Sommaire

Avant-propos

Ce livret a pour double objectif d'aider à l'enseignement de Python pour la classe de seconde, ainsi qu'à faire vivre la liaison troisième/seconde en permettant aux uns de travailler sur un logiciel de programmation par blocs comme Scratch et aux autres de travailler les mêmes problématiques sur la calculatrice TI-83 Premium CE Edition Python avec son module Python. Des défis peuvent alors se faire d'une classe à l'autre.

Ce livret est constitué d'une quinzaine de fiches réparties selon différents thèmes et comportant parfois une certaine progression :

- Introduction : instructions (fiche 1) ; figures géométriques (fiches 2 et 3) ; notion de hasard (fiche 4) ;
- Une figure géométrique plus complexe (fiche 5) ;
- Fonctions (fiches 6 à 9) ;
- Géométrie analytique (fiches 10 à 12) ;
- Pour aller plus loin (fiches 13 à 15) ;

Mis à part la première, les fiches ont été conçues selon plusieurs rubriques :

- Introduction avec résumé et mots-clés ; Compétences visées ; Situation déclenchante et Problématique ;
- Scénario pédagogique possible ;
- Avec Scratch ;
- Avec Python ;
- Conclusion avec Mode opératoire ; Prolongements possibles et parfois Pour mieux lire le code Python.

Les fiches sont accompagnées de programmations possibles :

- pour Scratch, directement en ligne : scratch.mit.edu/studios/27615196/ ;
- pour la calculatrice TI-83 Premium CE Edition Python, une archive contenant l'ensemble des scripts sur le site TI : education.ti.com/fr/scratch-python. Dans ce cas, il faut soit utiliser le TI-SmartView™ CE pour la famille 83, soit le logiciel TI Connect™ CE afin d'utiliser les scripts ou les transmettre aux élèves depuis un ordinateur.

L'algorithmique et la programmation se construisent depuis le début du cycle 2 (CP), d'abord comme une pensée, puis en codant sur des logiciels. Il est nécessaire de prendre du temps afin que les élèves puissent tester, se tromper, se questionner, retester pour produire une solution possible à une problématique donnée. L'engagement sur des projets longs de quelques heures permet à chaque élève de se saisir de cette problématique. Certains élèves iront plus vite et on pourra leur complexifier la tâche par l'ajout d'un compteur, de modifier des couleurs ou de chercher à produire une figure plus complexe. Résoudre un problème permet aussi de travailler l'ensemble des six compétences de l'activité

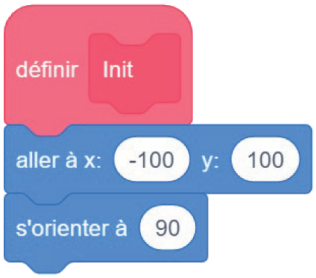
mathématique et informatique : chercher, modéliser ou simuler, représenter, raisonner, calculer et communiquer.

L'élève de fin de cycle 4 a des notions de blocs algorithmiques qu'il convient d'accompagner au lycée lors du passage au langage Python selon différentes phases essentielles que nous vous proposons :

- S'appuyer, dès le début d'année de seconde, sur la géométrie grâce au module graphique `ce_turtl`. Pour se faire, la première fiche permet d'avoir les instructions du module `turtle` afin de bien commencer.
- Montrer le parallèle entre un bloc Scratch et le script associé sur Python par l'ajout de bordures comme ci-dessous avec la boucle répéter. L'indentation ◆◆ se comprend alors comme moyen de rester dans la boucle, et pour sortir de la boucle, il faut sortir de l'indentation.


```
for i in range():
◆◆
```

- De même, utiliser la notion de bloc utilisateur dans le cas de Scratch pour faire comprendre la décomposition en plusieurs fonctions sur Python. Par exemple, ci-dessous, voici un bloc utilisateur d'initialisation de programmes Scratch et son pendant en Python.




```
def fonction(p1,p2,...):
◆◆instructions
return ...
```

Suivant les fiches, il est proposé des programmations qui sont soit déjà construites à envoyer aux élèves, soit à compléter, soit à construire entièrement par les élèves. Dans la fiche n°15, le crible d'Eratosthène, est détaillé un procédé pour exécuter un script envoyé par le professeur, qui aura, au préalable, supprimé certaines parties. Ce procédé permettra l'exécution des autres fonctions sans provoquer d'erreur.

Il faut bien penser à procéder aux mises à jour des calculatrices afin de garantir pour tous les élèves les mêmes versions et les mêmes fonctionnalités de Python.

Pour nous contacter par mail : Sylvain ETIENNE, sylvain-julien.etienne@ac-nice.fr et Florent GIROD, florent.girod@ac-grenoble.fr.

Merci à Boris, Carlos et Marthe pour leur accompagnement,

ainsi qu'à Fred from Dallas pour la fiche d'instructions originelle.

Les instructions du module turtle

Pour utiliser le module

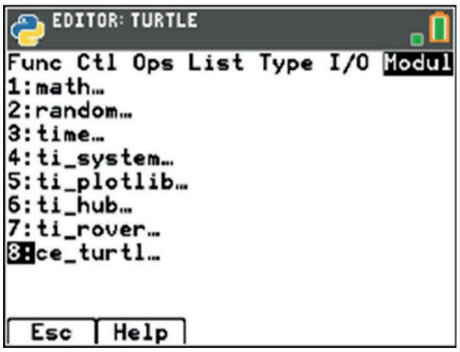
- Installer le module `ce_turtl` sur votre calculatrice TI-83 Premium CE Edition Python.

  Importer le module dans un script en écrivant l'instruction ci-contre en haut du programme.

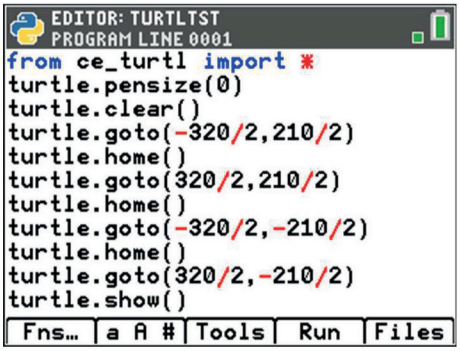
  Pour ce faire : [2nde] [catalog] et choisir `from SCRIPT import *`

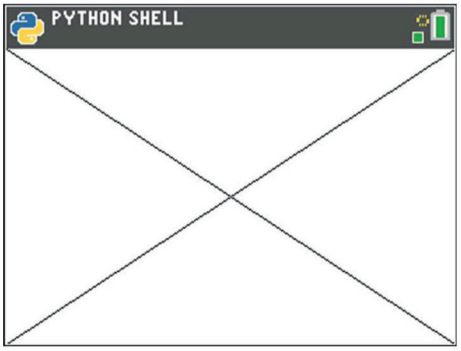
  Taper lettre à lettre `ce_turtl` ; le raccourci touche pour l'underscore _ est [2nde] suivie de [(-)].

  Sélectionner l'onglet `Fns…` par appui sur la touche [f(x)] puis `Modul` ; on remarque que `8:ce_turtl` est ajouté en fin de liste des bibliothèques proposées.



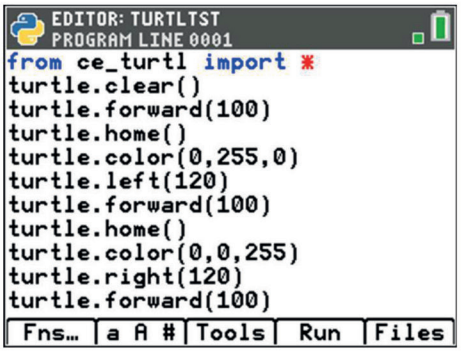
- L'écran du module turtle est centrée aux coordonnées $(0\,;0)$. La largeur de l'écran est 320 pixels, et sa hauteur 240 pixels. Cependant, la barre de statut en haut de l'écran a une hauteur de 30 pixels et il n'est pas possible d'écrire à cet emplacement. Ainsi, la partie utile de l'écran a une hauteur de 210 pixels.

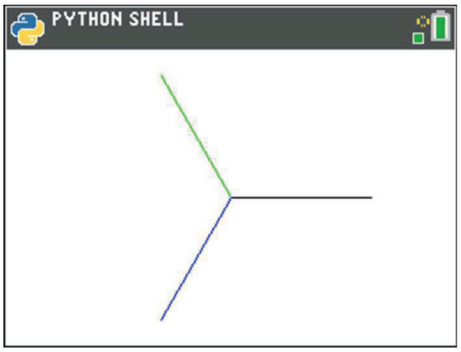




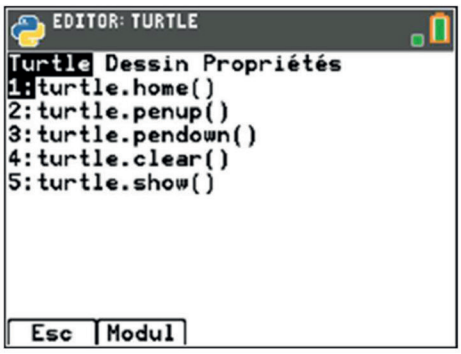
Les instructions du module turtle

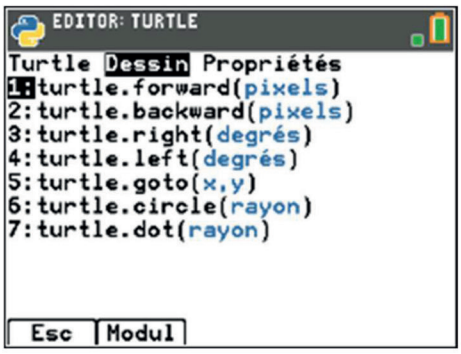
- Les angles sont exprimés en degrés. « Tourner à gauche » signifie tourner dans le sens inverse des aiguilles d'une montre. La position « home » correspond à un crayon en position d'écriture avec un angle de zéro degré, orienté vers l'Est donc. Le programme suivant permet d'illustrer ce descriptif.

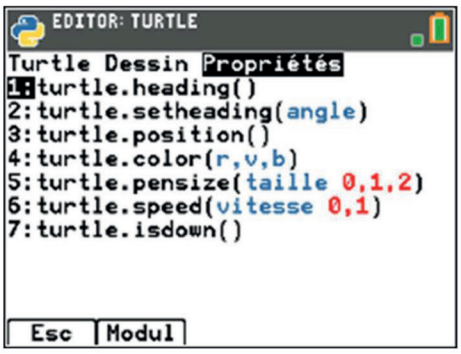




- Sélectionner `ce_turtl` pour voir les diverses méthodes de la bibliothèque. Choisir la méthode voulue et la coller dans le script.







Aide sur les diverses fonctions

Dans l'onglet « `Turtle` » :

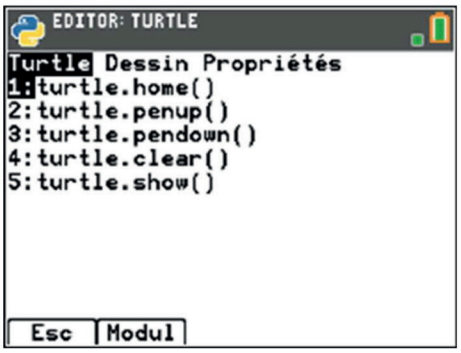
`turtle.home()` positionne la tortue en position $(0\,;0)$, au centre de l'écran, avec une orientation de 0° vers l'Est.

