FONDATION
La main à la pâte
POUR L'ÉDUCATION À LA SCIENCE

Cycle 4

Cryptographie branchée avec Scratch

Une séquence du projet *1,2,3… CODEZ !*

Résumé

Cette séquence de cryptographie optionnelle propose aux élèves de programmer (dans l'environnement Scratch) le chiffrement de César (chiffrer/déchiffrer) et l'analyse fréquentielle (trouver les fréquences de chaque lettre dans un texte et afficher ce résultat sous la forme d'un graphique). Cette séquence permet une appropriation de tous les concepts de programmation utiles au collège : séquence, boucle, test, variable (variables simples et tableaux), fonction…

Séance 8 – Programmer le chiffrement de César (1/4)

Discipline dominante	Mathématiques
Résumé	Les élèves explicitent l'algorithme et listent les étapes qui vont structurer leur projet de programmation du chiffrement de César. Ils programment, dans l'environnement *Scratch*, une première fonction permettant de trouver la lettre correspondant à un rang dans l'alphabet.
Notions *(cf. scénario conceptuel, page 117)*	Langage : • Un programme est l'expression d'un algorithme dans un langage de programmation. • *Scratch* est un environnement de programmation graphique. • Certaines instructions s'exécutent à la suite les unes des autres : on parle de programmation séquentielle. Machine : • Les machines qui nous entourent ne font qu'exécuter des ordres (instructions) exprimés dans des langages particuliers. • En combinant des instructions élémentaires, nous pouvons leur faire exécuter des tâches complexes. • Une variable est un nom que l'on donne à une zone de mémoire. Elle permet de stocker une valeur et de la réutiliser plus tard, ou de la modifier. Bonnes habitudes de programmation : • Lorsqu'un même bloc d'instructions doit être utilisé plusieurs fois dans un programme, il est judicieux de l'intégrer dans une fonction. • Il est préférable d'initialiser une variable dès sa création. • Les variables et les fonctions doivent avoir des noms explicites. Algorithme : • Un algorithme peut contenir des instructions, des boucles, des tests, des variables. • Un test permet de choisir quelle action effectuer si une condition est vérifiée ou non.
Matériel	Pour chaque binôme : • Un ordinateur possédant une connexion Internet (pour utiliser la version en ligne de *Scratch*) ou ayant *Scratch* préinstallé • (facultatif) photocopie de la Fiche 11 (page 164) Pour la classe : • Un vidéoprojecteur, utile lors des mises en commun

Notes pédagogiques

• Dans ce qui suit, nous supposons que les élèves (et le professeur) ont déjà programmé en *Scratch* et en connaissent les bases (créer un programme, l'enregistrer, utiliser les commandes les plus courantes).
– Si c'est le cas, les séances qui suivent ne posent pas de problème majeur.
– Si cela n'est pas le cas, nous renvoyons le lecteur au chapitre « Introduction générale à *Scratch* » (page 68) pour une discussion générale (pourquoi le choix de *Scratch*, quels sont ses avantages et ses limites, comment l'installer, etc.) ainsi que pour une séance de prise en main (familiarisation avec l'interface, exercices d'initiation, etc.).

• Comme expliqué dans le chapitre « Initiation à *Scratch* », les élèves avancent à des rythmes très différents dans ces activités de programmation. Nous encourageons le professeur à leur permettre, autant que possible, d'avancer à leur rythme.
• Pour cette raison, le découpage en séance est purement indicatif et ne doit pas être considéré comme un cadre à suivre par la classe dans son ensemble. À la fin de chaque séance, certains binômes n'auront pas terminé toutes les étapes, tandis que d'autres auront démarré celles des séances suivantes.

Étape 1 : définir les différentes étapes du projet (20 minutes ; collectivement)

Le professeur explique aux élèves qu'ils vont réaliser des programmes permettant d'exécuter plus facilement certaines actions qu'ils ont faites de manière « débranchée » (ou « à la main ») jusqu'à présent. Il leur propose de commencer par le plus simple : le chiffrement de César.
Dans un premier temps, la classe réfléchit collectivement à l'algorithme de ce chiffrement et fait la liste des éléments qui seront nécessaires pour que le programme produise ce que l'on attend de lui :
• Chaque lettre figurant dans le message clair est remplacée par une lettre qui est décalée d'un certain nombre de rangs (la « clé ») dans l'alphabet.
• Le programme doit donc :
– Permettre à l'utilisateur de saisir le message à chiffrer (qui doit être enregistré dans une variable, nommée par exemple « message_clair »)
– Permettre à l'utilisateur de saisir la valeur de la clé (qui doit également être stockée dans une variable, nommée « clé »)
– Stocker l'alphabet dans une variable (que nous nommerons simplement « alphabet »)
– Chiffrer le message, et enregistrer résultat dans une nouvelle variable (nommée, par exemple « message_chiffré »), qui doit être affichée à l'écran.

Notes pédagogiques

• Même si les élèves ont déjà été initiés à *Scratch*, il est peu probable qu'ils soient familiers avec la notion d'algorithme au point de pouvoir verbaliser ce que doit faire un programme avant de le programmer. Pour cette raison, nous conseillons de réaliser cette étape collectivement. À mesure qu'ils progresseront dans le projet, ils pourront prendre de l'autonomie sur ce type de tâche.
• Cette explicitation collective a par ailleurs un autre avantage : elle permet de fixer un cadre général que les élèves vont noter pour, ensuite, pouvoir suivre en autonomie sans forcément avoir besoin de tous avancer au même rythme.

Une fois que la classe sait ce que doit faire le programme, il reste une étape importante à discuter collectivement : comment allons-nous nous y prendre ?
Il s'agit maintenant de découper le projet (une tâche complexe) en plusieurs sous-projets (tâches plus élémentaires).
À titre d'exemple, voici comment on peut « découper » un tel projet :
• Dans un premier temps, on va chercher à ne traiter qu'une seule lettre, et pas un message complet
• Pour savoir comment décaler une lettre d'un certain rang, on doit être capable de trouver quel est le rang d'une lettre donnée dans l'alphabet (par exemple, la lettre n° 3 est C) et, réciproquement, quelle est la lettre qui correspond à un rang donné (la lettre F occupe le rang n° 6). Le professeur conseille

de réaliser une tâche après l'autre et de commencer par la tâche la plus facile : trouver la lettre qui correspond à un certain rang.

• On doit savoir décaler les lettres d'un certain rang mais, quand on dépasse 26 (lettre Z), on doit revenir à 1 (lettre A).

Les étapes du projet, telles qu'elles sont définies par la classe, deviennent donc :

Difficulté	Nom de l'étape	Nature des tâches à réaliser
	**Étape 1** - Définir les étapes du projet	C'est le travail que la classe est en train de faire, étape primordiale au démarrage d'un projet.
	**Étape 2** - Obtenir la lettre d'un certain rang	Écrire un programme qui demande un rang et donne la lettre ayant ce rang dans l'alphabet.
	**Étape 3** - Créer une fonction (1/2)	Apprendre à manipuler les fonctions dans un programme.
	**Étape 4** - Créer une fonction (2/2)	Écrire une fonction qui remplit l'objectif de l'étape 2. Écrire un programme pour tester cette fonction.
	**Étape 5** - Obtenir le rang d'une certaine lettre	Écrire un programme qui demande une lettre et donne son rang dans l'alphabet. En faire une fonction, et la tester.
	**Étape 6** - Chiffrer une lettre	Écrire un programme qui demande une clé, puis une lettre qu'il chiffre avec cette clé. En faire une fonction, et la tester. Transformer cette fonction pour qu'elle fonctionne si le rang + la clé dépasse 26.
	**Étape 7** - Chiffrer le message entier	Écrire un programme qui demande une clé et un message, et qui affiche le message chiffré.

Notes scientifiques

• Découper le travail de programmation en tâches élémentaires permet de tester les différentes parties du programme au fur et à mesure. C'est beaucoup plus facile, et plus sûr, que d'écrire un long programme et de le tester à la fin (dans ce cas, on obtient en général de nombreux bugs qu'il est très difficile de résoudre)

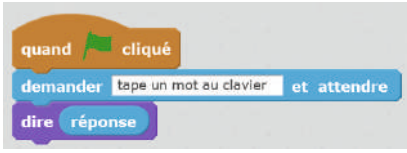
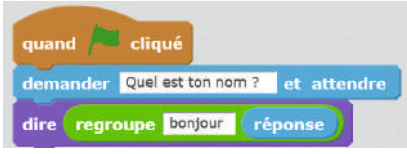
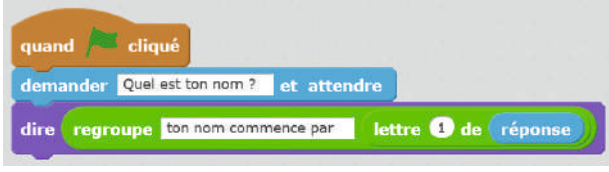
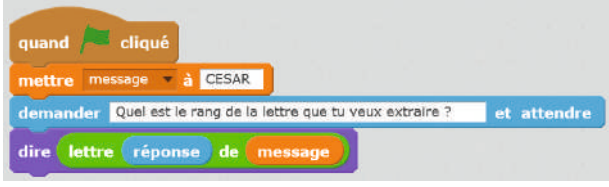
• Travailler à l'aide de fonctions permet, une fois que la fonction est écrite, de passer à autre chose sans avoir besoin de toucher ce morceau de programme.

Étape 2 : obtenir la lettre d'un certain rang dans l'alphabet (50 minutes)

Note pédagogique

Cette première étape est elle-même une tâche complexe dans la mesure où beaucoup d'opérations sont réalisées pour la première fois par les élèves (manipulation des chaînes de caractères et des variables…). Ne pas hésiter à faire de fréquentes mises en commun.

