerId=="GA":
        tramePrefixesValue="Galileo"
    elif talkerId=="GP":
        tramePrefixesValue="GPS"
    elif talkerId=="GL":
        tramePrefixesValue="GLONASS"
    return tramePrefixesValue
```

On teste alors la fonction.

```
deg                    PYTHON
>>> from gps import *
>>> tramePrefixes(trame)
'GPS'
>>>
```

Question 5 Ecrire une fonction `ggaUtc(trame)` qui reçoit une trame complexe et renvoie l'heure en h, min, s.

```
def ggaUtc(trame):
    #on transforme la trame en liste
    attribut=trame.split(',')
    #on sélectionne l'élément qui correspond à l'heure
    time=attribut[1]
    utc=time[:2]+" h "+time[2:4]+" min "+time[4:]+" s"
    return utc
```



Récupération de l'heure UTC

Question 6 Ecrire une fonction `ggaLat(trame)` qui reçoit une trame complète et renvoie la latitude convertie en DMS.

```
def ggaLat(trame):
    attribut=trame.split(',')
```

```
    lat=attribut[2]
    lat_deg=lat[:2]
    lat_min=lat[2:4]
    lat_sec=str(float(lat[4:])*60)
    latDMS=lat_deg+" D "+lat_min+" M "+lat_sec+" S "
    return latDMS
```

```
deg                    PYTHON
>>> from gps import *
>>> ggaLat(trame)
'48 D 36 M 32.25 S '
>>>
```

Récupération de la latitude en DMS

5.6 A retenir

Pour définir sa position sur la Terre, on utilise le plus souvent les coordonnées géographiques : la latitude, la longitude et l'altitude. Pour la latitude et la longitude on rencontre trois notations :

— Degré Minute Seconde (DMS)
— Degré Minute (DM)
— Degré décimal (DD)

Les récepteurs “GPS” fournissent la localisation sous une forme normalisée facilement décodable, par exemple selon le protocole NMEA 0183 (National Marine Electronics Association).

Activité 6

Le codage des couleurs

Cette activité s'inscrit dans la thématique **La photographie numérique** du programme de SNT. Elle est proposée par **Dominique Gluck**, actuellement professeur de Sciences de l'Ingénieur au lycée Raoul Dautry à Limoges.

— **Contenu** : Photosites, pixels, résolution, profondeur de couleur. Traitement d'image.
— **Capacités attendues** : Distinguer les photosites du capteur et les pixels de l'image en comparant les résolutions du capteur et de l'image selon les réglages de l'appareil. Traiter par programme une image pour la transformer en agissant sur les trois composantes de ses pixels.

6.1 Introduction

Cette séance a pour objectif de comprendre, à l'aide d'instructions Python, la structure d'une image numérique et le codage des pixels selon leurs composantes rouge, vert, bleu.

On s'exercera à jouer avec les couleurs à l'écran et à effectuer des traitements simples sur l'image.

La calculatrice NumWorks intègre un module graphique appelé `kandinsky` qui va nous permettre d'analyser et de définir la couleur de chacun des pixels de l'image (écran 320*222 points).

Système de coordonnées

6.2 Découverte du codage des couleurs

On aura besoin ici de deux fonctions du module graphique `kandinsky` :

- `set_pixel(x,y,color)` : colore le pixel de coordonnées (`x`, `y`) de la couleur définie par `color`.
- `color(r,g,b)` : définit une couleur en fonction de ses composantes rouge, vert, bleu.

Question 1 À partir de l'écran principal, aller sur l'application Python (valider en appuyant sur (ok)).

Faire défiler la page et ajoutez un script que l'on nommera `couleurs.py` (valider par (ok)).

Saisir ensuite le code suivant :