`turtle.penup()` place la tortue en mode de non-écriture.

`turtle.pendown()` place la tortue en mode d'écriture.

`turtle.clear()` efface l'écran.

`turtle.show()` montre la figure construite par la tortue jusqu'à ce qu'on appuie sur la touche [on] ou sur [annul].



Les instructions du module turtle

Dans l'onglet « Dessin » :

`turtle.forward(pixels)` : la tortue avance du nombre de pixels dans la direction de la tête de la tortue.

`turtle.backward(pixels)` : même chose mais dans le sens opposé.

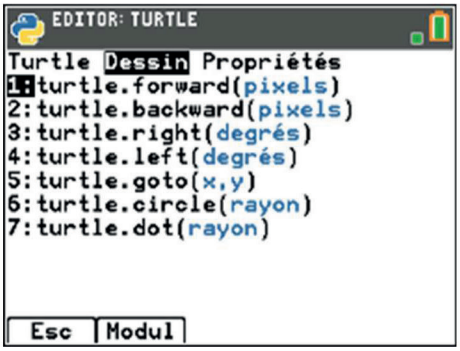
`turtle.right(angle)` : change l'orientation de la tête de la tortue en tournant d'un angle (en degrés) vers la droite par rapport à l'orientation courante.

`turtle.left(angle)` : même chose mais dans le sens opposé.

`turtle.goto(x,y)` : fait passer la tortue de sa position courante vers le point de coordonnées $(x\,;y)$.

`turtle.circle(rayon)` : dessine un cercle centré à la position courante de la tortue avec la valeur du rayon donnée en paramètre.

`turtle.dot(rayon)` : dessine un disque centré à la position courante de la tortue avec la valeur du rayon donnée en paramètre.



Dans l'onglet « Propriétés » :

`var = turtle.heading()` : renvoie la valeur de l'angle de la tête de la tortue.

`turtle.setheading(angle)` : modifie l'angle de la tête de la tortue en la valeur mise en paramètre.

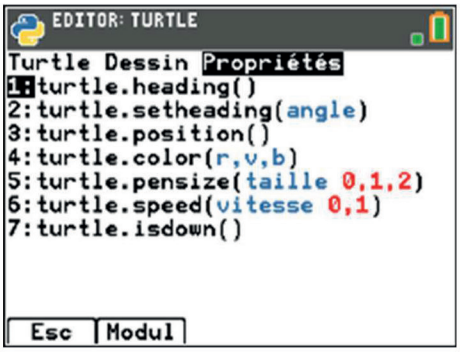
`x,y = turtle.position()` : renvoie la position courante de la tortue en donnant ses coordonnées, dans le repère décrit précédemment.

`turtle.color(r,v,b)` : règle la couleur du dessin de la tortue.

`turtle.pensize(taille)` : règle l'épaisseur du trait de la tortue.

`turtle.speed(vitesse)` : règle la vitesse de déplacement de la tortue sur l'écran.

`var = turtle.isdown()` : renvoie `True` si la tortue est en position d'écriture et `False` dans le cas contraire.



Des carrés

Résumé : il s'agit d'un travail de première approche de la programmation en langage Python permettant d'appréhender les notions de boucle itérative et de fonction.

Mots-clés : bibliothèque `turtle` ; boucle itérative ; fonction

Compétences visées

Chercher : « observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » avec ici un résultat qui s'observe directement par la construction de la figure.

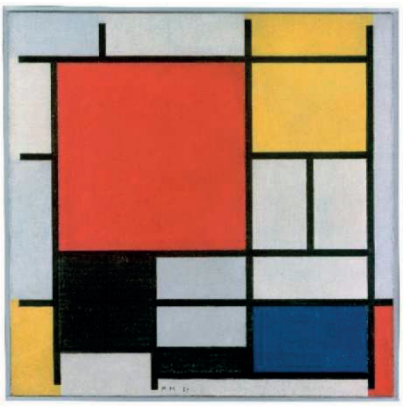
Représenter : « changer de registre » en passant d'une représentation graphique à un langage informatique.

Raisonner : « mettre en œuvre des algorithmes simples », en utilisant ici la répétition.

Situation déclenchante

De nombreux artistes utilisent des figures géométriques simples pour composer leurs œuvres. Il est possible d'obtenir des figures intéressantes à partir de rectangles par exemple.

Cette fiche propose la construction de figures simples à partir de carrés. Pour cela, il est nécessaire d'amener un raisonnement algorithmique permettant la répétition par le biais d'une boucle itérative et il faudra être en mesure de construire des carrés dont les côtés auront des longueurs différentes : les fonctions répondent à ce problème.

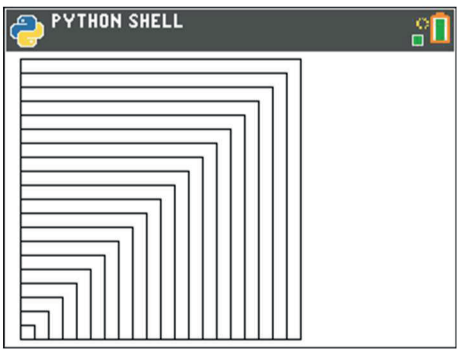


Piet Mondrian, 1921 – Composition en Rouge, jaune, bleu et noir

Image d'après Wikipédia

Problématique

Comment construire de manière efficace des figures géométriques composées de carrés (comme celle proposée ci-contre) ?



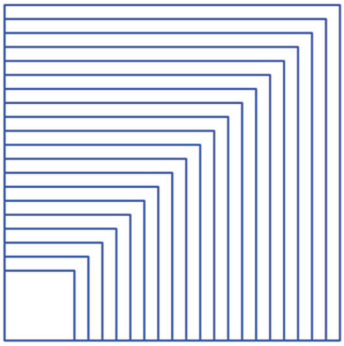
Des carrés

Scénario pédagogique

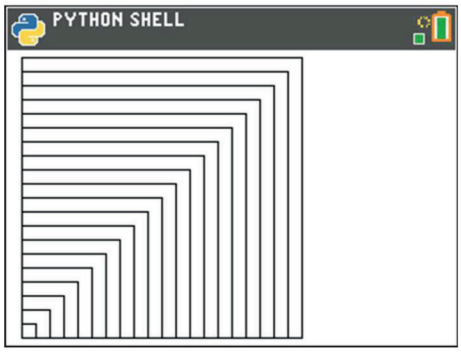
- **Avec la classe :** demander à chaque élève de réfléchir à une manière de construire un carré sur le plan algorithmique.
- **Mise en commun :** mettre en évidence l'efficacité d'une itération et demander à l'ensemble des élèves de coder la construction d'un carré.
- **Plusieurs carrés :** souligner le besoin de demander la valeur de la longueur du côté à l'utilisateur pour construire plusieurs carrés et l'intérêt d'utiliser ce qui a été fait précédemment pour construire un une figure contenant plusieurs carrés.
- **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



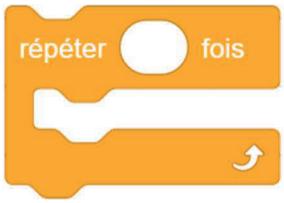
avec la TI-83 Premium CE Edition Python

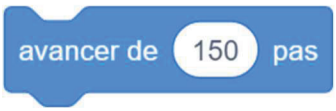
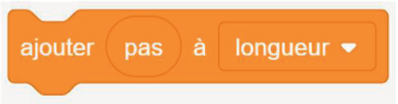


Des carrés

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

	Créer une boucle répéter.	`for i in range(n):`
		`••instructions`
		Les valeurs de i commencent à 0 et augmentent avec un pas de 1.
		L'indentation avec •• est nécessaire pour définir les instructions à exécuter dans un bloc du code Python.

	Avancer du nombre de pas indiqué, ici 150 pas ou pixels, dans une direction à définir en amont (vers la droite par défaut).	`turtle.forward(150)` *Forward* signifie « vers l'avant » en anglais.
	Changer la direction du lutin en la tournant vers la gauche de l'angle indiqué en degré, ici 90°.	`turtle.left(90)` *Left* signifie « à gauche » en anglais.
	Définir un bloc utilisateur nommé carre qui comporte ici le paramètre a.	`def carre(a):` `••instructions`
	Utiliser le bloc utilisateur nommé carre en lui injectant un paramètre issu de la variable longueur ici.	
	Définir la variable longueur par incrément du pas que l'utilisateur aura donné.	`longueur+=pas` Cette instruction comporte un opérateur d'affectation augmenté. Il est aussi possible d'écrire `longueur=longueur+pas`.

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/

Des carrés

Avec Python

Cette fiche peut être vue comme une introduction au langage Python ; la syntaxe sera découverte et expliquée au fur et à mesure de la rédaction du code par le professeur (avec projection au tableau par exemple) ; voire même, le script pourra être diffusé et expliqué aux élèves.

Le code ci-contre, `T01_FOR.py`, est tout à fait compréhensible par analogie avec Scratch ; il est important de bien expliquer la syntaxe de l'itération `for i in range(4) :` vue ici comme une simple répétition, 4 fois.

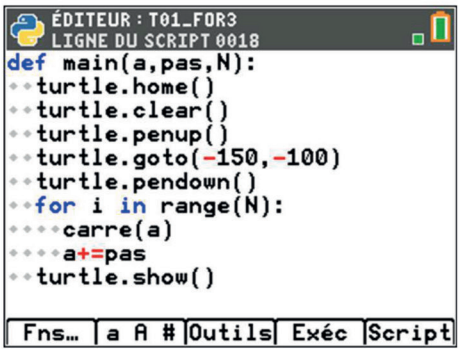
La notion d'indentation sera explicitée et le parallèle avec Scratch largement mis en avant ; le symbole « `:` » peut être vu comme l'ouverture d'une accolade dans Scratch, l'indentation ◆◆ comprenant tout ce qui est contenu dans l'accolade et le retour sans indentation comme la fin de l'accolade.

Pour la suite, le fait d'avoir à construire plusieurs carrés permet d'introduire la notion de fonction. Dans le code ci-contre, `T01_FOR2.py`, il est adapté de mettre la longueur du côté du carré comme paramètre pour être efficace dans la construction d'un carré dont la longueur du côté est choisie par l'utilisateur.

Ainsi, pour obtenir la figure donnée au paragraphe « situation déclenchante », l'appel à la fonction `carre` est très efficace.

Il est important de préciser la façon dont la longueur du côté du carré est augmentée par la syntaxe `a+=pas` qui peut être pertinente de présenter dès à présent.

L'exemple de la fonction `main` est aussi intéressant car elle comporte plusieurs paramètres. Il faudra bien préciser que ceux-ci doivent être saisis dans le bon ordre à l'appel de cette fonction.