Cette étape nécessite d'utiliser la commande « demander » dans *Scratch*. Disponible dans l'onglet bleu clair intitulé « capteurs », cette commande permet de poser une question et de stocker le résultat dans une variable (déjà existante) appelée « réponse ».
Si les élèves n'ont jamais utilisé cette fonctionnalité, ni manipulé de chaines de caractères dans *Scratch*, nous conseillons de leur proposer les exercices de la Fiche 11 (page 164). Le corrigé est :

Programme 1 	Le programme recopie à l'écran le mot que l'utilisateur a saisi au clavier (et qui a été enregistré dans une variable « réponse », créée automatiquement).
Programme 2 	Le programme répète le nom qui a été saisi au clavier, en le faisant précéder de « bonjour » (avec un espace derrière « bonjour » de façon à ce que les 2 mots ne soient pas collés l'un à l'autre).
Programme 3 	Le programme renvoie la première lettre du nom saisi au clavier.
Programme 4 	Le programme renvoie la lettre correspondant au numéro qui a été saisi au clavier. Par exemple, si l'utilisateur tape « 2 », le programme renvoie la 2e lettre du mot CESAR, soit « E ».

La réponse au problème posé initialement (faire afficher la lettre d'un certain rang de l'alphabet) peut se programmer de la façon suivante :

Si nécessaire, l'enseignant peut découper cette étape en tâches plus élémentaires, en leur demandant :
1. de faire afficher l'alphabet
2. de faire afficher la lettre 13 de l'alphabet
3. de demander le rang, puis de faire afficher la lettre correspondante à ce rang.

Pour éviter d'avoir à répéter plusieurs fois tout l'alphabet, et simplifier la lecture du programme, stocker l'alphabet dans une variable. Le programme devient simplement :

Note scientifique

Créer une variable se fait via l'onglet orange intitulé « données ».

On peut améliorer la réponse et, ce faisant, se familiariser avec la concaténation des chaînes de caractères (qui sera fort utile pour la dernière étape) :

Bilan intermédiaire

La classe établit un bilan d'étape :

- le programme permet de trouver quelle lettre se situe à un rang donné dans l'alphabet
- Certains concepts clés de la programmation ont été vus (ou revus) :

– séquence d'instructions
– variables
– entrées, sorties.

FICHE 11
Exercices de base en Scratch

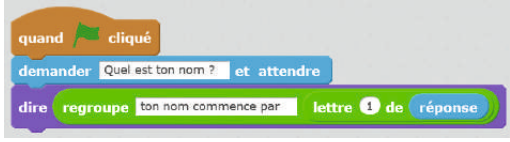
Consigne : Pour chacun de ces programmes :

- écris, à côté de l'illustration, ce que fais le programme ;
- programme-le à ton tour dans *Scratch* pour vérifier ta réponse.

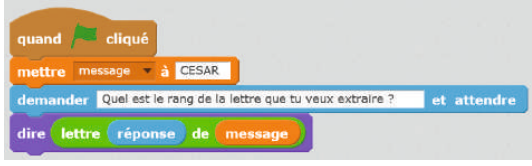
Programme 1

Programme 2

Programme 3



Programme 4



Séance 9 – Programmer le chiffrement de César (2/4)

Discipline dominante	Mathématiques
Résumé	Les élèves modifient leur programme précédent pour introduire une « fonction » (bloc personnalisé).
Notions *(cf. scénario conceptuel, page 117)*	Bonnes habitudes de programmation : • Lorsqu'un même bloc d'instructions doit être utilisé plusieurs fois dans un programme, il est judicieux de l'intégrer dans une fonction.
Matériel	Idem Séance 8 En plus, pour chaque binôme : • photocopie de la Fiche 12 (page 54)

La classe reprend le fil conducteur établi à la Séance 8. En particulier, l'enseignant rappelle qu'il serait préférable de mettre le programme créé précédemment dans une fonction, et d'écrire un programme qui permet de tester cette fonction. *Scratch* permet de créer ses propres fonctions (dans l'onglet violet « ajouter blocs », commande « créer un bloc »).

Note scientifique

Attention : *Scratch* souffre d'une limitation importante : les fonctions ne peuvent pas retourner de valeur. Si l'on souhaite qu'une fonction renvoie un résultat, il faut avoir créé une variable *ad hoc* et faire en sorte que la fonction modifie cette valeur. En informatique, de tels sous-programmes (sans valeur de sortie) sont plutôt appelés des « procédures » : nous employons le mot « fonction » (qui est un abus de langage) par souci de cohérence avec les programmes scolaires.

Étape 3 : créer une fonction (1/2) (15 minutes)

Le professeur propose aux élèves de commencer par des petits exercices leur permettant de se familiariser avec la notion de « fonction », et distribue pour cela la Fiche 12.

Nous conseillons d'organiser une mise en commun après chacun des exercices afin de s'assurer que cette nouvelle notion soit bien comprise par les élèves, tout comme la manière de l'implémenter dans *Scratch*.

Solution des exercices :

• **Programme 1** : ce programme dessine un carré rouge de 100 pixels de coté

• **Programme 2** : ce programme réalise exactement la même tâche, mais en utilisant une « fonction » (un bloc personnalisé, créé par l'utilisateur).

– Ce bloc est défini par l'utilisateur en cliquant sur « créer un bloc » (onglet violet « Ajouter blocs ». Une fois défini, il peut être appelé par le « programme principal » (déclenché par le drapeau vert).

– Pour vérifier que ces 2 premiers programmes font bien la même chose, il peut être utile de repartir d'un écran vierge entre chaque exécution (commande « effacer tout », dans l'onglet vert « Stylo »).

– Pour l'instant, l'utilité de passer par ce bloc personnalisé ne saute pas aux yeux !

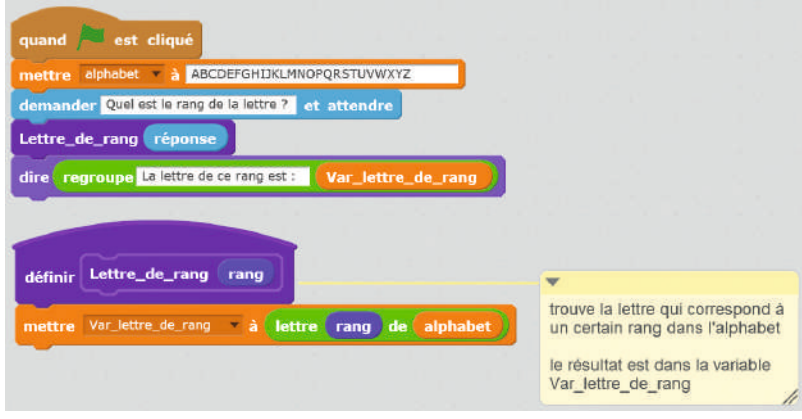
• **Programme 3** :
– Ce programme dessine 3 carrés (rouge, vert, bleu), de même dimension, et décalés de 15 degrés.
– Ici, on commence à comprendre l'intérêt d'avoir défini un bloc personnalisé : cela évite d'avoir à répéter 3 fois la même suite d'instructions.
• **Programme 4** :
– Ce programme dessine 3 carrés (rouge, vert, bleu), mais cette fois-ci de dimensions différentes.
– Pour faire cela, il faut modifier les options du bloc « dessine_carré » : cliquer avec le bouton droit sur son titre, puis « éditer », cocher « options », et « ajouter une entrée nombre ».
– Désormais, ce bloc accepte un « argument » d'entrée. Cette valeur correspond ici à la taille souhaitée du carré. Le programme principal appelle la même fonction 3 fois, en changeant simplement la valeur de cet argument, ce qui a pour effet de dessiner 3 carrés de tailles différentes.
• **Programme 5** :
– Ce programme demande à l'utilisateur d'entrer une longueur, et affiche le périmètre du carré dont le coté vaut cette longueur.
– Note : puisqu'il n'est pas possible, dans *Scratch*, qu'un bloc personnalisé puisse retourner une valeur (il ne s'agit pas d'une « fonction » à proprement parler, mais d'une « procédure »), nous sommes obligés d'utiliser une astuce : on crée une variable dédiée. Cette variable est manipulée dans la fonction, et peut être ensuite utilisée dans le programme principal.

Notes scientifiques

• Pour bien comprendre que la variable créée ne sert qu'à renvoyer le résultat de la « fonction », il est utile de lui donner un nom très proche (mais pas identique) du nom de la fonction. Ainsi, à la lecture du programme, le lien entre les deux devient évident.
• Lorsque l'on crée une fonction (ou « bloc »), on peut donner autant d'arguments qu'on le souhaite, et spécifier leur type (nombre, texte, booléen).

Étape 4 : créer une fonction (2/2) (15 minutes)

Les élèves ayant désormais compris le fonctionnement (et l'intérêt) des blocs personnalisés dans *Scratch*, ils peuvent transformer leur programme créé à la séance précédente pour introduire une fonction, par exemple appelée « Lettre_de_rang ». La variable associée à cette fonction est nommée « Var_lettre_de_rang ».



NB : l'ajout de commentaires est traité en étape 11, page 62.

Les élèves enregistrent leur programme.

Bilan intermédiaire

La classe établit un bilan d'étape :

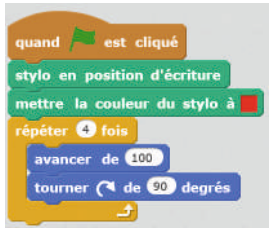
- le programme permet de trouver quelle lettre se situe à un rang donné dans l'alphabet ;
- un nouveau concept-clé de la programmation a été vu : la fonction.

FICHE 12
Créer ses propres blocs dans *Scratch*

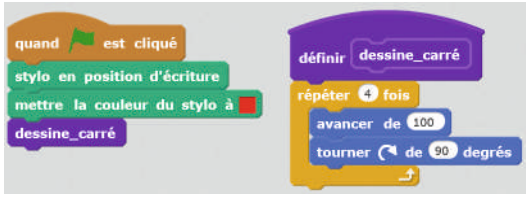
Consigne : Pour chacun de ces programmes :

- écris, à côté de l'illustration, ce que fais le programme ;
- programme-le à ton tour dans *Scratch* pour vérifier ta réponse.