```
from kandinsky import *
for y in range(25):
  for x in range(320):
    col=color(x,0,0)
    set_pixel(x,y,col)
```

Question 2 Exécuter le script. Qu'observe-t-on ?

> On observe une barre de dégradé allant du noir au rouge. Le dégradé ne va pas jusqu'au bord de l'écran ; il semble

qu'une nouvelle barre de dégradé débute à droite de l'écran

Quelques explications sur le programme :

- `x` désigne la position horizontale du point (abscisse), `y` sa position verticale (ordonnée)
- ligne 1 : on importe les fonctions graphiques du module `kandinsky`
- ligne 2 : on fait varier `y` de 0 à 24 (boucle principale du programme)
- ligne 3 : on fait varier `x` de 0 à 319 (boucle secondaire du programme)
  Pour chaque valeur de `y` et `x`, on exécute les lignes 4 et 5 :
- ligne 4 : on définit une couleur : ici l'intensité du rouge dépend de la position `x`, l'intensité du vert et du bleu sont

fixées à zéro.
— ligne 5 : on donne au pixel courant la couleur que l'on vient de définir

La couleur du point dépend donc de la position `x`, d'où un dégradé sur la ligne horizontale ; on reproduit le tracé sur 25 lignes, d'où la création d'une bande horizontale de dégradé d'une hauteur de 25 lignes.

Nous allons tenter de répondre à la question suivante : pourquoi le dégradé ne s'étend-il pas sur l'ensemble de la largeur de l'écran ?

Le dégradé du noir vers le rouge s'étend sur les 4/5e de la largeur de l'écran, puis reprend à partir du noir. La couleur est définie par `color(r,g,b)` selon le système rouge, vert, bleu (rgb = red, green, blue) où les paramètres `r`, `g`, `b`, sont des nombres entiers dont la valeur doit être comprise entre 0 et 255.

Cela vient du fait que la valeur de chacune des composantes rouge, vert, bleu est codée sur un octet. En effet, un octet est un nombre binaire de 8 bits valant chacun 0 ou 1, ce qui donne 256 combinaisons possibles.

Pour s'en convaincre, tapez dans la console `bin(13)` : la calculatrice affiche l'expression binaire (notée `0b`) de 13 : `0b1011`, on voit qu'il faut 4 bits pour coder en binaire le nombre 13 : `1011`. En effet $13 = 1\times8 + 0\times4 + 1\times2 + 1\times1$.

Question 3 Combien de bits sont nécessaires pour coder 255 et 256 en binaire ? Pour répondre, tapez `bin(255)` et `bin(256)` et relevez les expressions binaires correspondantes.

`bin(255)` = `'0b 1111 1111'` : 8 chiffres sont nécessaires
`bin(256)` = `'0b 1 0000 0000'` : 9 chiffres sont nécessaires
255 est bien la plus grande valeur que l'on peut coder sur 8 bits ;
$255 = 1\times128+1\times64+1\times32+1\times16+1\times8+1\times4+1\times2+1\times1$

Question 4 Chaque point étant défini par ses trois couleurs et chaque couleur étant codée sur un octet, calculer en octets, puis en ko (kilo-octets), la mémoire nécessaire pour coder, selon ce principe, l'ensemble des points de l'écran de la calculatrice.

$n = 222\,\text{lignes}\times320\,\text{colonnes}\times3\,\text{couleurs} = 71040\,\text{pixels}\times3\,\text{octets} = 213120\text{o}$

Remarque : diverses solutions permettent de diminuer la taille du fichier nécessaire au stockage de l'image...

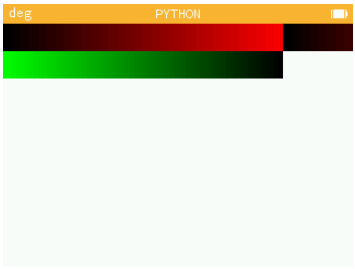
Editer le script couleurs et ajouter les lignes suivantes :

```
for y in range(25,50):
```

```
  for x in range(256):
    col=color(0,255-x,0)
    set_pixel(x,y,col)
```

Question 5 Exécutez le script et observez l'écran. Que fait ce morceau de programme ?

Il trace une barre de dégradés allant du vert au noir.



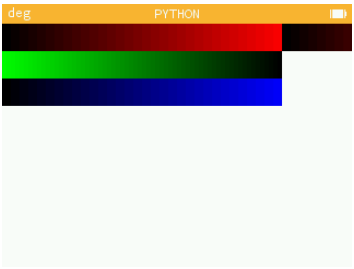
Question 6 Expliquez le fonctionnement du programme en commentant chaque ligne.

