Thème : arithmétique

Crible d'Eratosthène

TI-83 Premium CE

python

Résumé : il s'agira d'utiliser la méthode bien connue dite du « crible d'Eratosthène » pour déterminer les nombres premiers inférieurs ou égaux à 100.

Mots-clés : nombre premier ; liste ; reste division euclidienne ; crible d'Eratosthène

Compétences visées

Chercher : « observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » ; le but ici est de démarrer le travail de manière autonome et de poursuivre par l'outil informatique.

Représenter : « changer de registre » en passant d'un travail « à la main » à un travail informatique.

Calculer : « mettre en œuvre des algorithmes simples », en utilisant et faisant évoluer un programme.

Situation déclenchante

Il est important de connaître les nombres premiers pour diverses raisons, comme la simplification de fractions.

Ces nombres jouent un rôle important en arithmétique et une connaissance d'une liste des « premiers » nombres premiers est intéressante : une méthode pour l'établir porte le nom de son auteur : le crible d'Eratosthène. Cela permet également de prendre conscience que cette exploration a alimenté la recherche depuis déjà de nombreux siècles !

Image libre de droits d'après Pixabay

Problématique

Comment déterminer efficacement la liste des nombres premiers inférieurs ou égaux à 100 ?

Thème : arithmétique

Crible d'Eratosthène

TI-83 Premium CE

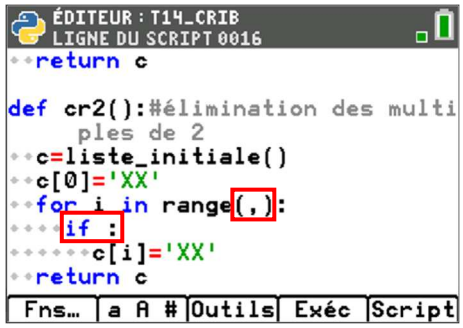
python™

Scénario pédagogique

- **Avec la classe :** rappeler ce que sont des nombres premiers et demander d'établir une liste de ces nombres.
- **Mise en commun** (éventuellement au sein de petits groupes) : quelle(s) méthode(s) ont été mises en place ? Si besoin, présentation de la méthode dite du « crible d'Eratosthène ».
- **Code :** une partie du code est donnée, en particulier celle qui permet l'affichage. La fonction permettant de créer la liste des nombres de 1 à 100 est l'occasion d'un travail sur les listes.

La suite du code peut être donnée tout ou partie aux élèves avec un travail de complétion si besoin.

Dans le cas où l'enseignant souhaite donner le script à compléter par l'élève en supprimant certaines parties à l'exemple du contenu du `range` et de la condition du `if` comme ci-contre encadré en rouge.



Si l'on procède de la sorte, le script en lui-même ne fonctionnera pas, un message d'erreur apparaîtra. Or, il peut être intéressant d'utiliser d'autres fonctions du script, même si une fonction est à compléter dans celui-ci.

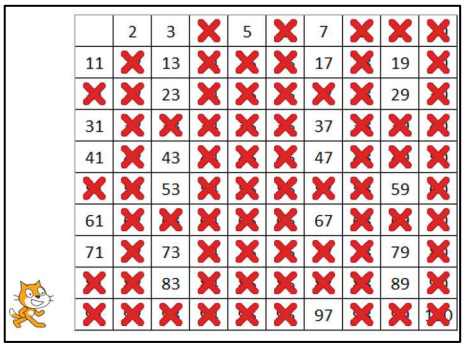
Il est possible que l'exécution du script ne tienne pas compte d'une ligne en la commentant par # qui s'obtient, par exemple, par la touche `[2nde]` suivie de la touche `[(-)]`.

Il est beaucoup plus efficace d'utiliser les triples guillemets, en plaçant entre ces guillemets la partie à commenter, comme montré ci-contre. Le texte prend alors la couleur verte des chaines de caractères (`strings`). Au départ, ces triples guillemets servent à documenter une fonction : le texte placé entre eux est appelé `docstrings`. L'usage en est ici détourné pour proposer une programmation fonctionnelle avec des parties plus tard à compléter.

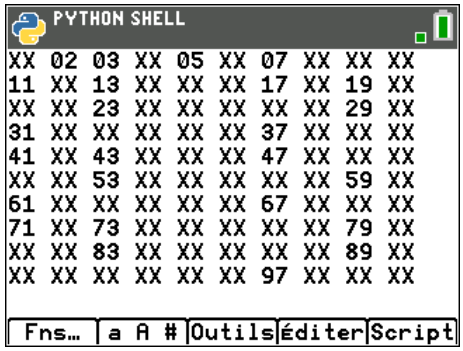
- **Pour les élèves les plus en avance :** il est possible de leur proposer un ou plusieurs prolongements possibles, décrit en fin de fiche.