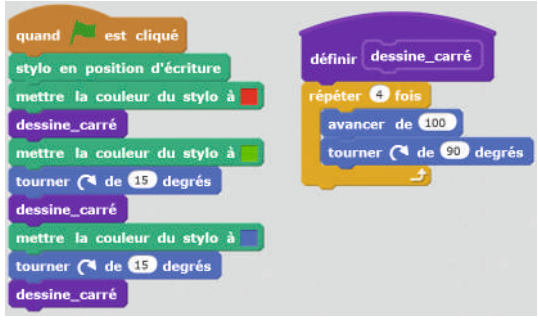
Programme 1



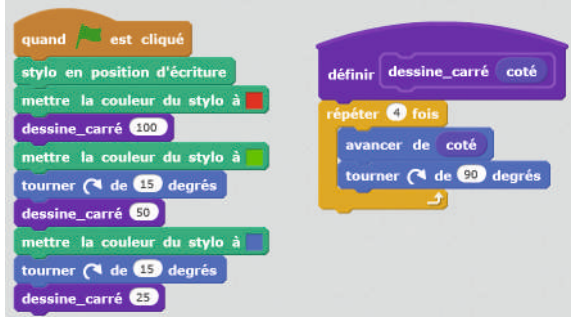
Programme 2



Programme 3



Programme 4



Programme 5

Séance 10 – Programmer le chiffrement de César (3/4)

Discipline dominante	Mathématiques
Résumé	Les élèves avancent leur programme du chiffrement de César : ils sont capables de trouver le rang d'une lettre, puis de chiffrer cette lettre en décalant le rang d'un certain nombre (la clé).
Notions *(cf. scénario conceptuel, page 117)*	Algorithmes : • Une boucle permet de répéter plusieurs fois la même action. • Certaines boucles, dites « itératives », sont répétées un nombre prédéfini de fois. • Un test permet de choisir quelle action effectuer si une condition est vérifiée ou non. • Une condition est une expression qui est soit vraie, soit fausse • On peut utiliser des opérateurs logiques comme ET, OU, NON pour fabriquer des expressions logiques.
Matériel	Idem Séance 8

Le professeur rappelle les différentes étapes du projet. La classe enchaîne par l'étape n° 5.

Note pédagogique

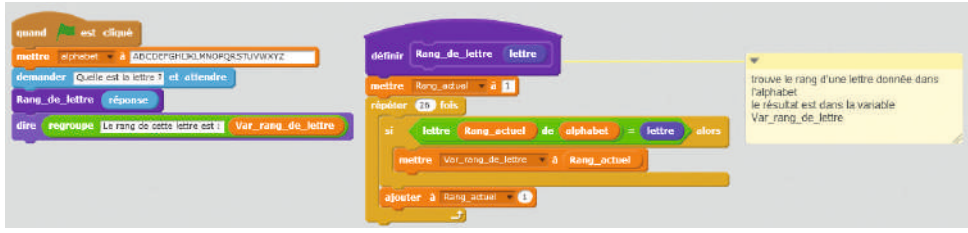
À partir de maintenant, les élèves vont réutiliser des éléments qu'ils ont déjà vus (fonctions, chaînes de caractères) : ils vont donc être davantage autonomes et avancer à leur propre rythme.

Étape 5 : obtenir le rang d'une certaine lettre (20 minutes)

Trouver quel est le rang d'une lettre donnée nécessite de parcourir tout l'alphabet et de tester, pour chaque rang, si la lettre de ce rang est bien la lettre considérée. Il faut donc utiliser une boucle.
Le programme ressemble à :

Ce programme nécessite la création de 2 variables :
• « Rang_de_lettre », qui est notre résultat final ;
• « Rang_actuel », qui sert à se repérer dans la boucle.

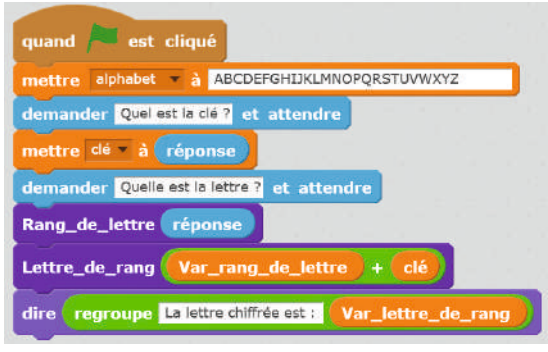
De la même manière qu'à l'étape précédente, les élèves vont devoir créer une fonction qui remplit cette tâche, et écrire un programme qui permet de tester cette fonction. Attention, ici l'argument de la fonction n'est plus un nombre, mais une chaîne de caractères. On note au passage qu'il a fallu renommer la variable Rang_de_lettre en Var_rang_de_lettre, pour rester en cohérence avec notre convention de nommage.



NB : l'ajout de commentaires est traité en étape 11, page 176.

Étape 6 : chiffrer une lettre (20 minutes)

Au cours de cette étape, les élèves vont devoir utiliser les deux fonctions qu'ils ont définies précédemment afin de trouver le rang d'une lettre, lui ajouter un certain nombre (la clé), et renvoyer la lettre correspondant à ce nouveau rang décalé.
La difficulté tient dans l'utilisation de la fonction « modulo » (voir plus bas à la fin de cette étape).



À noter : on ne touche plus aux deux fonctions « Rang_de_lettre » et « Lettre_de_Rang ».
On remarque que ce programme fonctionne bien, sauf lorsque le rang de la lettre + la clé dépasse 26. En effet, si la clé vaut 3 et que l'on veut chiffrer la lettre Y, il faut la remplacer par la lettre de rang 25+3 = 28 cela correspond à la lettre B puisqu'à la place du rang 27 on reboucle sur le rang 1 (lettre A).
Le professeur introduit la fonction « modulo » dont il explique le principe. La fonction modulo *(n)* renvoie une valeur qui correspond au reste de la division euclidienne d'un nombre par *n*. Ainsi :

- 13 modulo 3 vaut 1, car 13 = 4 x 3 + 1 (en français : le reste de la division de 13 par 3 vaut 1)
- 1 modulo 26 vaut 1
- 26 modulo 26 vaut 0
- 27 modulo 26 vaut 1
- 28 modulo 26 vaut 2
- etc.

Cette fonction s'approche du but recherché. Cependant, il n'existe pas de lettre de rang = 0 (nous commençons à compter les rangs à partir de 1). Pour cette raison, il faut d'abord retrancher 1 à la

variable, puis utiliser la fonction modulo, puis rajouter 1. Le programme est alors modifié de la façon suivante.

Notes pédagogiques

• Il est très peu probable que les élèves trouvent cette astuce par eux-mêmes. Ne pas hésiter à la leur présenter et vérifier collectivement qu'elle répond bien au problème qui nous préoccupe.

• On peut éventuellement se passer de la fonction « modulo » en effectuant un test sur le rang de la lettre et la clé. SI rang+clé <27, ALORS prendre la lettre (rang+clé), SINON prendre la lettre « rang+clé-26 ». Cela fonctionnera si le rang + la clé donne un nombre compris entre 1 et 52, ce qui est moins satisfaisant que la fonction modulo (qui donnera le bon résultat pour tout nombre entier).

Bilan intermédiaire

La classe établit un bilan d'étape :

• A ce stade, le programme permet de chiffrer une lettre (mais pas encore un message complet).

• La notion de « boucle », essentielle en programmation, a été vue (ou revue).

• Les élèves ont également appris à utiliser des fonctions mathématiques dans un programme (addition, fonction modulo).

Séance 11 – Programmer le chiffrement de César (4/4)

Discipline dominante	Mathématiques
Résumé	Les élèves terminent leur programme du chiffrement de César, qui permet désormais de chiffrer un message entier, en tenant compte des espaces, de la ponctuation et des accents. Ils apprennent également à commenter un programme de façon à le rendre lisible et compréhensible. Une activité de prolongement est proposée afin de les aider à manipuler les opérateurs logiques.
Notions *(cf. scénario conceptuel, page 117)*	Algorithmes : • On peut utiliser des opérateurs logiques comme ET, OU, NON pour fabriquer des expressions logiques. Machines : • Pour la plupart des tâches de chiffrement ou de cryptanalyse, un ordinateur est bien plus efficace qu'un être humain. « Bonnes pratiques de programmation » : • Placer des commentaires dans un programme en facilite la lecture et le partage avec autrui.
Matériel	Idem Séance 8 En plus, pour chaque élève : • Fiche 13, page 179

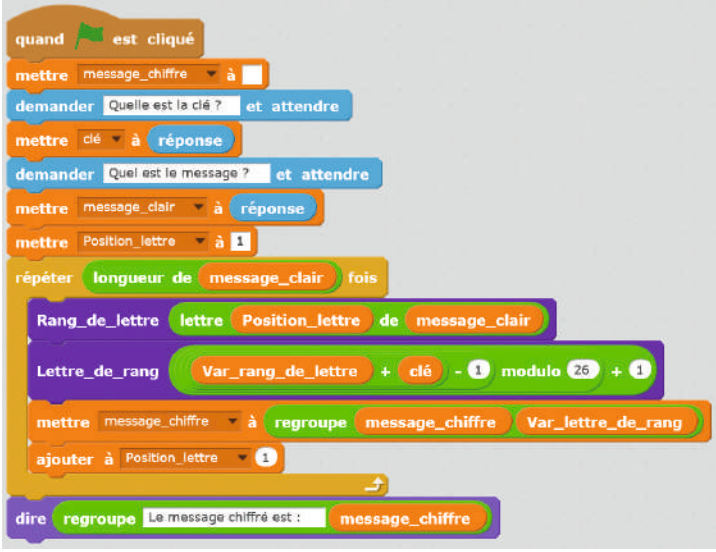
Étape 7 : chiffrer un message entier (20 minutes)

Cette étape ne présente pas de difficulté majeure. Tout comme à l'étape 3, il faut introduire une boucle car on ne chiffre pas une seule lettre, mais toutes les lettres du message.