```
for y in range(25,50):
# pour les 25 lignes suivantes de l'écran
```

```
  for x in range(256):
  # pour x variant de 0 à 255
    col=color(0,255-x,0)
    # les niveaux du rouge et du bleu restent à 0
    # le niveau du vert varie de 255 à 0 en fonction de x
    set_pixel(x,y,col)
    # on attribue la couleur au pixel de coordonnées (x,y)
```

Question 7 Compléter le programme pour créer une troisième bande de dégradé noir vers bleu et de longueur 256.

On ajoute les lignes de code suivantes pour obtenir le résultat.

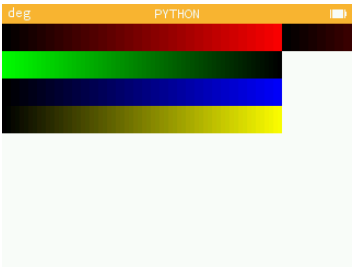



```
for y in range(50,75):
```

```
  for x in range(256):
    col=color(0,0,x)
    set_pixel(x,y,col)
```

Question 8 Créer une quatrième bande de dégradé noir vers jaune : le jaune correspond à une intensité égale du rouge et du vert ; le bleu est à 0.

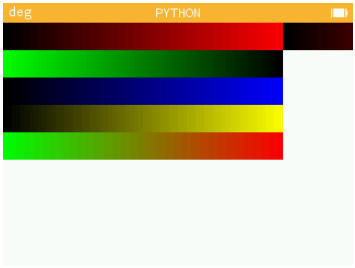
On ajoute les lignes de code suivantes pour obtenir le résultat.




```
for y in range(75,100):
  for x in range(256):
    col=color(x,x,0)
    set_pixel(x,y,col)
```

Question 9 Créer une cinquième bande de dégradé vert vers rouge.

On ajoute les lignes de code suivantes pour obtenir le résultat.



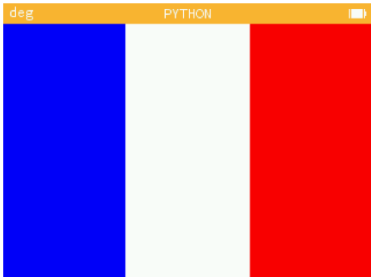

```
for y in range(100,125):
  for x in range(256):
    col=color(x,255-x,0)
    set_pixel(x,y,col)
```

Question 10 Observons de près : la calculatrice paraît-elle produire réellement 256 niveaux dans chaque couleur ?

Non, on observe en fait une succession de bandelettes verticales à l'intérieur desquelles la couleur est homogène. L'observation de bandelettes verticales montre que la calculatrice ne produit en fait que 32 niveaux différents dans chaque couleur, ce qui constribue déjà à économiser la taille de mémoire nécessaire pour stocker l'image.

6.3 Drapeau

L'objectif de cette partie est d'utiliser ses nouvelles connaissances pour dessiner le drapeau français.



Drapeau français

Indications

1. *Définir les couleurs* : On donne `blanc=color(255,255,255)`.
2. *Colorier* : On utilisera plusieurs boucles `for`. Plusieurs méthodes sont possibles : soit tracer les 3 rectangles de couleur, soit tracer une ligne entière et la reproduire 222 fois.
3. *Créer le script* `drapeau.py` : le tester, le mettre au point ; faire vérifier.
   Essayez de résoudre cet exercice en autonomie.

Besoin d'aide ?

Si l'on adopte la deuxième méthode (balayage ligne par ligne) du dessin du drapeau, le script peut commencer ainsi :