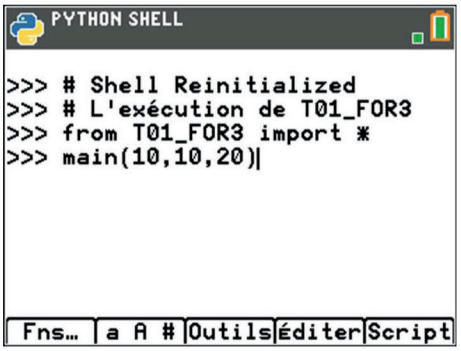
Une programmation possible est disponible sur le site TI : education.ti.com/fr/scratch-python

Des carrés

Mode opératoire

Ceci concerne le code `T01_FOR3.py` et plus généralement tous les codes comprenant des fonctions Python.

- Une fois le script exécuté, il faut appuyer sur la touche `[var]` : la ou les fonctions définies dans le script apparaissent.
- Par les flèches directionnelles, il faut sélectionner la fonction choisie (`main` ici), valider par `Ok`, puis ajouter la valeur des paramètres en respectant l'ordre : la longueur du premier carré à tracer, celle du pas choisi et enfin celle du nombre de carrés à dessiner.

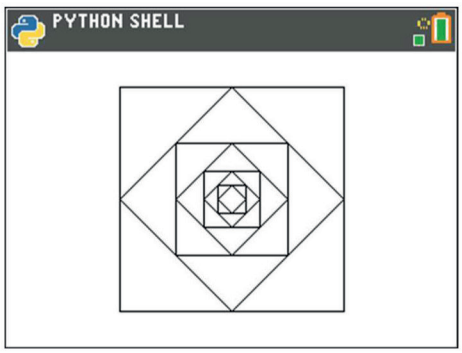


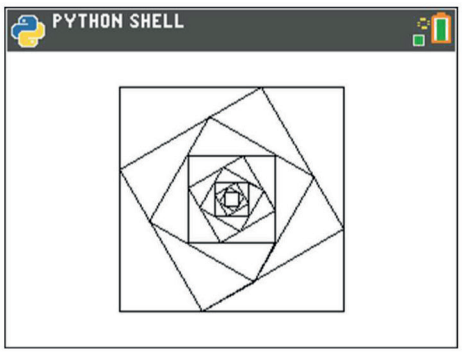
Prolongements possibles

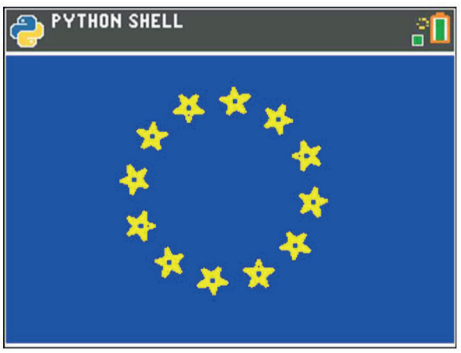
Voici des pistes pour les élèves les plus rapides ou qui ont envie de prolonger le travail :

- programmer d'autres constructions de figures basées sur des carrés ;
- programmer la construction d'un polygone régulier à n côtés ;
- programmer des constructions basées sur des triangles équilatéraux ;
- laisser libre cours à son imagination…

Voici quelques propositions de figures :







Des spirales

Résumé : cette activité propose de construire une spirale en se basant sur des carrés ; la démarche s'appuie sur une boucle itérative puis sur une boucle « tant que » avec comme critère d'arrêt la longueur de la spirale.

Mots-clés : bibliothèque `turtle` ; boucle itérative ; tant que ; boucle `while` ; spirale ; somme

Compétences visées

Chercher : « observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » avec ici un résultat qui s'observe directement par la construction de la figure.

Représenter : « changer de registre » en passant d'une représentation graphique à un langage informatique.

Raisonner : « mettre en œuvre des algorithmes simples », en utilisant ici une boucle « tant que ».

Situation déclenchante

Il existe de nombreuses figures intéressantes esthétiquement (et mathématiquement !) qui appartiennent à la famille des « spirales », que l'on retrouve aussi dans la nature.

Comment construire une spirale simple, à quel moment stopper sa construction ? En route vers un chemin infini, ou pas !

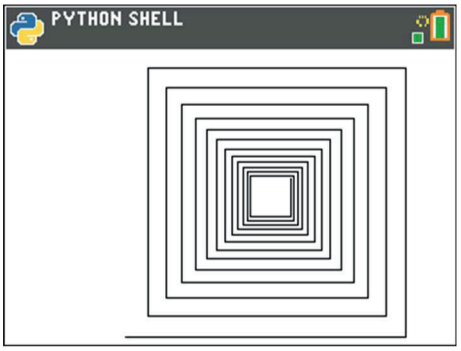
Image libre de droits d'après Pixabay

Problématique

Proposer un code permettant de construire la figure ci-contre : une spirale composée de segments perpendiculaires successivement.

Le premier segment a pour longueur 200 pixels ; les longueurs diminuent de 4 % à chaque fois. Cette spirale compte 50 segments.

On souhaite ensuite avoir comme critère d'arrêt de la construction la donnée de la longueur totale de la spirale ; est-il possible de construire une telle spirale de longueur 4 000 pixels ? 6 000 pixels ?



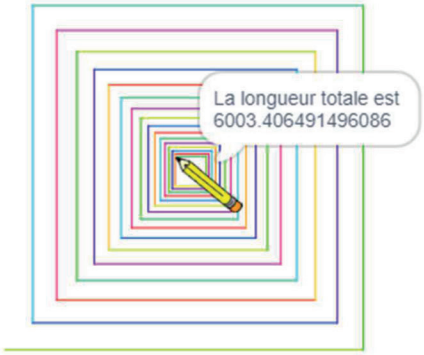
Des spirales

Scénario pédagogique

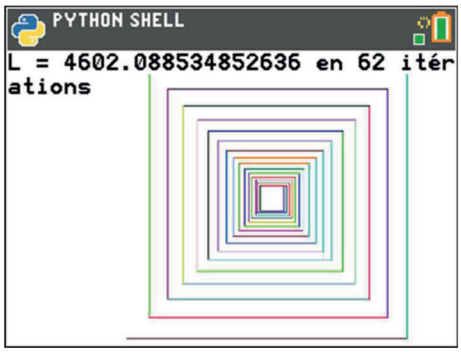
- **Avec la classe :** effectuer une recherche individuelle pour bien assimiler le problème avec une construction « à la main » et une ébauche d'algorithme.
- **Mise en commun** (éventuellement au sein de petits groupes) : écrire d'un premier script utilisant une boucle itérative dont le nombre de répétitions est prise en paramètre de la fonction.
- **Nouvelle recherche :** réfléchir à la manière de procéder pour prendre en compte la longueur totale de la spirale comme critère d'arrêt de la construction.
- **Mise en commun :** utiliser une boucle « tant que » sur Python ou « répéter jusqu'à ce que » sur Scratch et expliciter une méthode pour obtenir la longueur totale de la spirale.
- **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

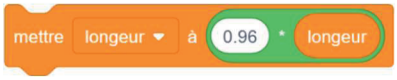


avec la TI-83 Premium CE Edition Python

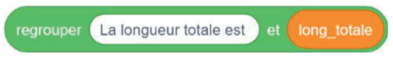
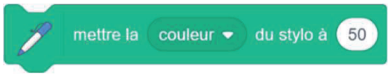


Des spirales

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
	Affecter la valeur de la longueur précédente diminuée de 4 %.	`a=a*0.96`

	Répéter les instructions contenues dans la boucle jusqu'à ce que la condition se réalise.
	Il n'y a pas de boucle « répéter tant que » dans Scratch, et pas de boucle « répéter jusqu'à ce que » dans le langage Python.
	Attention donc à l'adaptation en langage Python.
		`while condition inverse :`
		`••instructions`
		L'indentation avec •• est nécessaire pour définir les instructions à exécuter d'une structure dans le code Python. *While* en anglais signfie « tant que » en anglais

	Comparer deux valeurs et renvoyer un booléen.	`L<L_tot`
	Lier du texte et des valeurs de variables.	`print("L=",l_tot,"en",cpt, "itérations")`
	Modifier la couleur du stylo. Pour rendre aléatoire, il faut rajouter le bloc dédié. Les valeurs possibles vont de 0 à 199.	`turtle.color(randint(0,255), randint(0,255),randint(0,255))` L'instruction `color` demande trois paramètres : rouge, vert, bleu. L'instruction `randint(a,b)` renvoie un nombre entier entre `a` et `b` inclus.

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/

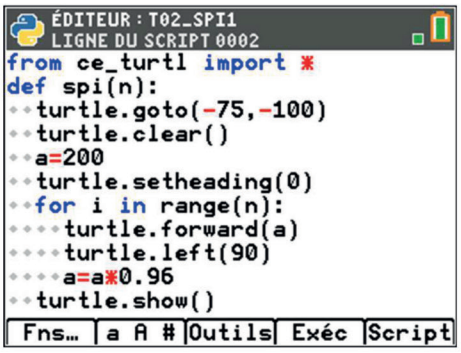
Des spirales

Avec Python

Avec le travail effectué sur les boucles itératives (fiche n°2, des carrés), les élèves sont en mesure de réaliser en partie ce code.

L'enseignant éclairera les élèves sur les points suivants :

- proposer les coordonnées `(-75,100)` comme point de départ de la spirale ;
- indiquer le rôle de `turtle.setheading(0)` : cela permet de repositionner la tête de la tortue vers l'Est lors d'une nouvelle exécution de la fonction `spi` ;
- bien lire « `a` prend la valeur `a*0.96` » pour `a=a*0.96` et faire prendre conscience de la signification du symbole « = » en programmation Python.



La volonté de stopper l'arrêt de la construction de la spirale amène à utiliser une boucle `tant que`. L'enseignant pourra coconstruire le code avec les élèves.

Il peut être judicieux de dupliquer le script précédent : aller dans `Script`, puis sélectionner le script à dupliquer, puis sur `Gérer` et `Dupliquer le script`. Ainsi, il suffira de ne modifier que quelques lignes.

Bien insister sur la ligne `l_tot+=a` qui peut aussi avoir pour syntaxe `l_tot = l_tot + a` et expliciter à nouveau la signification du symbole « = » en Python.

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

Des spirales

Prolongements possibles

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

- en recherche : toutes les longueurs sont-elles « atteignables » ? En effectuant plusieurs essais, les élèves constateront que l'on ne peut pas dépasser une longueur totale de 5 000 pixels. On peut les inciter à afficher la longueur (`print(l_tot)` mis en commentaire dans le script) ; cela permet d'une part de savoir quand la construction s'arrête, et d'autre part de connaître une valeur approchée de la longueur totale.
- Il peut être intéressant d'afficher un compteur indiquant le nombre d'itérations de la boucle pour atteindre une longueur demandée. Le principe d'un compteur est fréquent en programmation.
- Construire une spirale avec des couleurs qui diffèrent pour chaque segment ; on peut à cet effet demander de faire une recherche rapide sur la génération de couleurs en codage RGB.