Le programme introduit deux nouvelles variables : « Message_clair » (qui est le message que l'on veut chiffrer), et « Position_lettre », qui indique quelle lettre nous sommes en train de chiffrer dans ce message.

Le message chiffré est obtenu en regroupant les lettres chiffrées à chaque étape de la boucle.

Un programme possible est donc :



Étape 8 : utiliser le programme ! (10 minutes)

Il ne faut pas perdre de vue que, si l'on a écrit un programme, c'est pour l'utiliser après ! Les élèves peuvent prendre quelques minutes pour « jouer » avec : ils peuvent d'ailleurs vérifier que ce programme peut être utilisé à la fois pour chiffrer et pour déchiffrer des messages. En effet, pour déchiffrer un message qui a été chiffré à l'aide de la clé 3, il suffit d'utiliser le même programme, mais avec une clé égale à 26-3 = 23 (cela décale les lettres du même nombre de rangs, mais dans l'autre sens ce qui ramène donc au message d'origine).

Attention : pour l'instant, leur programme ne sait pas gérer ni la ponctuation (espace inclus), ni les caractères accentués ! Ce sera le cas après les 2 étapes (optionnelles) suivantes.

La classe en conclut que le chiffrement de César est un chiffrement dit « symétrique » : connaître la clé qui a été utilisée pour chiffrer le message permet de le déchiffrer.

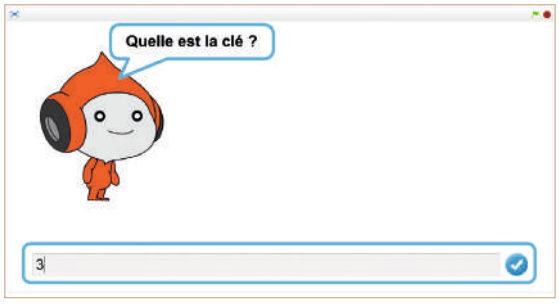
Notes scientifiques

• Les chiffrements symétriques, quels que soient leur complexité, possèdent tous le même point faible : il faut communiquer la clé à son interlocuteur, pour qu'il soit à même de déchiffrer le message. Si cette clé est interceptée par un « espion », ce dernier peut alors accéder au contenu du message.

• Il existe des méthodes de chiffrement dites « asymétriques », qui nécessitent des clés différentes pour chiffrer et déchiffrer le message. La complexité de ces algorithmes (comme RSA par exemple) dépasse largement ce qu'on peut attendre d'élèves de collège. En revanche, il peut être intéressant de se pencher sur la question de l'échange des clés. De tels algorithmes font appel à des clés publiques et des clés privées. On peut chiffrer un message à destination d'un interlocuteur connaissant sa clé publique, message que lui seul sera en mesure de déchiffrer à l'aide de sa clé privée.

Démonstration du programme :

1 : L'utilisateur donne la clé avec laquelle il souhaite chiffrer son message



2 : L'utilisateur tape le message à chiffrer

3 : Le programme affiche le message chiffré

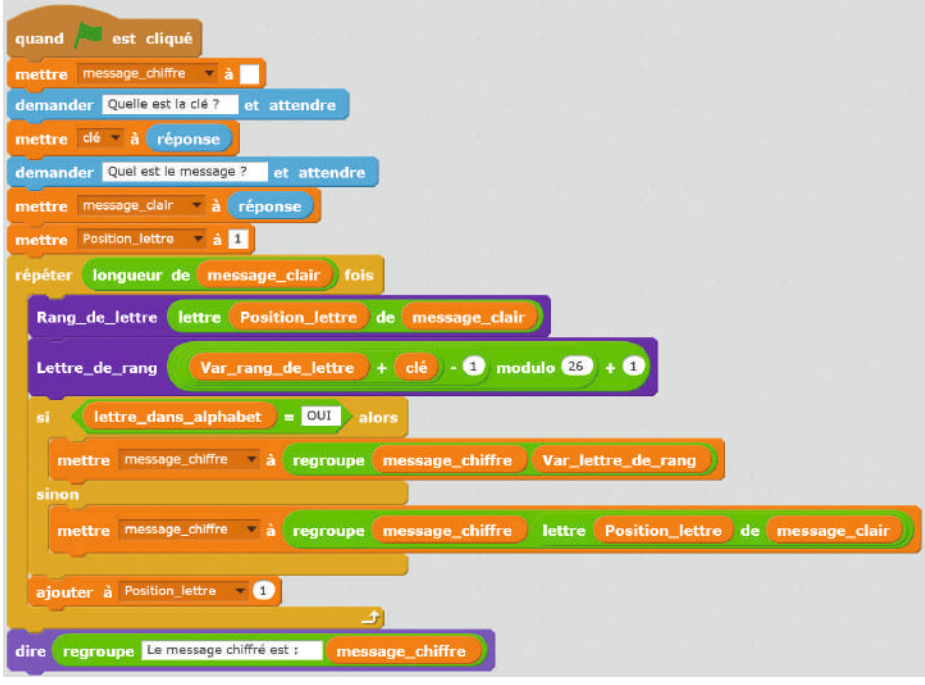
Dans le cas, probable, où certains groupes d'élèves ont pris de l'avance par rapport à d'autres, le professeur donne une liste de nouvelles tâches permettant d'améliorer le programme :

Difficulté	Nom de l'étape	Nature des tâches à réaliser
	Etape 9 – prendre en compte les espaces et la ponctuation	Modifier le programme pour qu'il ne chiffre pas les espaces ni les caractères de ponctuation, qui doivent être recopiés tels quels.
	Etape 10 – gérer les caractères accentués	Créer une fonction qui remplace les caractères accentués du message clair par leur équivalent sans accent. Modifier le programme principal de façon à ce qu'il chiffre le message sans accent.
	Etape 11 – commenter son programme	Ajouter des commentaires à son programme pour le rendre plus lisible.

À noter que, si les étapes 9 et 10 sont optionnelles, l'étape 11 doit être réalisée par tous les élèves.

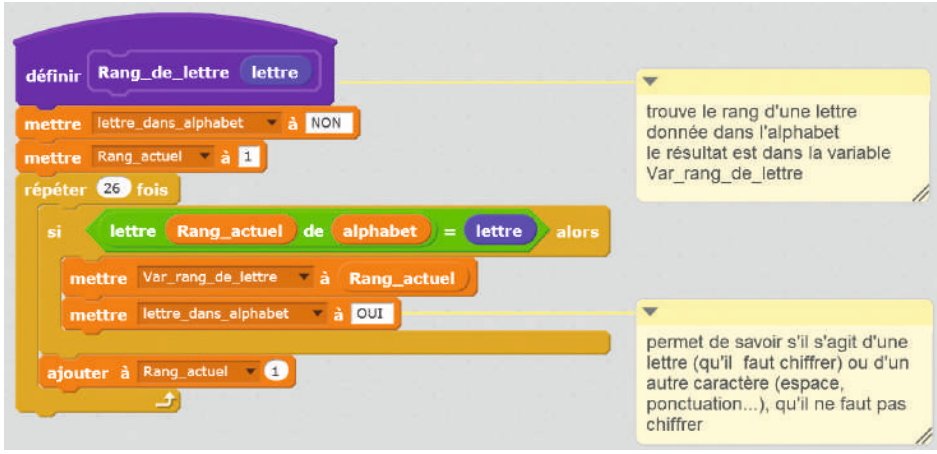
 Étape 9 : prendre en compte les espaces et de la ponctuation (20 minutes)

À ce stade, notre programme ne fonctionne pas si le message clair contient autre chose qu'une lettre spécifiée dans notre alphabet simplifié (espace, signe de ponctuation…).
Pour résoudre le problème, il suffit de spécifier au programme de ne pas chiffrer les caractères qui ne font pas partie de l'alphabet. On introduit une variable (nommée par exemple « lettre_dans_alphabet »). Si cette variable vaut « oui », alors le programme fonctionne normalement (on chiffre la lettre). Si cette variable vaut « non », le programme se contente de recopier le caractère non chiffré.



Le programme décale toutes les lettres de la valeur de la clé, sans toucher aux caractères qui ne sont pas dans l'alphabet.

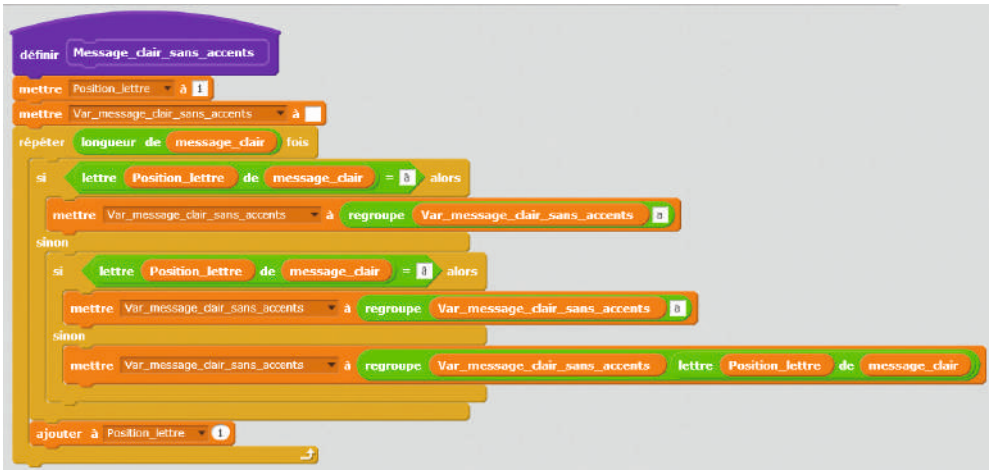
Il faut vérifier, dans la fonction « Rang_de_lettre » si la lettre a bien été trouvée ou non. La fonction devient donc :



Cette version, modifiée, de la fonction « Rang_de_lettre » permet de détecter si le caractère est présent dans l'alphabet ou non.
NB : l'ajout de commentaires est traité en étape 11, ci-dessous.