Voici les visuels à l'issue des programmes :

en Scratch

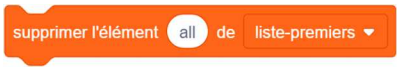
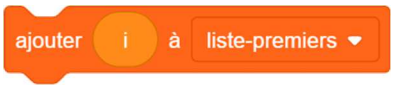


avec la TI-83 Premium CE Edition Python



Crible d'Eratosthène

Avec Scratch

Les briques de codes principales en Scratch pour ce programme	Explications Cette instruction permet de :	Traduction en langage Python sur la TI-83 Premium CE Edition Python
	Supprimer l'ensemble des éléments de la liste « liste-premiers » ; elle devient vide.	`c=[]` Cette instruction permet de créer une liste vide nommée « c ».
	Ajouter la valeur de la variable « i » à la suite des autres éléments de la liste « liste-premiers ».	c.append(i) *append* signifie en anglais « ajouter ».
	Choisir le i-ième de la liste « liste-premiers ».	`c[i]`
	De créer un tampon du lutin choisi, la croix ici.	Cette fonction n'a pas d'utilité dans la programmation Python puisque celle-ci n'est pas dans un mode graphique.

A noter que dans la dernière version de Scratch, il faut chercher ce qui concerne le stylo dans les extensions : 

Une programmation possible est disponible sur le site de Scratch : scratch.mit.edu/studios/27615196/

Thème : arithmétique

Crible d'Eratosthène

TI-83 Premium CE

python™

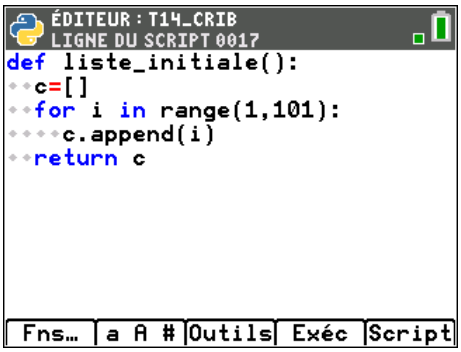
Avec Python

Des parties de code peuvent être rédigées au fur et à mesure avec les élèves, d'autres leur pourront être directement fournies et commentées. Ce code est composé de plusieurs fonctions :

- La fonction `liste_initiale`, fonction sans paramètre, permet de générer la liste des nombres de 1 à 100.

  Cela permet un travail sur les listes avec notamment la méthode `.append` qui permet d'ajouter un élément à la fin d'une liste.

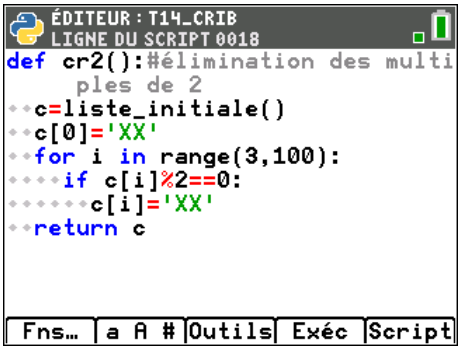
  Bien noter que l'instruction `range(1,101)` permet d'aller de 1 (inclus) à 101 (exclu), donc jusqu'à 100.



- La fonction `cr2()`, fonction sans paramètre, qui permet de remplacer tous les multiples de 2 qui lui sont supérieurs par la chaine de caractère `'XX'`.

  Sur le plan mathématique et syntaxique, il est important d'insister sur la ligne `c[i]%2==0`. L'opérateur `%` permet de récupérer le reste d'une division euclidienne. On pourra, par exemple, transférer un code aux élèves sans cette ligne et la rédiger ensemble.

  Bien comprendre pourquoi il est écrit : `for i in range(3,100) :` la dernière valeur prise pour `i` est bien `99` et `c[99]` est la dernière valeur de la liste, soit initialement 100.



- La fonction `cr3()`, fonction sans paramètre, qui permet de remplacer dans la liste précédemment créée tous les multiples de 3. Il est envisageable de laisser les élèves rédiger totalement cette fonction.

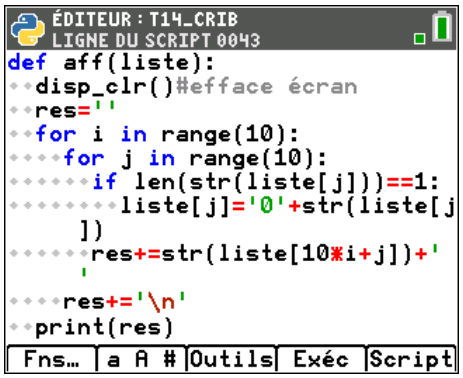
  Il est efficace d'utiliser les copier/coller : on se place sur la ligne à copier puis `Outils` par appui sur la touche [`zoom`] et enfin [5] ; on se place ensuite sur la ligne où l'on veut effectuer cette copie puis `Outils` et [7].