Voici le code correspondant aux deux premières remarques et une proposition de spirale (plus complexe !) :

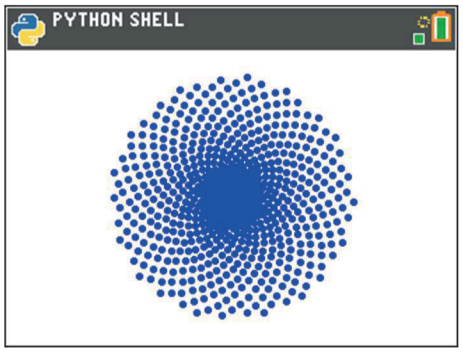
ÉDITEUR : T02_SPI3
LIGNE DU SCRIPT 0001

```
from ce_turtl import *
from random import *

def spi(L):
  turtle.goto(-75,-100)
  turtle.clear()
  a=200
  l_tot=0
  cpt=0
  turtle.setheading(0)
  while l_tot<=L:
    turtle.color(randint(0,255),
     randint(0,255),randint(0,25
     5))
    turtle.forward(a)
    turtle.left(90)
    l_tot+=a
    cpt+=1
    a=a*0.96
  print("L =",l_tot,"en",cpt,"it
     érations")
  turtle.show()
```

Fns… | a A # | Outils | Exéc | Script

PYTHON SHELL



Thème : statistiques

Ce dé est-il truqué ?

TI-83 Premium CE

python™

Résumé : cette activité propose une simulation classique de lancers de dé à six faces (dé équilibré ou non).

Mots-clés : bibliothèque `random` ; conditionnelle `if` ; simulation ; statistique ; loi des grands nombres

Compétences visées

Chercher : « observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » avec ici des résultats numériques à considérer avec un regard critique.

Modéliser : « utiliser, comprendre, élaborer une simulation numérique ».

Calculer : « mettre en œuvre des algorithmes simples », en utilisant ici une structure conditionnelle.

Situation déclenchante

Si un dé à six faces est donné, comment savoir s'il est bien équilibré ou non ? L'observation de lancers donnera des indications, pas tout à fait des certitudes !

Et avec combien de lancers ?

Image libre de droits d'après Pixabay

Problématique

Comment simuler un lancer de dé à six faces :

- S'il est bien équilibré ?
- S'il est pipé ?

Au passage, il est intéressant de se questionner sur la simulation numérique : remplace-t-elle réellement un dé ?

Ce dé est-il truqué ?

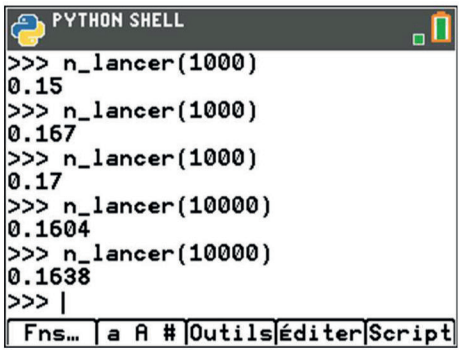
Scénario pédagogique

- **Avec la classe :** lancer une recherche par binôme ou trinôme en distribuant des dés à six faces, avec dans l'idéal, certains de ces dés pipés.
- **Tests :** se mettre d'accord sur un protocole pour déterminer si un dé est bien équilibré ou pas. Après plusieurs essais, basculer vers une simulation numérique nécessitant la création d'un code.
- **Utilisation d'un code :** rédaction de ce code avec la classe avec mise en évidence de l'instruction conditionnelle.
- **Des défis :** les élèves peuvent par la suite modifier le code pour simuler un dé bien équilibré ou pas, et demander à un camarade de « deviner » si le dé est bien équilibré en lançant quelques simulations ; la loi des grands nombres est alors évoquée en pleine classe.
- **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

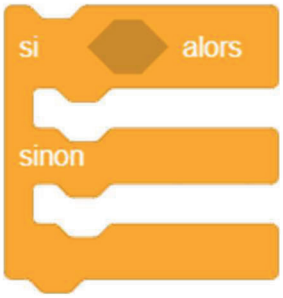
avec la TI-83 Premium CE Edition Python



Ce dé est-il truqué ?

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
	Tirer un nombre entier aléatoire entre les bornes définies par l'utilisateur, 1 et 6 ici.	`randint(0,6)` L'instruction `randint(a,b)` renvoie un nombre entier entre a et b inclus.

	Créer une structure de contrôle conditionnelle.	`if condition:`
		`••instructions si vraie`
		`else:`
		`••instructions si faux`
		L'indentation avec •• est nécessaire pour définir les instructions à exécuter dans un bloc du code Python.

	Incrémenter de 1 la valeur de la variable « reussite ».	`c+=1` Cette instruction comporte un opérateur d'affectation augmenté. Il est aussi possible d'écrire `x=x+pas`
	Effectuer le quotient des valeurs des variables « reussite » et « nb_lancers ».	`c/n`

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

Ce dé est-il truqué ?

Avec Python

Le code complet est construit avec les élèves. Il est composé de deux fonctions.

- La fonction `un_lancer`, sans paramètre, qui retourne un booléen : `True` ou `False`, selon la condition requise.

  C'est à cet endroit que l'on pourra modifier la fréquence d'apparition d'une face lors du lancer du dé en modifiant la condition `if randint(1,6)==1`.

  La condition `if randint(1,25)<=4` a été commentée, par l'appui de la touche `[2nde]` suivie de `[3]` par exemple.

  Cette ligne pourra intervenir dans un second temps de l'activité pour simuler la réalisation d'une issue de probabilité égale à $\frac{4}{25}$, valeur proche de $\frac{1}{6}$.

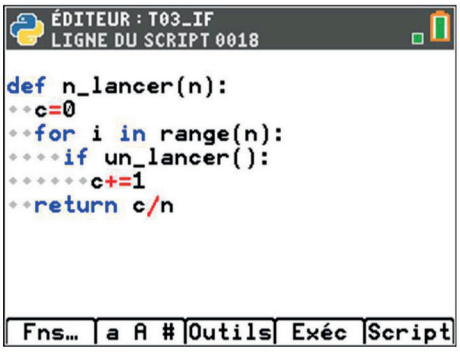
  A noter la nécessité d'importer la bibliothèque `random` pour pouvoir utiliser la fonction `randint`.

- La fonction `n_lancer` de paramètre `n`, le nombre de répétitions, qui va simuler n lancers de dés et retourner la fréquence de réussite de la fonction `un_lancer`.

  A noter dans ce code l'utilisation d'un compteur : la variable `c`, initialisée à 0, qui va cumuler les réussites de la fonction `un_lancer`.

  La valeur `c/n` correspond bien à la fréquence de réussite de la fonction `un_lancer` après n répétitions.

  La syntaxe `if un_lancer()` est très efficace : il faut bien se rappeler que `un_lancer()` est un booléen (`True` ou `False`) ; il est donc inutile de saisir : `if un_lancer()==True`.



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

Prolongements possibles

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

- demander des explications sur le fait que `n_lancer(1000)` ne donnera pas toujours la même valeur ;
- représenter graphiquement, grâce à la bibliothèque `ti_plotlib`, la fréquence de réussite de la fonction `un_lancer` en fonction du nombre de lancers ;
- travailler sur la somme des numéros des faces supérieures de deux dés lancés simultanément ;
- varier le nombre de faces ou le nombre de dés raisonnablement.

Thème : constructions géométriques

Etoile de Pompéi

TI-83 Premium CE

python™

Résumé : cette activité, assez complexe et longue, réinvestit les notions des fiches précédentes afin de construire une étoile de Pompéi, constituée d'un hexagone régulier, de carrés, de losanges et de triangles équilatéraux.

Mots-clés : bibliothèque `turtle` ; géométrie ; étoile de Pompéi

Compétences visées

Chercher : « Observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » avec ici des résultats graphiques à observer directement.

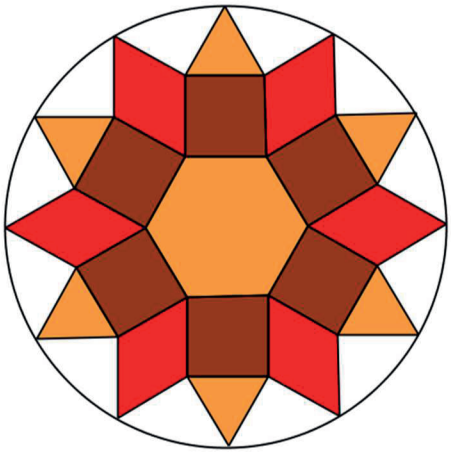
Représenter : « passer d'un mode de représentation à un autre » en commençant par une construction « à la main » puis en faisant des allers-retours entre le code et la construction.

Calculer : « mettre en œuvre des algorithmes simples », en utilisant ici des fonctions à réutiliser au cours du programme principal.

Situation déclenchante

La reproduction d'une figure complexe demande d'abord de l'analyser, puis de la décomposer en sous-problèmes.

Proposer l'étoile dite de Pompéi dont une représentation est ci-contre aux élèves en leur demandant de programmer un lutin ou `turtle`.



Travail d'un élève du collège Sidney Bechet, Alpes-Maritimes – 2019

Problématique

Comment construire une telle étoile par programmation ?

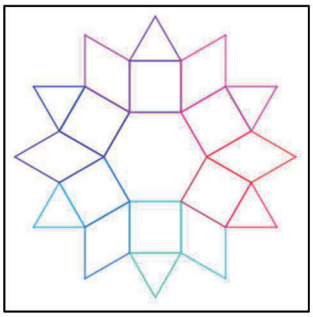
Etoile de Pompéi

Scénario pédagogique

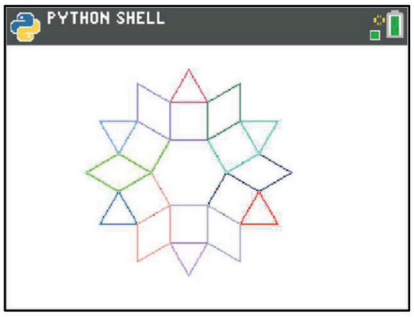
- **Avec la classe :** afficher l'image de l'étoile de Pompéi et demander aux élèves comment ils pourraient en faire une reproduction par programmation. Il est possible de demander un premier travail hors classe et de demander dans un premier temps une reproduction à la main, ou avec un logiciel de géométrie dynamique.
- **En groupe de 2 à 4 élèves :** les élèves doivent se mettre d'accord sur une procédure afin de gérer les différentes sous-figures simples qui composent la figure complexe. Ce travail nécessite un quart d'heure environ.
- **Mise en commun :** restitution de quelques groupes, parfois un seul suffit. Bien indiquer de faire par couches successives et de modifier parfois une sous-fonction en la décomposant davantage. Les difficultés de points de départ et d'arrivée, ainsi que les mesures des angles peuvent être évoquées en plénière ou au sein des petits groupes par la suite.
- **En groupe de 2 à 4 élèves :** les élèves doivent poursuivre leur programmation, qu'ils doivent finaliser hors la classe.
- **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 :

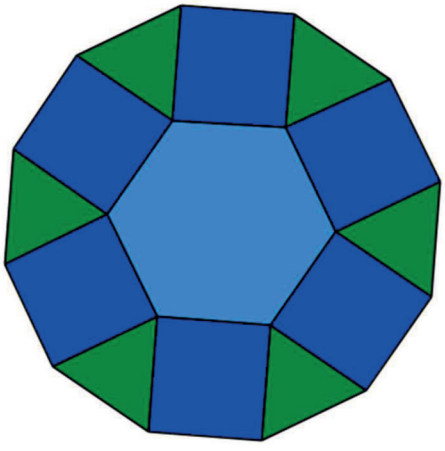
en Scratch



avec la TI-83 Premium CE Edition Python



Pour la différenciation sur cette fiche, il est possible de proposer une étoile moins complexe sur le même modèle et permettant de construire un dodécagone régulier :



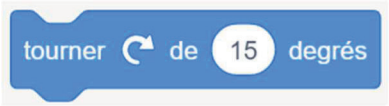
Etoile de Pompéi

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

	Définir des blocs utilisateurs nommés hexagone régulier ; carre ; triangle équilatéral et losange.	`def hexa(l):`
		`··instructions`
		L'indentation avec ·· est nécessaire pour définir les instructions à exécuter dans un bloc du code Python.