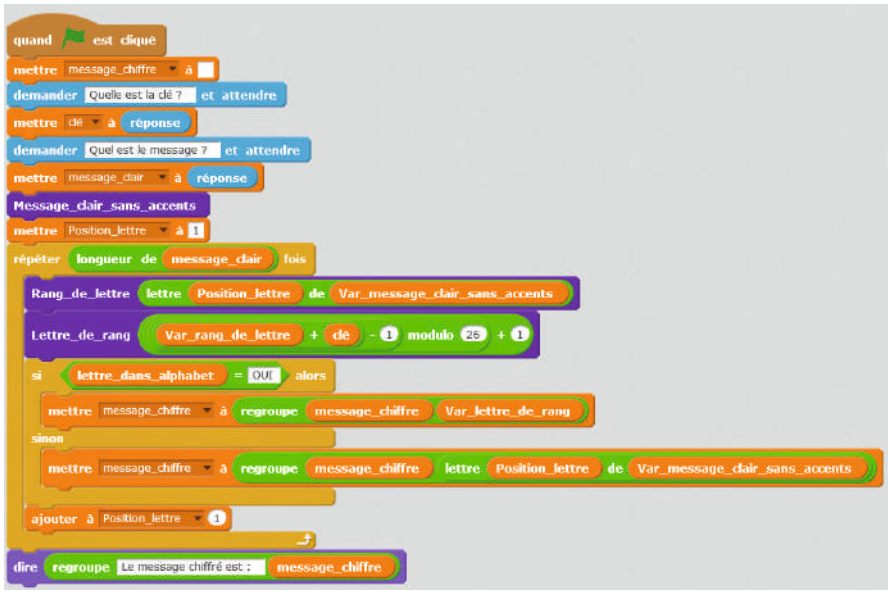
Étape 10 : gérer les lettres accentuées (20 minutes)

Le programme, dans son état actuel, ne prend pas en compte les caractères accentués. Bien que cela ne soit pas indispensable, la classe peut décider de consacrer 20 minutes supplémentaires pour pallier ce défaut. On pourra par exemple remplacer tous les « à » et les « â » du message clair par des « a », etc. Ce travail nécessite d'imbriquer plusieurs tests les uns dans les autres ; ce qui ne présente pas de difficulté particulière et peut être réalisé en autonomie par les élèves. Cela se fait à travers une nouvelle fonction nommée « Message_clair_sans_accent ». Cette fonction recopie tous les caractères non accentués tels quels, mais remplace les caractères accentués par leurs équivalents sans accents.



Pour une meilleure visibilité, nous n'avons affiché ici qu'une version incomplète de cette fonction (elle ne traite que les « à » et les « â »). Il faut imbriquer d'autres boucles SI… ALORS… SINON pour traiter les autres caractères accentués.

Bien entendu, il faut remplacer la variable « Mmessage_clair » par « Var_message_clair_sans_accent » dans le programme principal, qui devient :



Note pédagogique

Plutôt que d'imbriquer les SI ALORS dans la fonction « Message_clair_sans_accents », on pourrait regrouper certains tests à l'aide d'opérateurs logiques OU. Par exemple : SI lettre = « à » OU lettre = « â » ALORS remplacer par « a ». Nous proposons à cet effet une fiche documentaire permettant de se familiariser avec les opérateurs logiques (*cf.* Fiche 13).

Étape 11 : commenter son programme (10 minutes)

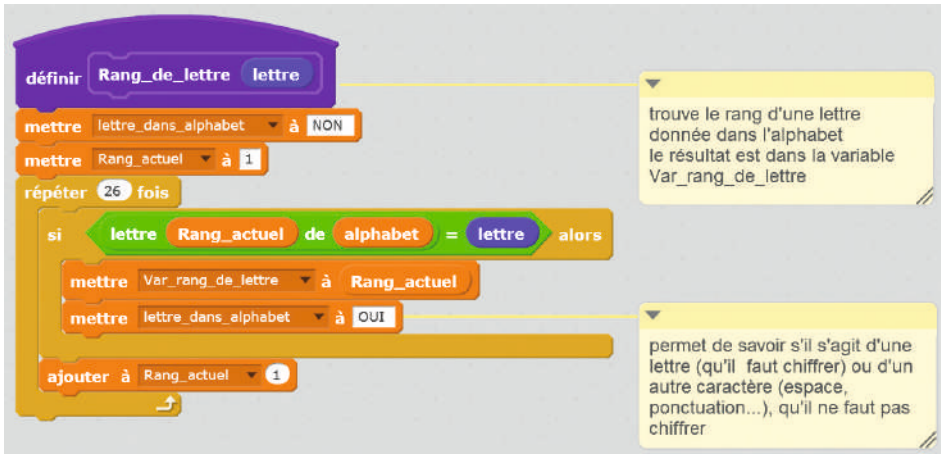
Le programme fonctionne correctement, mais il est vite devenu complexe. Il y a fort à parier qu'il sera difficilement compréhensible par d'autres personnes que ceux qui l'ont produit (ou par ses producteurs, après quelques mois passés sans le lire).

Le professeur explique donc que commenter ses programmes, les aérer et les organiser de façon logique est une bonne habitude.

Note pédagogique

Nous aurions pu introduire cet aspect plus tôt, et avoir ainsi un programme « propre » dès le départ, mais cela nous semble plus formateur de laisser les élèves se confronter à la difficulté.

Ci-dessous, la fonction « Rang_de_lettre » commentée :



Prolongement conseillé (20 minutes)

Les élèves peuvent réaliser une activité débranchée pour manipuler quelques opérateurs logiques. Les exercices fournis sur la Fiche 13 peuvent servir de support à cette activité. Il est préférable que les élèves y répondent d'abord « à la main », quitte, en fin de séance, à réaliser des programmes qui permettront de vérifier leurs réponses.

Les réponses aux questions posées sont :

• **Exercice 1 :**

– métro

– marche

– marche

– vélo (on remarque que les 2 derniers programmes sont équivalents, même si leur forme diffère. Le résultat n'est pas le même car une des variables a changé de valeur)

• **Exercice 2 :**

– 12 (l'unité n'est pas précisée on peut supposer qu'il s'agit d'euros)

– 7 + 12 - 3 = 16

– 7 + 12 + 6 - 3 = 22

– 7 + 6 = 13 (pas de réduction dans ce cas !)

Prolongement optionnel (2 séances)

Le programme réalisé par les élèves permet simplement de décaler les lettres d'un message d'un certain rang (chiffrement de César). Les élèves peuvent être tentés d'aller plus loin et de programmer un chiffrement par substitution mono-alphabétique. Un tel programme :

• demande un mot-clé servant au chiffrement ;

• le transforme en un mot-clé pangrammique (suppression des espaces, doublons) ;

• génère un alphabet chiffré ;

• demande à l'utilisateur quel texte il souhaite chiffrer ;

• chiffre ce texte par substitution mono-alphabétique.

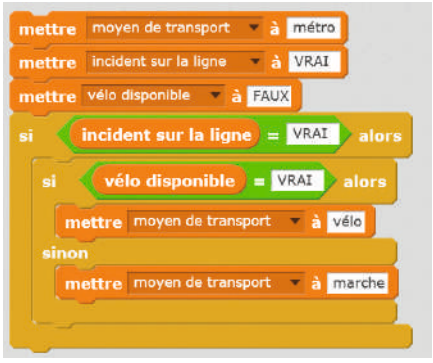
Si les élèves sont très à l'aise en programmation *Scratch*, alors le professeur peut leur proposer un tel projet, qui sera néanmoins un challenge, avec une utilisation massive des tests, fonctions, opérateurs logiques, boucles imbriquées les unes dans les autres…

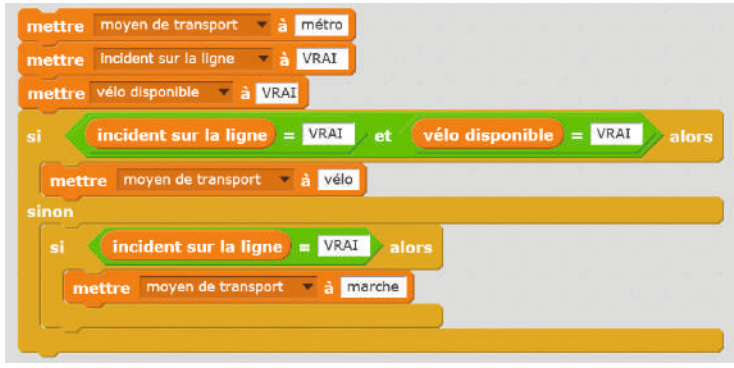
L'analyse de fréquence, que nous proposons dans les séances suivantes, est plus facile (sauf éventuellement ce qui concerne le tracé du graphique).
Nous fournissons néanmoins à l'enseignant un programme complet et commenté de chiffrement par substitution mono-alphabétique afin de lui permettre :

- de générer facilement des exercices (et leurs corrigés) pour s'entrainer à cette méthode de chiffrement
- de mettre un œuvre un tel projet s'il le souhaite avec ses élèves. Ils pourront soit le programmer in extenso, soit programmer seulement certaines parties, et examiner le programme pour les parties les plus difficiles : analyser un programme existant pour comprendre comment il marche est aussi un travail intéressant.