```
from kandinsky import *

# definition des couleurs (a completer):
bleu=color(..., ..., ...)
blanc=color(255,255,255)
rouge=color(..., ..., ...)

# pour toutes les lignes :
for y in range(222): # y varie de 0 à 221
  # partie gauche en bleu :
  for x in range(0,107): # premier tiers de la ligne
    set_pixel(x,y,bleu)
  # milieu en blanc :
  for x in range(108,215): # deuxieme tiers de la ligne
```

```
    # ...
```

Si l'on adopte la première méthode (dessin de 3 rectangles), on peut définir une fonction : `def rectangle(x,y,couleur)` qui trace un rectangle de largeur 107 et de hauteur 222, où `x` et `y` sont les coordonnées du point de départ et appeler trois fois la fonction `rectangle(x,y,couleur)`.

6.4 Nuances de gris

Question 11 Quelle est la couleur définie par `color(0,0,0)` ?

> Noir

Question 12 Quelle est la couleur définie par `color(255, 255, 255)` ?

> Blanc

Question 13 En déduire comment définir un gris moyen. Testez.

Les paramètres `r`, `g`, `b`, de la fonction `color(r,g,b)` doivent avoir les mêmes valeurs, par exemple : `color(128,128,128)`.

6.4.1 Mire de gris

Objectif : on veut écrire un programme permettant de partager l'écran en 10 bandes verticales de gris, allant du noir au blanc, le tout sur une hauteur de 100 pixels.

Question 14 Quelle sera la largeur d'une bande ?

$\frac{320}{10} = 32$ pixels

Question 15 On veut que les intensités de gris soient bien réparties entre les valeurs extrêmes : `color(0, 0, 0)` et `color(255, 255, 255)`. Quelle valeur faut-il ajouter au paramètre de couleur pour passer d'une bande à sa voisine ?

28, car si la première nuance est 0, la dixième sera $0+9\times28 = 252$.

Question 16 Saisir le script. La structure du programme aura trois boucles imbriquées.

```
from kandinsky import *

# pour les 100 lignes du haut :
for y in range(100):
  # pour les 10 barres verticales:
  for b in range(10):
    # pour les 32 points horizontaux de chaque barre :
    for x in range(b*32,b*32+32):
```

Question 17 Saisie du script (suite) : écrire les deux lignes de définition de la couleur du pixel.

```
c=color(b*28,...,...)
set_pixel(x,y,c)
```

> Pour faire du gris, les paramètres `r`, `g`, `b` doivent être identiques, d'où : `c=color(b*28,b*28,b*28)`

Question 18 Variante : on veut que le dégradé aille du blanc au noir en allant de gauche à droite. Modifier la ligne `c=color(b*28, ..., ...)`

> `c=color(255-b*28,255-b*28,255-b*28)`

6.4.2 Exercice

Mire

Les couleurs orangées du thème NumWorks respectent les proportions suivantes : rouge 100%, vert 75%, et bleu 25%. Reprendre l'exercice en présentant un dégradé de 10 bandes verticales orangées sous la mire de gris précédente.

Indication : il suffit de modifier dans le script précédent la plage des lignes `y`, ainsi que les arguments `g` et `b` de la fonction `color(r,g,b)`.

6.5 Conclusion

On a vu le principe de codage des couleurs d'une image de petite dimension. A raison de 256 niveaux possibles pour chacune des 3 couleurs de base, il faut 3 octets par pixel, et davantage si l'on encode par exemple plus de 1000 niveaux par couleur (on parle de profondeur de couleur). Ce nombre d'octets est à multiplier par le nombre de pixels de l'image, ce qui peut facilement nécessiter plus de 10 Mo par image.

On dispose alors de procédés de compression d'images consistant à

- indiquer par exemple que n pixels qui se suivent ont la même couleur (inutile de tous les coder)
- à faire une table des couleurs présentes dans l'image (on crée une palette) et à coder pour chaque pixel l'emplacement de sa couleur dans la table
- à faire des approximations : on utilise la même couleur pour des couleurs voisines