	Créer une boucle répéter.	`for i in range(n):`
		`··instructions`
		Par défaut, les valeurs de `i` commencent à 0 et augmentent de 1 à chaque itération.

	Avancer du nombre de pas indiqué, ici 10 pas ou pixels, dans une direction à définir en amont (vers la droite par défaut).	`turtle.forward(10)` *Forward* signifie « vers l'avant » en anglais.
	Tourner vers la droite d'une certaine mesure d'angle, ici 15°.	`turtle.left(90)` *Left* signifie « à gauche » en anglais.

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/

Etoile de Pompéi

Avec python

Cette fiche réinvestit les différents éléments vus dans les précédentes fiches concernant les boucles et les instructions sur `turtle`. Il faut définir de nombreuses sous-fonctions (hexagone régulier ; carré ; triangle et losange) afin de construire l'étoile en entier. Les codes suivants sont des exemples de programmation de l'étoile et d'autres codes peuvent tout à fait convenir.

Le fichier `T04_POMP.py` contient plusieurs parties. Il est nécessaire de définir les bibliothèques qui seront utilisées : `ce_turtl` pour les instructions de dessin et éventuellement `random` si une couleur aléatoire pour les différents objets géométriques est souhaitée. Par la suite, les fonctions `r`, `g` et `b` permettent de définir la couleur précédemment citée.

Les fonctions `tri` (pour les triangles équilatéraux) ; `carre` (pour les carrés) ; `los` (pour les losanges) et `hexa` (pour l'hexagone régulier) ont toutes pour paramètres `l` qui est la longueur caractéristique donnée par l'utilisateur. A la fin de chacune de ces fonctions, on trouve l'instruction `turtle.show()` en commentaire. En effet, afin de tester si les fonctions donnent le résultat attendu, il faut afficher l'objet géométrique. L'instruction `turtle.show()` provoque l'arrêt du script et dessine tout ce qui a été mis en mémoire. La mise en commentaire permet d'éviter de supprimer le code qui peut resservir lors d'autres tests.

Parmi les autres instructions de ces fonctions, on retrouve la possibilité de colorer le chemin tracé par le `turtle` avec l'instruction `turtle.color(rouge,vert,bleu)` où rouge ; vert et bleu sont des nombres entiers entre 0 et 255. Les autres instructions sont classiquement des instructions d'avance et de changement de direction du `turtle`, ainsi que des boucles `for`.

Enfin, la fonction `etoile` permet de construire l'étoile en initialisant le script, puis en lançant la sous-fonction hexa selon la longueur caractéristique `l`.

ÉDITEUR : T04_POMP
LIGNE DU SCRIPT 0001

```
from ce_turtl import *
from random import *

turtle.clear()

def r():
  return randint(0,255)
def g():
  return randint(0,255)
def b():
  return randint(0,255)

def tri(l):
  turtle.color(r(),g(),b())
  turtle.left(30)
  for i in range(3):
    turtle.right(120)
    turtle.forward(l)
  turtle.right(30)
  #turtle.show()

def carre(l):
  turtle.color(r(),g(),b())
  turtle.left(30)
  for i in range(2):
    turtle.forward(l)
    turtle.right(90)
  tri(l)
  for i in range(2):
    turtle.forward(l)
    turtle.right(90)
  turtle.right(30)
  #turtle.show()

def los(l):
  turtle.color(r(),g(),b())
  turtle.left(90)
  for i in range(2):
    turtle.forward(l)
    turtle.right(60)
    turtle.forward(l)
    turtle.right(120)
  turtle.right(90)
  #turtle.show()

def hexa(l):
  turtle.color(r(),g(),b())
  for i in range(6):
    turtle.right(60)
    turtle.forward(l)
    carre(l)
    los(l)
  #turtle.show()

def etoile(l):
  turtle.clear()
  turtle.penup()
  turtle.goto(0,l)
  turtle.pendown()
  hexa(l)
  turtle.show()
```

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

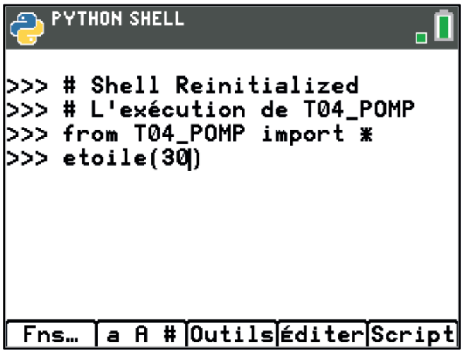
Etoile de Pompéi

Mode opératoire

Ceci concerne le code `T04_POMP.py`.

- Une fois le script exécuté, il faut appuyer sur la touche `[var]` : la ou les fonctions définies dans le script apparaissent.
- Par les flèches directionnelles, il faut sélectionner la fonction choisie (`etoile` ici), valider par `Ok`, puis ajouter la valeur du paramètre : la longueur d'un côté de l'hexagone régulier à tracer.

Pour sortir de l'écran de dessin, il faut appuyer sur la touche `[on]` de la calculatrice.



Prolongements possibles

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

- programmer un cercle circonscrit à la figure ;
- écrire un programme permettant de colorier l'intérieur des figures ; cela peut réinvestir le travail sur les spirales en fiche 3, des spirales, de ce livret ;
- chercher d'autres figures géométriques et essayer de les reproduire, comme par exemple les mosaïques arabes.

Construction de droite graduée et de repère

Résumé : dans cette activité, les élèves doivent programmer un lutin ou `turtle` afin de tracer une droite graduée, puis par extension un repère du plan, afin de pouvoir tracer des fonctions dans l'activité suivante.

Mots-clés : bibliothèque `turtle` ; droite graduée ; repère du plan

Compétences visées

Chercher : « Observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » avec ici des résultats graphiques à observer directement.

Représenter : « passer d'un mode de représentation à un autre » en comprenant bien comment représenter un axe gradué puis en finalisant la construction d'un repère orthonormé.

Calculer : « mettre en œuvre des algorithmes simples », en utilisant ici des fonctions à réutiliser au cours du programme principal.

Situation déclenchante

Tracer un simple repère orthonormé ou du moins orthogonal du plan peut s'avérer long pour certains élèves. L'utilisation du mode graphique d'une calculatrice, comme la TI-83 Premium CE Edition Python, peut sembler être une boîte noire qu'il est intéressant d'ouvrir aux élèves afin de remédier à certaines difficultés.

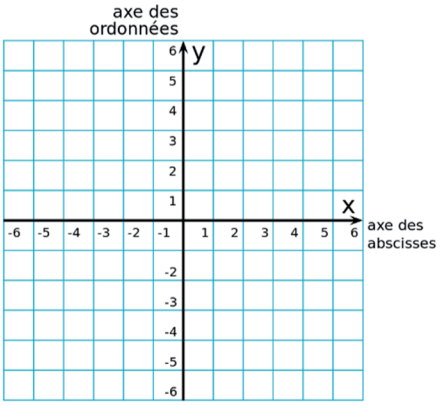


Image libre de droits d'après Wikipédia

Problématique

Comment tracer dans un premier temps une droite graduée, puis, dans un deuxième temps, un repère du plan par programmation ?

Construction de droite graduée et de repère

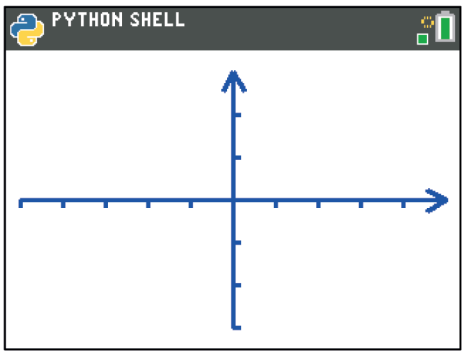
Scénario pédagogique

- **Avec la classe :** lors de la restitution d'un devoir sur la représentation graphique ou avec la volonté d'ouvrir une boîte noire de la calculatrice : le tracé des axes d'une fenêtre graphique, proposer aux élèves cette activité.
- **Individuellement ou en groupe pour la droite graduée :** il est possible de donner le code du script `T05_DTEG.py` et de demander de comprendre à quoi peut servir ce code. Il est possible aussi de demander la création du script permettant de tracer une droite graduée, en devoir hors classe ou lors d'une séance informatique.
- **Avec la classe :** faire un point en listant les différentes propositions des élèves, et donner la programmation possible `T05_GRAD.py` que les élèves peuvent récupérer en utilisant un câble mini-USB pour ceux dont la programmation est défaillante ou ne permettant pas son utilisation.
- **Individuellement ou en groupe pour le repère :** il s'agit d'inciter les élèves à utiliser le script de la droite graduée pour tracer le repère, en leur indiquant comment récupérer d'autres scripts par l'instruction `from nom du script import *`.
- **Mise en commun :** faire un point avec les élèves et donner le script `T05_REP.py`, ou un script équivalent créé par un élève si l'activité suivante de création de représentation graphique d'une fonction est envisagée.
- **Pour les élèves les plus en avance :** il est possible de leur proposer un ou plusieurs prolongements possibles, décrit en fin de fiche.
- **Difficultés rencontrées :**
    - la notion de « sous-fonction ».

Voici les visuels à l'issue des programmes :

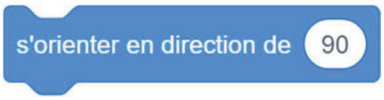
en Scratch pour la demi-droite graduée

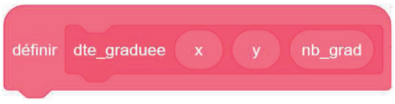
avec la TI-83 Premium CE Edition Python pour le repère

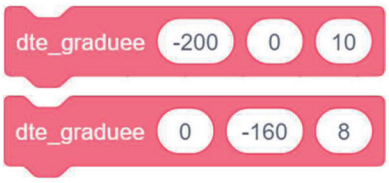


Construction de droite graduée et de repère

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
	Avancer de 10 pas ou pixels.	`turtle.forward(10)` *Forward* signifie « vers l'avant » en anglais.
	D'orienter le lutin selon un angle défini. Attention, les orientations sont différentes : voir la partie « Pour mieux lire le code Python ».	`turtle.setheading(90)` *Set heading* signifie « définir le cap » en anglais.

	Créer un bloc utilisateur avec trois paramètres : x ; y et nb_grad, les deux premiers désignent les coordonnées de départ et le troisième détermine le nombre de graduations à faire. Il faut les définir lors de la création ou modifier par la suite.	`def dte_grad(x,y,n):`
		`••instructions`
		
		L'indentation avec •• est nécessaire pour définir les instructions à exécuter dans un bloc du code Python.