FICHE 13
Manipuler quelques opérateurs logiques

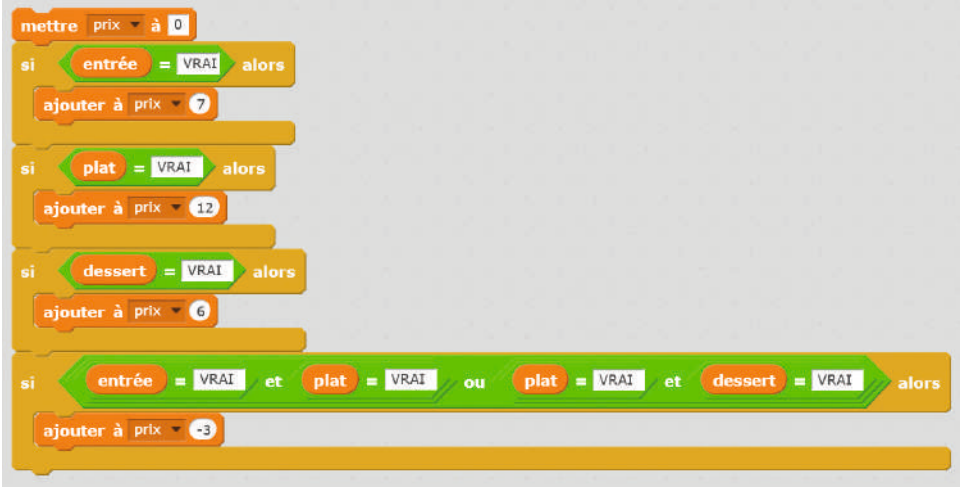
Exercice 1 : Quel est le moyen de transport utilisé dans chacun des cas ?





Exercice 2 : Au restaurant, quel sera le prix du repas si on mange

- un plat uniquement
- une entrée et un plat
- une entrée, un plat et un dessert
- une entrée et un dessert



Séance 12 – Programmer l'analyse fréquentielle

Discipline dominante	Mathématiques
Résumé	Les élèves élaborent un nouveau programme qui permet de calculer les fréquences de chaque lettre d'un message, de façon à faciliter sa cryptanalyse. Ils apprennent à manipuler des variables de type « liste ».
Notions *(cf. scénario conceptuel, page 117)*	Algorithmes : • Certaines boucles, dites « conditionnelles », sont répétées jusqu'à ce qu'une condition soit remplie.
Matériel	Idem séances précédentes En plus, pour chaque binôme : • Fiche 14 (page 185)

Le professeur propose à la classe de produire un nouveau programme, qui va aider à la cryptanalyse : il s'agit de construire l'histogramme de fréquences d'un texte, de façon automatisée. Ce travail se déroule en 2 temps :

• Créer un tableau contenant les fréquences de chaque lettre du message. Cette étape ne présente pas de difficulté majeure et fait l'objet de la présente séance.

• Créer un graphique permettant de visualiser ces fréquences. Cette étape, qui fait l'objet des 2 séances suivantes, est plus difficile (mais fort intéressante, notamment pour travailler les fonctions et leur représentation graphique !). Elle est considérée comme optionnelle.

Note pédagogique

L'emploi du terme « fréquence » est ici un abus de langage, car ce que l'on calcule, ce sont des nombres d'occurrences plutôt que des fréquences. Nous gardons néanmoins ce terme pour faire le lien avec la méthode de cryptanalyse appelée « analyse fréquentielle ».

Étape 1 : définir les différentes étapes du projet (15 minutes)

Dans un premier temps, les élèves réfléchissent à l'algorithme permettant de compter la fréquence d'une lettre donnée dans un texte. En fonction de leur aisance, le professeur peut les faire travailler en binôme, en groupe (suivi d'une mise en commun) ou conduire un travail collectif en classe entière.

• Pour compter le nombre de « A » présents dans un texte, la méthode est :

– Parcourir le texte, de la première à la dernière lettre.

– Pour chaque lettre examinée, tester si cette lettre est un « A ». Si oui, ajouter « 1 » à la variable qui compte les occurrences de cette lettre.

• Pour compter la fréquence de toutes les lettres d'un texte :

– Il suffit de généraliser la méthode ci-dessus et de faire une boucle portant sur toutes les lettres du texte.

– Les valeurs vont devoir être enregistrées dans un tableau de 26 cases (la première case contiendra la fréquence de la lettre « A », la seconde celle de la lettre « B »…). Le professeur explique que ceci est possible en *Scratch*, à l'aide d'un type de variable appelé « liste ».

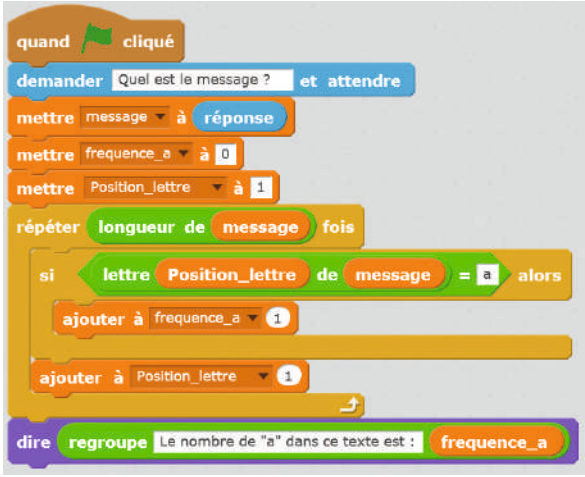
Voici un exemple de découpage du projet en différentes étapes (la réalisation du graphique sera traitée à la séance suivante) :

Difficulté	Nom de l'étape	Nature des tâches à réaliser
●	**1** - Définir les étapes du projet	C'est le travail que la classe est en train de faire.
●	**2** – Compter la fréquence d'une lettre dans un texte	Écrire un programme qui demande un texte et compte le nombre d'occurrences de « A » dans ce texte. Généraliser ce programme à une lettre quelconque de l'alphabet et en faire une fonction. Écrire un programme pour tester cette fonction.
●	**3** – Remplir un tableau de fréquence	Écrire un programme qui demande un texte, puis compte le nombre d'occurrences de chaque lettre dans ce texte, et stocke les valeurs dans une variable de type « liste ». En faire une fonction, et la tester.
●	**4** – Gérer les lettres accentuées	Modifier le programme pour qu'il remplace les caractères accentués avant de procéder à l'analyse de fréquence.

● Étape 2 : calculer la fréquence d'une lettre dans un texte (30 minutes)

Cette étape ne pose pas de difficulté car elle reprend les mêmes notions que les 4 séances de programmation précédentes sur le chiffrement de César.

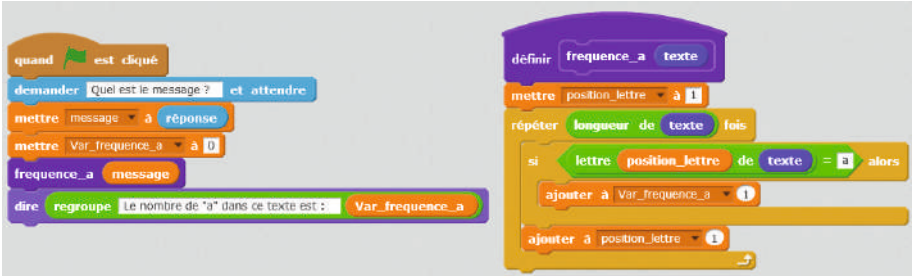
Un programme qui compte le nombre de « A » présents dans un texte s'écrit :



Note scientifique

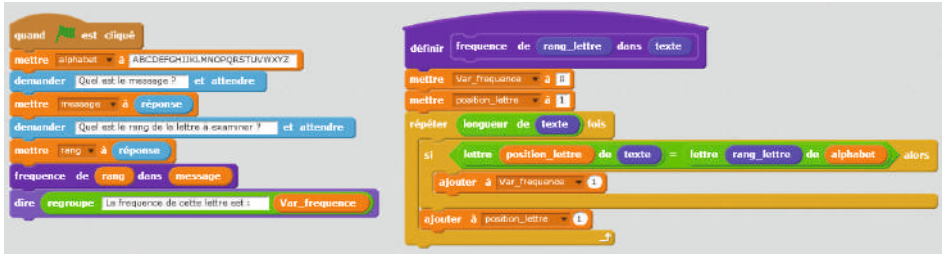
Rappel : *Scratch* n'est pas sensible à la casse. Les lettres « a » et « A » sont considérées comme identiques dans les opérations faisant intervenir des chaines de caractères.

Exprimé à l'aide d'une fonction, le programme devient :



Note : L'utilisation d'une fonction nécessite que l'on renomme notre variable « frequence_a » en « Var_frequence_a », si l'on souhaite respecter la convention utilisée jusqu'à présent.

Si maintenant on souhaite calculer la fréquence, non pas de la lettre « a », mais de n'importe quelle lettre de l'alphabet, on peut généraliser cette approche en remplaçant « a » par « élément (rang) de alphabet », où rang est un nombre entre 1 et 26 et alphabet une variable contenant les 26 lettres, dans l'ordre.



Pour modifier les propriétés de la fonction (la renommer en « frequence » et lui ajouter une seconde variable d'entrée), il faut effectuer un clic droit sur la définition de cette fonction et choisir « éditer ». On note que la variable « Var_frequence_a » a également été renommée en « Var_frequence ».

Étape 3 : remplir un tableau de fréquences (30 minutes)

Remplir le tableau de fréquence de toutes les lettres de l'alphabet nécessite une variable de type « liste ». Si les élèves n'ont jamais manipulé de listes en *Scratch*, ne pas hésiter à leur montrer comment créer, initialiser et utiliser de telles variables. C'est l'objet de la Fiche 14, qui propose un petit exercice guidé. Nous conseillons d'effectuer le premier exercice collectivement, puis de faire des mises en commun à chaque étape, de façon à vérifier la bonne appropriation de cette nouvelle notion par les élèves.

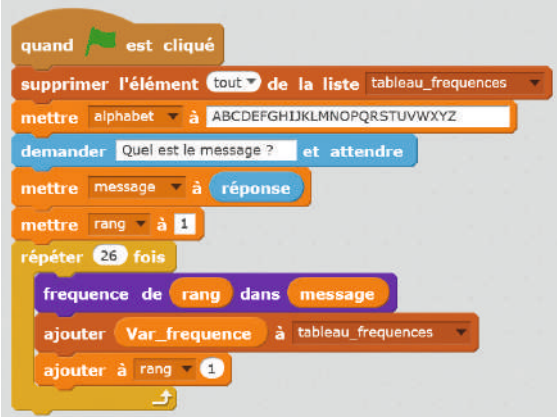
Réponses :

• **Programme 1** : ce programme demande à l'utilisateur de taper 3 textes, et stocke ces textes dans une variable de type « liste » (un tableau à une colonne, affiché à l'écran). Par défaut, elle est vide (« *empty* »), et sa longueur est 0.