- La fonction `aff` : elle a pour paramètre une liste et a pour but d'effectuer un affichage « propre » de la liste de nombres, avec éventuellement des `XX`.

  Elle est à fournir aux élèves sans forcément rentrer dans les détails.

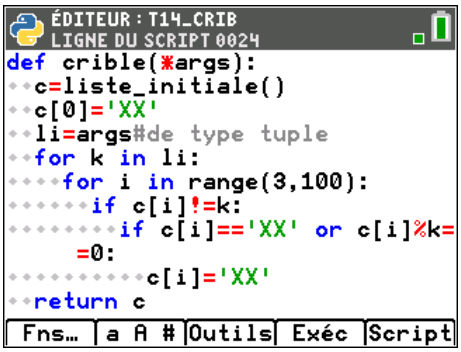
  La syntaxe `'\n'` permet un retour à la ligne.

  A noter que pour un résultat plus propre, on a effectué un « efface écran », avec `disp_clr`, qui a nécessité l'importation en préambule de la bibliothèque `ti_system`. Effacer l'écran n'est évidemment pas obligatoire.



Crible d'Eratosthène

- La fonction `crible` est réservée à l'enseignant : elle prend en paramètres les valeurs choisies par l'utilisateur. Elle permet de remplacer par des `XX` tous les multiples supérieurs aux valeurs inscrites. Cette programmation est expliquée dans la partie « Pour mieux lire le code Python ».



Une programmation possible est disponible sur le site TI : education.ti.com/fr/scratch-python

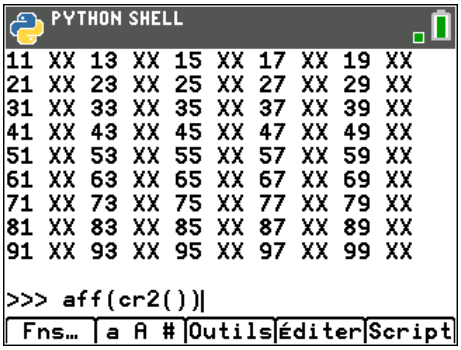
Mode opératoire

Une fois le script exécuté, il faut appuyer sur la touche [`var`] : les fonctions définies dans le script apparaissent.

Par les flèches directionnelles, il faut sélectionner la fonction `aff` puis la fonction `cr2` ; cette « composition » de fonctions permet d'afficher les nombres de 1 à 100 en éliminant 1 et les multiples de 2 qui lui sont supérieurs.

Si on veut « remonter » dans l'affichage des nombres, on appuie sur la touche [`2nde`] suivie de la flèche directionnelle [`haut`].

Si on veut rappeler une fonction précédente, on utilise la flèche directionnelle [`haut`] ; il est possible de le faire plusieurs fois de suite pour rappeler des commandes déjà effectuées précédemment.



Prolongements possibles

Voici des pistes pour les élèves les plus rapides ou qui ont envie de prolonger le travail :

- établir le crible d'Eratosthène non plus de 1 à 100 mais de 1 à 500 ;
- se demander à partir de quel nombre premier testé, il est sûr d'avoir éliminé tous les nombres non premiers entre 1 et 100, et prolonger de 1 à 10 000 ;
- se demander l'effet de : `crible(2,3)` par rapport à `crible(6)` ? de `crible(2,4)` par rapport à `crible(8)` ?

  La fonction `crible` permettra à l'enseignant et/ou aux élèves d'ouvrir des questionnements intéressants sur le plan arithmétique.

Crible d'Eratosthène

Pour mieux lire le code python

Dans la fonction `crible`, l'écriture `*args` en paramètre a été utilisée. Cela permet d'écrire le nombre de paramètres que l'on souhaite.

`args` est de type `tuple`. Un `tuple` est une collection ordonnée de plusieurs éléments. Cela ressemble à une liste, mais on ne peut pas le modifier une fois créé. Ainsi, `t = (1,8,12)` est un `tuple`, un triplet en langage mathématique. `t[0]` est égal à `1`.

Ainsi, `args` est un `tuple` qui contient tous les paramètres qui ont été saisis par l'utilisateur, séparés par des virgules. Cette programmation est hors programme mais peut permettre à certains élèves d'aller plus loin ou de débloquer certains scripts.

La fonction `test` ci-contre permet de comprendre comment cela fonctionne ; elle renvoie la longueur du `tuple`, c'est-à-dire le nombre de paramètres saisis.

De même, la fonction `somme` proposée permet de faire la somme des valeurs saisies comme paramètres, autant que souhaité.

`somme(1,2,3)` saisi dans le `shell` donnera `6` et il est possible de saisir le nombre de paramètres que l'on veut.