	Utiliser le bloc utilisateur défini au préalable. Attention, les fenêtres d'exécution du programme sont différentes : voir la partie « Pour mieux lire le code Python ».	`dte_grad(-150,0,10)` `dte_grad(0,-90,6)`

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/

Construction de droite graduée et de repère

Avec Python

Suivant le choix de procédure, soit le code complet est à construire par les élèves, soit partiellement, en ayant donné le script `T05_DTEG.py`.

Ce code est composé de deux scripts `T05_DTEG.py` et `T05_REP.py`, composés eux-mêmes de plusieurs fonctions. Il est d'ailleurs possible de rappeler un script précédent comme dans l'exemple choisi ici :

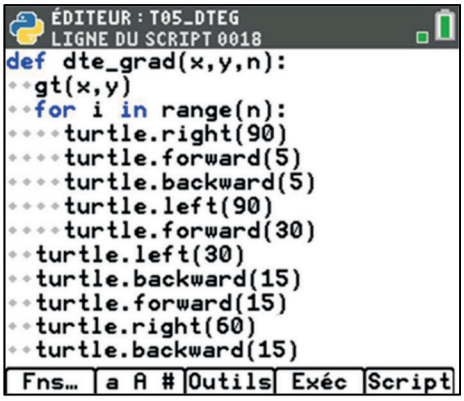
- Dans `T05_DTEG.py`, la fonction `gt` de paramètres `x` et `y` qui permet au stylo d'aller directement au point de coordonnées `(x,y)` sans écrire.

Dans `T05_DTEG.py`, la fonction `dte_grad` de paramètres `x` ; `y` et `n`.

`x` et `y` sont les coordonnées de départ de la droite graduée, tandis que `n` désigne le nombre de graduation à effectuer.

Il a été choisi de faire une graduation tous les 30 pixels, visible par l'instruction `turtle.forward(30)`.

Les cinq dernières instructions permettent de tracer la flèche indiquant le sens de la droite graduée.



- Dans `T05_DTEG.py`, la fonction `main` sans paramètre.

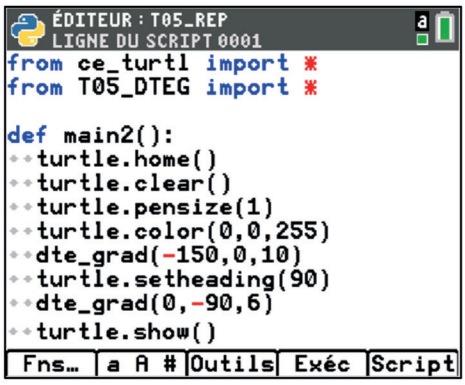
  C'est la fonction principale permettant de tracer la droite graduée.

  Après une initialisation classique, en mettant tout de même la taille du stylo au niveau moyen, le script lance la fonction `dte_grad`.

  L'instruction `turtle.show()` permet de tracer le dessin demandé.

- Dans `T05_REP.py`, la fonction `main2` sans paramètre permet de tracer le repère orthogonal du plan demandé.

  A noter que dans cette programmation, le script `T05_DTEG.py` est appelé par l'instruction `from T05_DTEG import *`. Cela permet de pourvoir utiliser les fonctions définies dans ce script.

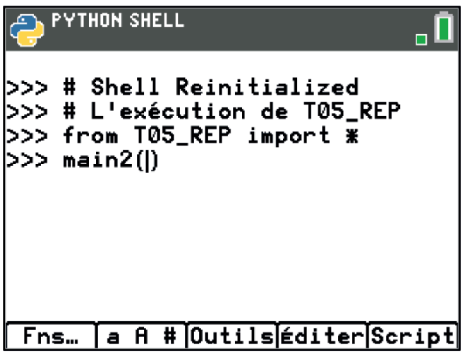


Une programmation possible est disponible sur le site TI : education.ti.com/fr/scratch-python

Construction de droite graduée et de repère

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 `main()` ou `main2()` suivant le script choisi, valider par `Ok`, puis lancer sans avoir besoin de mettre de paramètre.

Pour sortir du mode dessin, il faut appuyer sur la touche `[on]`.

Prolongements possibles

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

- programmer pour définir les axes par rapport à un point d'origine définit par l'utilisateur, de façon à pouvoir obtenir uniquement que le premier quadrant, utile en physique par exemple ;
- programmer afin de gérer l'espacement des graduations en fonction du nombre de celles-ci.

Pour mieux lire le code Python

A noter ici qu'il est nécessaire d'importer la bibliothèque `turtle` en l'appelant en début de programmation par la ligne : `from ce_turtl import *`.

Dans Scratch, 0 [degré] dirige le lutin vers le haut, et 90 [degrés] vers la droite, et ainsi de suite dans le sens horaire. Pour la calculatrice TI-83 Premium CE Edition Python, de même que dans Python, dans la version standard, le 0 [degré] désigne la droite, puis 90 [degrés] le haut, en tournant dans le sens trigonométrique.

La fenêtre d'exécution, autant dans Scratch que sur la calculatrice TI-83 Premium CE Edition Python, étant rectangulaire non carrée, il est nécessaire d'adapter les coordonnées de départ, ainsi que le nombre de graduation.

Pour information, sur Scratch, la taille d'écran est de 480 pixels par 360 pixels. Sur la calculatrice, elle est de 320 pixels par 240 pixels. Ces différences nécessitent donc des aménagements pour passer d'une programmation à l'autre, occasionnant des problèmes mathématiques qu'il peut être intéressant à développer pour les élèves les plus en avance.

Représentation graphique

Résumé : après avoir géré la construction du repère dans l'activité précédente, il s'agit ici de faire la représentation point par point ou pixel par pixel de la représentation graphique d'une fonction donnée.

Mots-clés : bibliothèque `turtle` ; fonction ; représentation graphique

Compétences visées

Chercher : « Observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » avec ici des résultats graphiques à observer directement.

Modéliser : « Utiliser, comprendre, élaborer une simulation numérique ».

Représenter : « passer d'un mode de représentation à un autre » en faisant le lien entre la représentation graphique d'une fonction que l'on peut faire sur une feuille et ce que l'on va exécuter par le code.

Calculer : « mettre en œuvre des algorithmes simples », en utilisant ici des fonctions à réutiliser au cours du programme principal.

Situation déclenchante

La calculatrice ou l'ordinateur est capable de tracer la représentation graphique d'une fonction. C'est souvent une boîte noire pour les élèves. Certains d'entre eux auront parfois du mal à tracer des fonctions à l'arrivée en seconde, pourtant déjà vu dans de nombreuses disciplines les années antérieures.

Image libre de droits d'après Pixabay

Problématique

Comment représenter graphiquement, point par point, une fonction donnée par programmation directe de la calculatrice ?

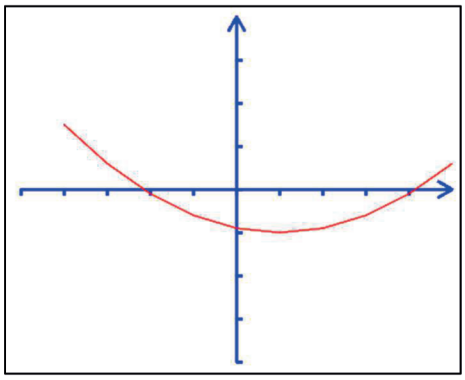
Représentation graphique

Scénario pédagogique

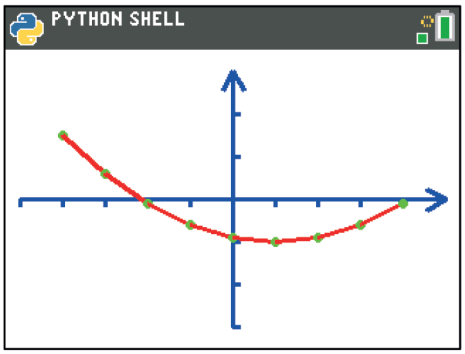
- **Avec la classe :** en relation avec l'activité précédente sur la construction d'un repère, ou en donnant le code correspondant, cette activité peut être proposée à l'ensemble de la classe, soit individuellement, soit en travail de groupe ou éventuellement dans un devoir hors classe. Il est possible de donner différentes fonctions à tracer pour inciter les élèves à réfléchir sur la problématique. L'exemple pris ici est la fonction $f : x \mapsto \frac{(x-1)^2}{10} - 1$, définie sur $[a\,;b]$ où a et b sont deux réels. Il faudra penser que par défaut la plage des ordonnées est de -3 à 3, ce qui nécessite de tester les fonctions données.
- **Individuellement ou en groupe :** il est possible de donner le code du script `T05_REP.py` en amont si l'activité précédente n'a pas été réalisée et de demander ce que permet de faire ce script. Sinon, laisser les élèves créer l'algorithme puis le script correspondant en faisant le point avec eux régulièrement.
- **Mise en commun :** faire un point avec les élèves et évoquer les difficultés rencontrées dans les groupes.
- **Pour les élèves les plus en avance :** il est possible de leur proposer un ou plusieurs prolongements possibles, décrit en fin de fiche.
- **Difficultés rencontrées :**
    - la notion de « sous-fonction » ;
    - le lien entre pixel et graduation.

Voici les visuels à l'issue des programmes :

en Scratch



avec la TI-83 Premium CE Edition Python



Représentation graphique

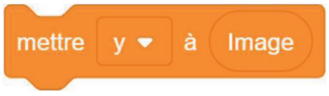
Avec Scratch

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

	Définir un bloc utilisateur nommé fonction qui comporte ici le paramètre x.	`def f(X):`
		`··instructions`
		L'indentation avec ·· est nécessaire pour définir les instructions à exécuter dans un bloc du code Python.

	Définir la fonction utilisée dans cet exemple. Attention à l'utilisation des blocs qui indiquent la priorité : chaque bloc correspond à un niveau de parenthésage, il n'y a pas ici de priorité de la multiplication sur l'addition par exemple.	`return (X-1)**2/10-1`
	Envoyer un message à un autre lutin afin qu'il trace le repère dans lequel la fonction sera représentée graphiquement.	`repere()`

	Faire une boucle qui permettra de calculer les coordonnées des points de la courbe. Par défaut, pour éviter d'alourdir les demandes, l'intervalle de définition est de −4 jusqu'à au moins 4.	`while x<x_max`
		`··instructions`
		Voir la partie « Pour mieux lire le code Python ».

	Définir les abscisses par incrément du pas que l'utilisateur aura donné.	`x+=pas` Cette instruction comporte un opérateur d'affectation augmenté. Il est aussi possible d'écrire `x=x+pas`.
	Définir l'image en se servant du bloc utilisateur fonction.	`y=f(x)` Le symbole égal signifie une affectation.
	Aller au point de coordonnées (x ; y) Le facteur « agrand » est expliqué dans la partie « Pour mieux lire le code Python ».	`turtle.goto(n(x),n(y))`

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/

Représentation graphique

Avec Python

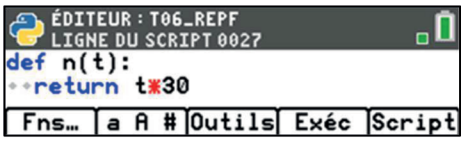
Le code complet est à construire par les élèves.