• **Programme 2** : l'instruction « supprimer » ajoutée en début de programme permet de réinitialiser la liste.

• **Programme 3** : le programme ne s'arrête de remplir la liste que lorsque l'utilisateur entre le mot « OK ». La liste contient donc tous les éléments cités par l'utilisateur (dont le « OK » final).

• **Programme 4** : Ce programme supprime le dernier élément de la liste (qui est le « OK » vu précédemment), et affiche la longueur (nombre d'éléments) de cette liste (sans le « OK », qui a été supprimé). Il envoie alors un message, qui peut déclencher un autre programme (*cf.* ci-dessous).
• **Programme 5** : Ce programme, déclenché à la fin du précédent, récapitule la liste (le chat affiche, un par un, les éléments de la liste).
Les élèves savent désormais comment créer, initialiser et manipuler une liste (ajouter des éléments, en supprimer, afficher la longueur de la liste). Ils peuvent donc reprendre le programme réalisé à l'étape précédente, et le mettre à jour pour calculer les fréquences de toutes les lettres de l'alphabet (ces fréquences sont stockées dans une variable de type « liste »).



Modification du programme principal pour calculer les fréquences de toutes les lettres de l'alphabet.
Note : nous n'avons pas besoin de modifier la fonction « frequence_de_rang_dans_message ».

Les élèves vérifient le bon fonctionnement du programme en le testant sur des textes courts, sur lesquels ils peuvent calculer rapidement les fréquences à la main.

Note pédagogique

Ne pas hésiter à poser de petits défis aux élèves en avance sur le reste de la classe. Ici, par exemple, on peut leur demander d'initialiser la liste avec des 0 puis d'incrémenter la case adéquate chaque fois qu'une lettre est trouvée dans le texte (*cf.* second algorithme, dans la note scientifique ci-dessous).

Notes scientifiques

Pour établir le tableau des fréquences, 2 algorithmes sont possibles :
• **Premier algorithme** (qui est illustré ici)
Pour la lettre A, on parcourt tout le texte, à la recherche de cette lettre, et on incrémente le compteur quand on la trouve. On boucle ensuite sur les 26 lettres de l'alphabet
• **Second algorithme**
Pour la première lettre du texte, on parcourt l'alphabet (pour trouver son rang) et on incrémente le compteur correspondant à ce rang. On boucle ensuite pour toutes les lettres du texte. Cette seconde méthode est légèrement plus efficace, car elle évite d'avoir à parcourir tout le texte pour chaque lettre de l'alphabet, y compris une lettre absente du texte. Cependant, cette méthode ne permet pas d'utiliser la fonction que nous avons introduite à l'étape précédente, intéressante car moins difficile.

Étape 4 (optionnelle) : remplacer les lettres accentuées (15 minutes)

Cette étape ressemble fortement à celle réalisée pour le chiffrement de César (*cf.* page 175) ; elle permet donc aux élèves de renforcer leur compréhension des notions impliquées et pose moins de difficulté que la première fois.

Les deux séances qui suivent, optionnelles, permettent de visualiser ce tableau de fréquences sous la forme d'un graphique.

FICHE 14
Créer et manipuler des « listes » dans Scratch

Consigne : Pour chacun de ces programmes :
- écris ce que fait le programme ;
- programme-le à ton tour dans *Scratch* pour vérifier ta réponse.

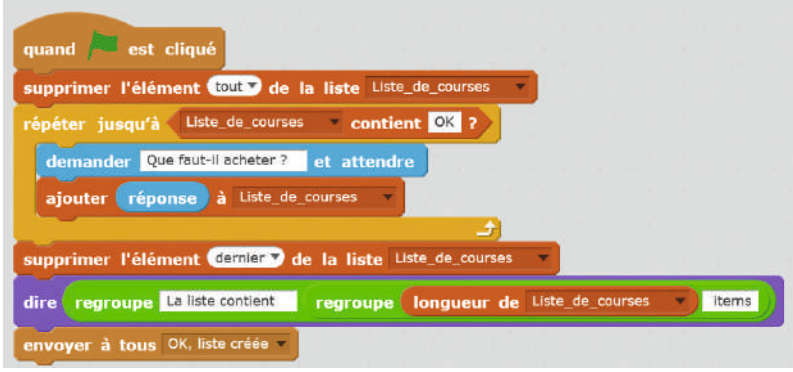
Programme 1 (regarde ce qu'il se passe si tu cliques plusieurs fois de suite sur le drapeau vert).

Astuce : créer une nouvelle variable de type « liste » se fait via l'onglet orange « données », puis « créer une liste ».

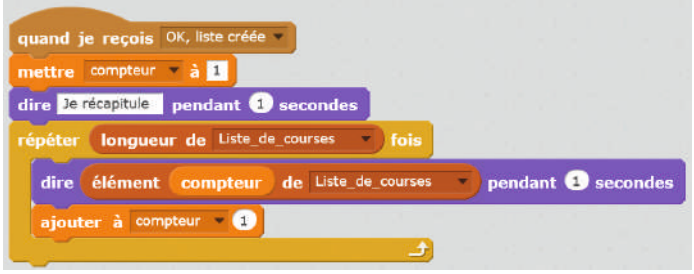
Programme 2

Programme 3

Programme 4



Programme 5



Séance 13 – (optionnelle) Programmer l'affichage de l'histogramme des fréquences (1/2)

Discipline dominante	Mathématiques
Résumé	Les élèves perfectionnent leur programme précédent pour qu'il affiche l'histogramme des fréquences du texte analysé. Ils tracent les axes du graphique et utilisent des costumes (notions propres à *Scratch*) afin de légender l'axe des abscisses.
Notions *(cf. scénario conceptuel, page 117)*	Idem séances précédentes
Matériel	Idem séances précédentes

Note pédagogique

Dans *Scratch*, chaque lutin possède un « stylo », ce qui lui permet (s'il l'abaisse) de tracer des traits. Cette fonctionnalité est très intéressante pour réaliser des figures géométriques ou, comme ici, tracer des courbes.

Étape 1 : définir les différentes étapes du projet (10 minutes)

Il est très peu probable que des élèves de cycle 4, qui découvrent tout juste les fonctions (en mathématiques) et leur représentation graphique, soient capables d'expliciter, tous seuls, un algorithme permettant de créer un graphique et de dessiner des courbes.
Nous conseillons donc de réaliser cette étape collectivement.

Difficulté	Nom de l'étape	Nature des tâches à réaliser
●	**Étape 1** : définir les différentes étapes du projet	• C'est le travail que la classe est en train de faire.
●	**Étape 2** : tracer les axes du graphique	• Créer un nouveau lutin pour accueillir les programmes liés au tracé du graphique. • Définir une position d'origine. • Tracer un axe horizontal partant de cette position. • Tracer un axe vertical partant de cette position. • Mettre ce tracé dans une fonction, et appeler cette fonction depuis le programme principal du lutin.
●	**Étape 3** : placer les graduations sur le graphique	• Afficher les 26 lettres de l'alphabet sur l'axe des abscisses. • En faire une fonction, appelée par le programme principal.
●	**Étape 4** : tracer l'histogramme	• Pour chaque lettre, partir de l'axe des abscisses (à la position correspondant à cette lettre), et tracer un trait vertical, allant jusqu'à l'ordonnée correspondant à la fréquence de cette lettre.
●	**Étape 5** : normaliser l'histogramme	• Repérer la lettre la plus fréquente. • Faire en sorte que la fréquence de cette lettre occupe toujours toute la hauteur du graphique.

	Étape 6: graduer l'axe des ordonnées	• Afficher un trait horizontal, sur l'axe des ordonnées, pour chaque valeur possible de la fréquence des lettres.
	Étape 7: analyser quelques textes	• Tester le programme à l'aide de quelques textes courts.

Les étapes 5 et 6 permettent d'améliorer le rendu final du graphique, mais ne sont pas indispensables.

Étape 2 : tracer les axes du graphique (15 minutes)

Cette étape ne contient pas de difficulté majeure. Si les élèves ont déjà manipulé de stylo dans *Scratch*, et s'ils sont familiers avec les représentations graphiques (axes des abscisses et ordonnées), ils y arrivent sans problème. Dans le cas contraire, ne pas hésiter à les guider légèrement.

Notes pédagogiques

• On peut choisir de mettre des valeurs « en dur » dans le programme, comme les coordonnées du point d'origine, ou la longueur des axes, etc. Nous le déconseillons car, dans ce cas, le programme contient des données numériques qu'il devient difficile d'interpréter. Par ailleurs, si l'on souhaite changer une de ces valeurs, il faut penser à la changer partout où elle apparait, ce qui peut facilement être source d'erreur.
• Pour ces raisons, nous conseillons d'utiliser des variables *ad hoc*. Dans les exemples qui suivent, nous avons défini les variables suivantes :
– X0 et Y0 : coordonnées du point d'origine du graphique (idéalement, en bas à gauche de l'écran).
– Xmax et Ymax : abscisse et ordonnée maximales (auxquelles on stoppe le tracé des axes).
• Par ailleurs, afin de ne pas surcharger le programme en cours, nous conseillons de placer ce nouveau programme dans un nouveau lutin (qu'on peut nommer « initialise graphique » par exemple).

Étape 3 : placer les graduations sur le graphique (20 minutes)

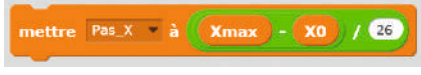
Cette étape n'est pas compliquée en soi, mais elle nécessite de faire appel à une nouvelle instruction : estampiller (qui permet d'utiliser le lutin comme un tampon encreur : une trace définitive est laissée à l'écran, avec l'apparence du lutin au moment où l'instruction « estampiller » est utilisée.