Ce code est composé de plusieurs fonctions et nécessite l'importation de la bibliothèque `turtle` et le script `T05_REP.py`.

- La fonction `repere` sans paramètre. C'est la fonction déjà utilisée dans T05_REP à laquelle il faut enlever l'instruction `turtle.show()` car celle-ci arrête toute nouvelle programmation une fois lue.
- La fonction `f` de paramètre `X` qui est l'abscisse du point à créer. C'est cette fonction qu'il convient de modifier pour obtenir la fonction à représenter.

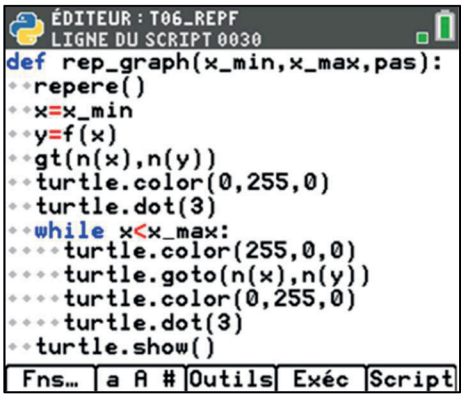
  A noter ici qu'il est nécessaire « d'écraser » la courbe par un facteur un dixième afin de visualiser une courbe.

- La fonction `n` de paramètre `t` : cette fonction permet de renvoyer `t×30`. Elle sera traitée dans la partie « Pour mieux lire le code Python ».



- La fonction principale `rep_graph` de paramètres `x_min` ; `x_max` et `pas`. Les deux premiers désignent l'intervalle sur lequel la fonction sera représentée tandis que le pas est l'incrément pour les abscisses des points constituant la courbe.
- Cette fonction permet de tracer le repère, puis, au moyen d'une boucle `while`, la courbe est tracée par des segments de droite auquel un point vert est ajouté, grâce aux instructions :

  ```
  turtle.color(0,255,0)
  turtle.dot(3)
  ```



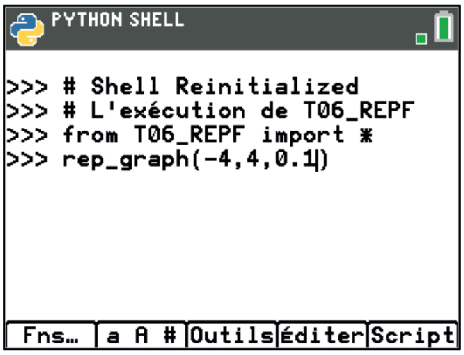
Une programmation possible est disponible sur le site TI : education.ti.com/fr/scratch-python

Représentation graphique

Mode opératoire

Une fois le script `T06_REPF.py` exécuté, il faut appuyer sur la touche `[var]` : les fonctions définies dans le script apparaissent.

Par les flèches directionnelles, il faut sélectionner la fonction `rep_graph`, valider par `Ok`, puis ajouter les trois paramètres séparés par une virgule : l'abscisse de départ, celle d'arrivée ainsi que le pas.



Prolongements possibles

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

- programmer la représentation graphique de la fonction racine carrée, en faisant attention à son ensemble de définition.

Pour mieux lire le code Python

Il faut faire attention avec les boucles « répéter tant que » (*while*) et « répéter jusqu'à ce que » (*until*). En effet, sans une condition d'arrêt, ces boucles peuvent créer une erreur en s'exécutant de façon infinie.

Il faudra appuyer sur la touche `[on]` pour arrêter le script.

Il n'y a pas de boucle « répéter tant que » dans Scratch, et pas de boucle « répéter jusqu'à ce que » dans le langage Python.

Le facteur « agrand » dans Scratch ou la fonction `n` dans Python sont nécessaires car l'écran est défini en pixel et on souhaite définir la fonction par rapport à une unité suffisamment visible à l'écran. Ainsi une unité vaudra 30 pixels dans le script Python.

Si un élève souhaite utiliser la fonction racine carrée, il faudra faire appel à la bibliothèque `math`. Il devra ensuite utiliser la fonction `sqrt()`.

Programme de calculs

Résumé : dans cette activité, les élèves doivent comparer deux programmes de calculs et trouver des antécédents éventuels à un troisième programme de calculs.

Mots-clés : programme de calculs ; fonctions ; calcul littéral

Compétences visées

Modéliser : « Utiliser, comprendre, élaborer une simulation numérique ».

Représenter : « passer d'un mode de représentation à un autre » en comparant des calculs faits « à la main » et les résultats issus des programmes.

Calculer : « mettre en œuvre des algorithmes simples »

Situation déclenchante

Les trois programmes de calculs suivants sont donnés aux élèves, qu'il faut coder en Scratch ou Python.

Programme A

- Choisir un nombre.
- Lui ajouter 2.
- Calculer le carré du résultat obtenu.
- Soustraire au résultat le carré du nombre de départ.

Programme B

- Choisir un nombre.
- Ajouter 1 à ce nombre.
- Prendre le quadruple du nombre obtenu.

Programme C

- Choisir un nombre.
- Prendre son opposé.
- Ajouter 3.
- Mettre le résultat au carré.

Image libre de droits d'après Pixabay

Problématique

Comparer les programmes de calculs A et B.

Est-il possible de trouver un (des) nombre(s) pour que le résultat du programme C soit 25 ? Et 2 ? Et −4 ?

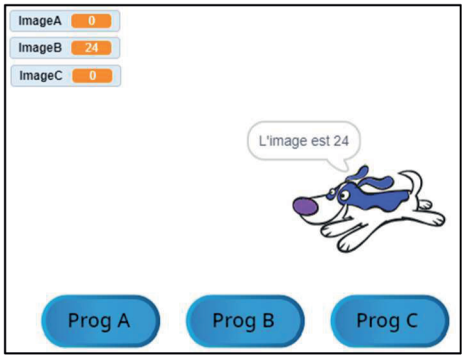
Programme de calculs

Scénario pédagogique

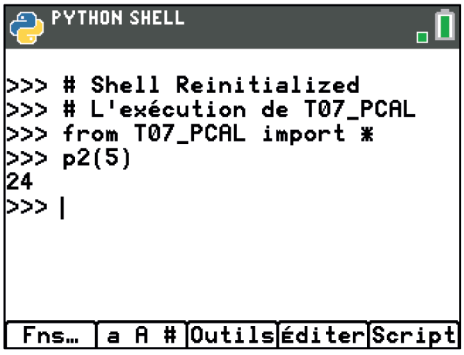
- **Avec la classe :** demander la programmation des trois programmes de calculs. Il est possible de faire cette partie en devoir hors classe.
- **En groupe de 2 élèves :** les élèves doivent ensuite répondre à la problématique. Pour la première partie, chaque élève du groupe peut tester un même nombre selon le programme de calculs attribué, ce qui permet de gagner du temps, et de confronter leurs idées par la suite.
- **Preuve :** les élèves doivent prouver leur conjecture en utilisant le calcul littéral : les programmes A et B sont égaux, il y a deux antécédents de 25, ainsi que de 2, mais aucun pour -4 pour le programme de calculs C. A noter que les antécédents de 2 sont exactement $3-\sqrt{2}$ et $3+\sqrt{2}$, nombres abscons pour certains élèves.
- **Mise en commun :** les élèves présentent leurs résultats de recherche et de preuve.
- **Pour les élèves les plus en avance :** il est possible de leur proposer un ou plusieurs prolongements possibles, décrit en fin de fiche.
- **Difficultés rencontrées :**
    - le mot « comparer » n'est pas toujours très bien compris des élèves ;
    - faire aussi le programme de calculs « à la main » pour s'assurer que les valeurs soient cohérentes ;
    - pour le programme C, les solutions négatives ou exactes ne sont pas d'emblée trouvées ni même parfois accessibles pour certains élèves.

Voici les visuels à l'issue des programmes :

en Scratch

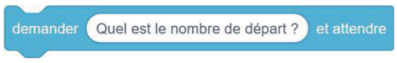
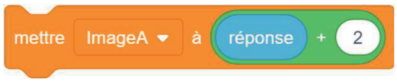
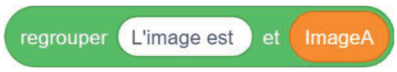


avec la TI-83 Premium CE Edition Python



Programme de calculs

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
	Demander une valeur à l'utilisateur.	Une fonction Python peut nécessiter la saisie d'aucun, d'un ou plusieurs paramètres.
	Affecter une valeur à la variable ImageA.	`b=a+2` La variable **a** est ici le paramètre correspondant de la fonction codant le programme de calculs A.
	Mettre au carré la variable ImageA demandée à l'utilisateur ; il n'existe pas de fonction puissance dans Scratch. De plus, le carré est réaffecté à la variable ImageA.	`b=b**2` `**` désigne la séquence puissance dans Python.
	Regrouper du texte et des variables pour les afficher dans un message texte.	L'utilisation de la fonction return limite les interactions avec l'utilisateur. Voir la partie « Pour mieux lire le code Python ».

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/

Programme de calculs

Avec Python

Le code complet est à construire par les élèves.

Ce code est composé de trois fonctions :

- La fonction `p1` de paramètre `a` qui est le nombre de départ. Cette fonction correspond au programme de calculs A. Elle est complètement détaillée comme le programme de calculs point par point.

  Les variables `b` ; `c` et `d` sont utilisées pour retenir chaque étape du programme de calculs afin de s'en resservir éventuellement.

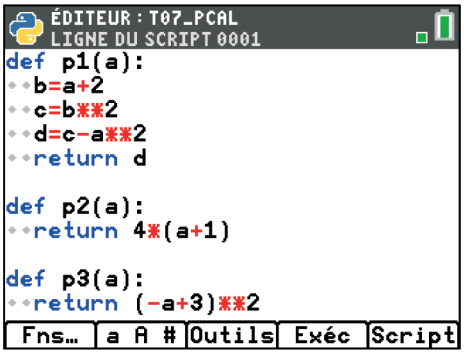
- La fonction `p2` de paramètre `a` qui est le nombre de départ

  Cette fonction correspond au programme de calculs B.

  Dans cette programmation possible, une seule ligne a été écrite, ne nécessitant pas de variable supplémentaire.

- La fonction `p3` de paramètre `a` qui est le nombre de départ.

  Cette fonction correspond au programme de calculs C.

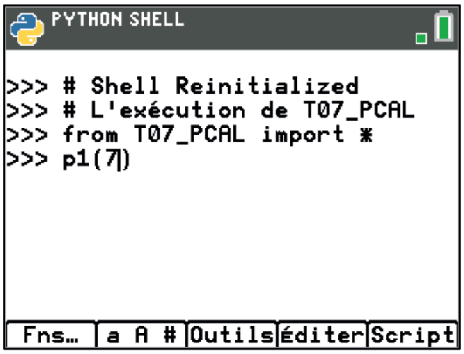


Une programmation possible est disponible sur le site TI : education.ti.com/fr/scratch-python

Mode opératoire

Une fois le script `T07_PCAL.py` exécuté, il faut appuyer sur la touche `[var]` : les trois fonctions définies dans le script apparaissent.