Note pédagogique

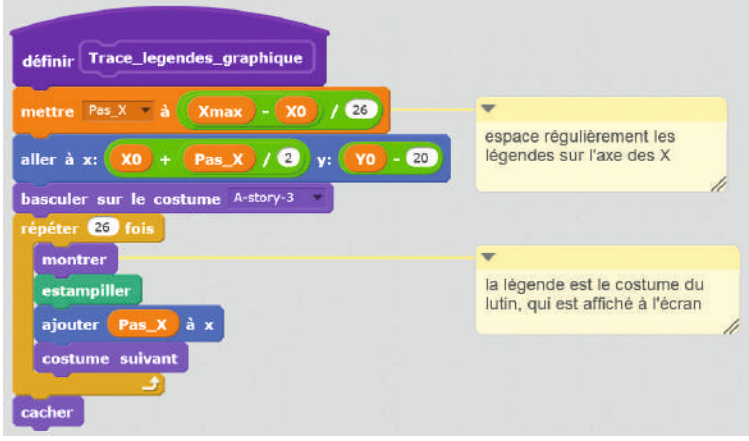
Une façon simple de placer les graduations aurait été de parcourir l'axe des X et d'écrire (un peu en dessous de cet axe) les lettres A, B, C jusqu'à Z. Malheureusement, *Scratch* ne permet pas d'écrire à l'écran : on peut soit dessiner à l'aide du stylo, soit afficher ou masquer des lutins (qui peuvent prendre différentes apparences : leurs « costumes »). Programmer le dessin de chaque lettre à l'aide du stylo est très fastidieux et ne présente aucun intérêt c'est pourquoi nous allons utiliser la seconde méthode.

Pour faire afficher une lettre à l'écran, le mieux est de créer un nouveau costume au lutin, costume qui aura l'apparence de cette lettre. On peut créer autant de costumes qu'on le souhaite, et donc on peut représenter toutes les lettres de l'alphabet. Ci-dessous, la méthode pour importer un costume prédéfini dans *Scratch*.

Importer les 26 costumes prend un peu de temps mais ne présente aucune difficulté. Une fois tous les costumes créés, il faut décider de leur emplacement. Si l'on veut espacer les lettres régulièrement, on peut calculer l'espacement entre les lettres et créer une variable (nommée « Pas_X » ci-après).



Il ne reste plus qu'à placer toutes les lettres. Pour cela, il faut déplacer le lutin, en lui faisant changer de costume et en l'estampillant à chaque étape.



Notes pédagogiques

• Ici, on s'est décalé horizontalement de « pas_X/2 » afin de centrer la position des lettres, et verticalement de « -20 » afin d'écrire en dessous de l'axe des abscisses.

• Il est aussi possible d'utiliser des « clones » du lutin, à la place de l'instruction « estampiller ». Les clones sont indispensables si l'on veut que les différentes instances du lutin puissent évoluer (se déplacer, interagir). Ici, il s'agit simplement de fixer une image à l'écran : l'instruction « estampiller » suffit largement.

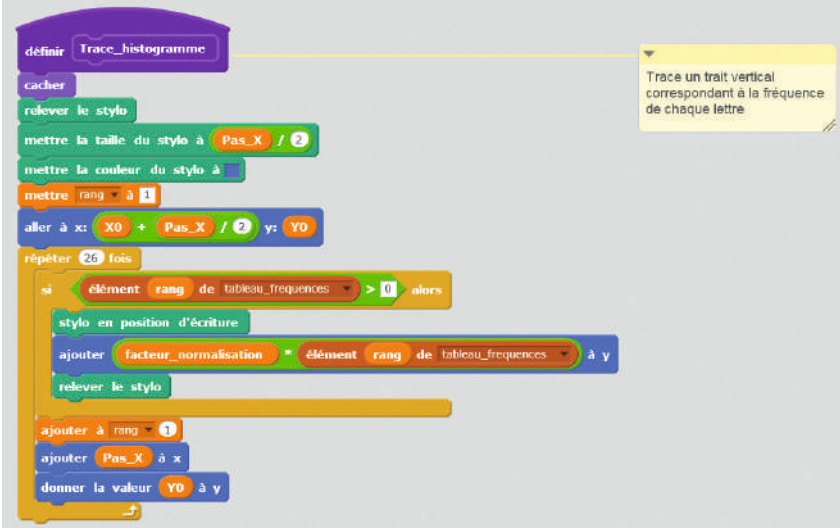
Le résultat final est un graphique qui ressemble à ceci :

Séance 14 – (optionnelle) Programmer l'affichage de l'histogramme des fréquences (2/2)

Discipline dominante	Mathématiques
Résumé	Les élèves terminent leur programme en lui faisant tracer le graphique correspondant à l'histogramme des fréquences.
Notions *(cf. scénario conceptuel, page 117)*	Idem séances précédentes
Matériel	Idem séances précédentes

Étape 4 : tracer l'histogramme (20 minutes)

Le tracé de l'histogramme reprend les notions déjà abordées au cours des étapes précédentes, le travail ne pose donc pas de grande difficulté.



Notes pédagogiques

• Si les élèves utilisent un texte court, la fréquence de chaque lettre est un tout petit nombre (entre zéro et quelques unités). Si l'on utilise directement ce nombre comme ordonnée, on obtiendra un trait de quelques pixels de hauteur seulement. Pour éviter cela, on utilise un coefficient multiplicateur. Si ce coefficient est fixé à 100, une lettre apparaissant 2 fois dans le texte sera représentée graphiquement par un trait de 1 x 100 = 200 pixels de hauteur.

• Dans un premier temps, on peut fixer arbitrairement la valeur de ce coefficient multiplicateur. Dans un second temps, il est préférable de le calculer, de façon à « normaliser le graphique » (voir l'étape ci-après). C'est pour cette raison qu'on a nommé la variable « facteur de normalisation »

Étape 5 : normaliser l'histogramme (20 minutes)

L'algorithme permettant de normaliser le graphique n'est pas trivial. Il est peu probable que des élèves de cycle 4 le trouvent tout seuls. Rien n'interdit en revanche de l'élaborer collectivement ou, autre façon de faire, de montrer le programme aux élèves et chercher à comprendre son fonctionnement. On peut « verbaliser » cet algorithme de la façon suivante. Pour normaliser le graphique, il faut :

• Trouver la lettre la plus fréquente

– Pour cela, on décide que la lettre la plus fréquente est la première (lettre A), puis on parcourt le tableau des fréquences : on compare la fréquence de la lettre la plus fréquente à celle de la lettre suivante. Si cette dernière est supérieure, alors cet élément suivant devient lui-même l'élément le plus fréquent

• Une fois que l'on connaît la lettre la plus fréquente, on s'arrange pour que l'ordonnée de la fréquence de cette lettre corresponde à l'ordonnée maximale qu'on a définie (Ymax).

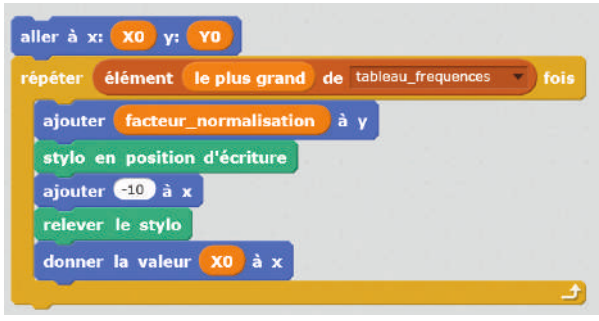
Le résultat est :

Étape 6 : graduer l'axe des ordonnées (10 minutes)

Le professeur explique que, maintenant que le graphique est normalisé, il devient facile de le graduer : on sait quelle est l'ordonnée maximale (Ymax) et à quelle valeur elle correspond (la fréquence de la lettre la plus fréquente).

Le facteur de normalisation (voir étape précédente) correspond justement à l'écart, sur l'axe des ordonnées, entre 2 fréquences unités.

Tracer des graduations sur l'axe Y se fait donc de la façon suivante :

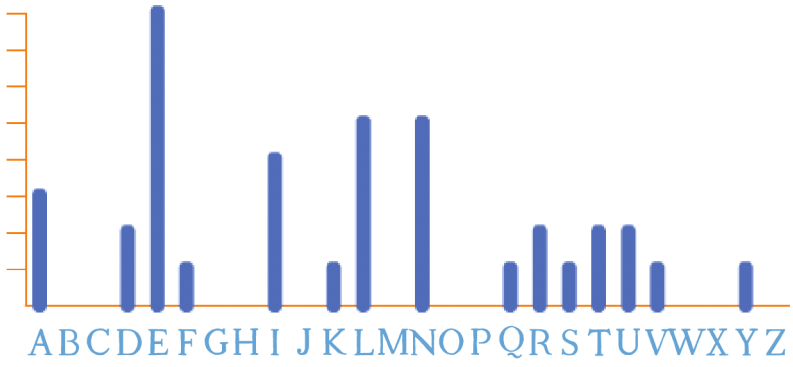


Le plus logique est de placer ces instructions à la fin de la fonction « Trace_legendes_graphique » créée précédemment.

Étape 7 : analyser quelques textes (10 minutes)

Les élèves vérifient le bon fonctionnement du programme en le testant sur des textes courts, sur lesquels ils peuvent calculer rapidement les fréquences à la main.

Le texte suivant : « Al-kindi, inventeur de l'analyse fréquentielle » (on peut laisser les accents, puisque notre programme les gère), entré dans notre programme, donne cet histogramme :



Prolongement

Les élèves peuvent désormais cryptanalyser un texte long, en reprenant la méthode utilisée à la séance 4 page 136 et en utilisant leur programme *Scratch* pour générer l'histogramme de fréquences.

Cette ressource est issue du projet thématique *1,2,3... CODEZ !*, paru aux Éditions Le Pommier.

Retrouvez l'intégralité de ce projet sur : https://www.fondation-lamap.org/projets-thematiques.

Fondation *La main à la pâte*
43 rue de Rennes
75006 Paris
01 85 08 71 79
contact@fondation-lamap.org

Site : www.fondation-lamap.org

FONDATION La main à la pâte
POUR L'ÉDUCATION À LA SCIENCE