Par les flèches directionnelles, il faut sélectionner la fonction correspondante au programme de calculs à tester et valider par `Ok`. Il faut ensuite ajouter le nombre de départ à tester.



Prolongements possibles

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

- donner des programmes de calculs plus complexes à réaliser ;
- définir en une ligne la programmation des fonctions si l'élève ne l'a pas déjà fait.

Golf : comment réussir son putt ?

Résumé : dans cette activité, l'élève doit déterminer les valeurs du coefficient directeur et de l'ordonnée à l'origine afin que l'équation de la droite correspondante permette de relier deux points. Cette situation se déroule dans le cadre d'une partie de golf où l'on doit faire rentrer la balle dans le trou. L'élève pourra de lui-même se rendre compte s'il a juste ou pas.

Mots-clés : équation de droite ; chercher ; raisonner ; bibliothèque `turtle`

Compétences visées

Chercher : « observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » avec ici un résultat qui s'observe directement par la construction de la figure.

Modéliser : « utiliser, comprendre, élaborer une simulation numérique ». Il s'agit en particulier ici de travailler avec un repère qui n'est pas représenté.

Calculer : « contrôler les calculs » ; ici la construction permet de savoir si la réponse est juste ou pas.

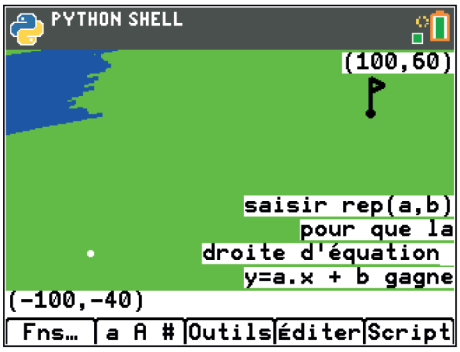
Situation déclenchante

Un golfeur doit faire rentrer sa balle dans le trou matérialisé par un drapeau. En supposant que le terrain soit plat, il faut donc construire une droite passant par le point de départ de la balle et la base du drapeau pour avoir une trajectoire gagnante : comment procéder ?

Image libre de droits d'après Pexels

Problématique

Les positions de la balle et du trou étant données par des coordonnées, quelle est l'équation de la droite qui permettra à la balle de tomber dans le trou ? Il est sous-entendu que le terrain est parfaitement plat et que la modélisation donnée est une vue de dessus.



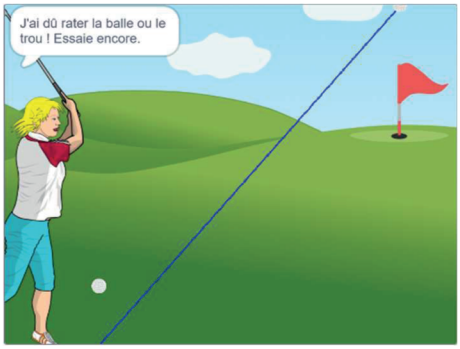
Golf : comment réussir son putt ?

Scénario pédagogique

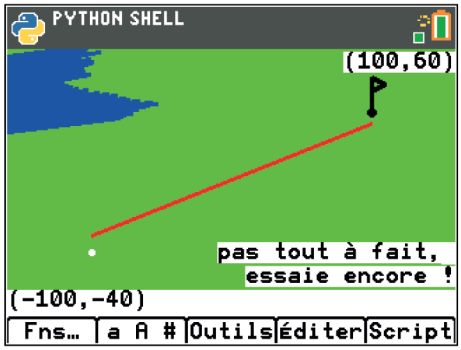
- **Avec la classe :** explicitation du problème par un court échange à l'oral avec la classe.
- **Tests :** les élèves testent individuellement le script pour s'approprier le problème.
- **Utilisation d'un code :**
    - Dans cette activité, l'outil informatique fournit une visualisation du résultat obtenu après calculs de l'élève et lui permet de se rendre compte par lui-même de la justesse de ses résultats.
    - Le code est fourni à l'élève : charge à lui de saisir les valeurs numériques du coefficient directeur et de l'ordonnée à l'origine et réagir en conséquence.
- **Mise en commun** (éventuellement au sein de petits groupes) : quels outils mathématiques sont en jeu ? Le problème mathématique posé est un peu plus complexe qu'il n'y paraît dans la mesure où seules les coordonnées sont fournies sans le repère.
- **Preuve :** les élèves doivent utiliser des méthodes permettant de déterminer les valeurs du coefficient directeur et de l'ordonnée à l'origine sans pouvoir faire de lecture graphique. S'ils ont trouvé après plusieurs essais, des calculs rigoureux sont attendus.
- **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

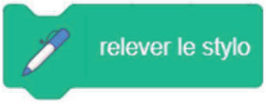
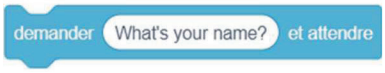
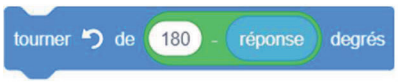


avec la TI-83 Premium CE Edition Python



Golf : comment réussir son putt ?

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
	Poser le stylo, depuis le centre du lutin par défaut, afin de laisser une trace.	`turtle.pendown()`
	Relever le stylo et donc de déplacer le lutin sans laisser de trace.	`turtle.penup()`
	Demander une valeur à l'utilisateur.	Une fonction Python peut nécessiter la saisie d'aucun, d'un ou plusieurs paramètres.
	Obtenir la réponse de l'utilisateur.	Cette valeur sera rentrée comme paramètre dans une fonction.
	Tourner vers la gauche de l'angle indiqué, ici de la différence entre 180° et la réponse donnée par l'utilisateur.	`turtle.left(180-a)`
	Avancer de 10 pixels selon une direction indiquée précédemment, vers la droite par défaut.	`turtle.forward(10)`

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/

Golf : comment réussir son putt ?

Avec Python

Le code complet est fourni aux élèves, dont voici quelques extraits.

La bibliothèque `random` permettra d'utiliser `randint` (utilisé pour le dessin du lac).

La bibliothèque `ti_system` est nécessaire à l'instruction `dispt_at` qui permet de positionner une chaîne de caractère sur l'écran.

Ce code est composé de plusieurs fonctions :

- La fonction `gt` de paramètres `x` et `y` qui permet au stylo d'aller directement au point de coordonnées $(x\,;\,y)$ sans écrire.

```
ÉDITEUR : T08_GOLF
LIGNE DU SCRIPT 0001
from ce_turtl import *
from random import *
from ti_system import *

turtle.clear()

def gt(x,y):#pour se déplacer sa
    ns écrire
  turtle.penup()
  turtle.goto(x,y)
  turtle.pendown()
Fns… | a A # | Outils | Exéc | Script
```

- La fonction `fig` (fonction sans paramètre) qui permet de créer la figure de fond de la situation. On n'en donne que le début ci-contre.

```
ÉDITEUR : T08_GOLF
LIGNE DU SCRIPT 0022
def fig():
  gt(-160,-75)
  for i in range(-75,110,5):
    turtle.color(0,255,0)
    turtle.pensize(2)
    turtle.goto(-160,i)
    turtle.goto(160,i)
  gt(-160,110)
  x=-80
  for i in range(0,55):
    turtle.color(0,0,255)
Fns… | a A # | Outils | Exéc | Script
```

- La fonction `consigne` (fonction sans paramètre) qui reprend la fonction précédente en ajoutant une aide à la saisie de la réponse.

```
ÉDITEUR : T08_GOLF
LIGNE DU SCRIPT 0044
def consigne():
  fig()
  disp_at(7,"saisir rep(a,b)","r
    ight")
  disp_at(8,"pour que la","right
    ")
  disp_at(9,"droite d'équation",
    "right")
  disp_at(10,"y=a.x + b gagne","
    right")
  turtle.show()
Fns… | a A # | Outils | Exéc | Script
```

La fonction `rep` : elle a pour paramètres la valeur du coefficient directeur et la valeur de l'ordonnée à l'origine.

Elle permet la visualisation de la réponse par le tracé du segment porté par la droite d'équation $y = ax + b$ (avec des abscisses comprises entre -100 et 100).

```
ÉDITEUR : T08_GOLF
LIGNE DU SCRIPT 0059

def rep(a,b):
  gt(-160,-75)
  fig()
  gt(-100,a*-100+b)
  turtle.pendown()
  turtle.color(255,0,0)
  turtle.goto(100,a*100+b)
  turtle.show()
Fns… | a A # | Outils | Exéc | Script
```

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

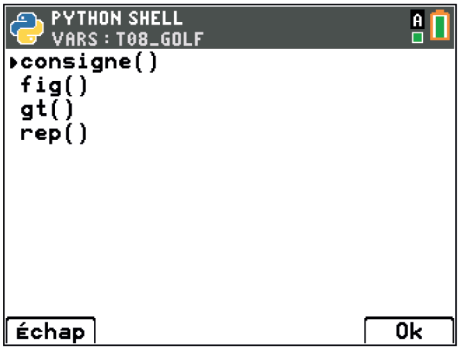
Golf : comment réussir son putt ?

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 valider la fonction `consigne`. L'élève voit apparaître la consigne. Il sort de cet écran en appuyant sur la touche [on].

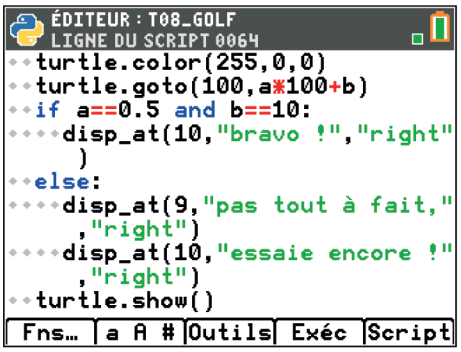
Il va chercher la fonction `rep` dans laquelle il saisit la valeur de `a` (coefficient directeur) et la valeur de `b` (ordonnée à l'origine). Par exemple, il peut saisir `rep(5.8,10)` ; attention à la signification de la virgule en Python, qui permet de séparer des paramètres, et du point. Dans le système anglo-saxon, le point décimal joue le rôle de la virgule décimale dans un nombre.



Prolongements possibles

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

- ajouter à la fonction `rep` un texte permettant de valider ou pas la réponse de l'élève ;
- modifier le code en jouant sur la position de la balle au départ et/ou celle du drapeau ;
- générer un point de départ aléatoire pour éviter la modification facile du code « aller à $(-60\,;60)$ » en fin de script par des élèves ;
- modifier le contexte en demandant une réponse sous la forme d'une trajectoire courbe ; on peut en effet imaginer le sol non plat et demander une trajectoire portée par une parabole par exemple.



Pour mieux lire le code Python

Il faut bien comprendre que dès que le script rencontre l'instruction `turtle.show`, l'écran propose une image : le dessin demandé par le script. Pour quitter cet écran, il faut appuyer sur la touche [on].

C'est pourquoi on a ici proposé une fonction `consigne` et une fonction `rep`.

Ces deux fonctions appellent la fonction `cons1`, ce qui permet d'éviter de recopier plusieurs fois le même texte.

Bien avoir en tête les coordonnées des extrémités de l'écran : $(-160\,;\,115)$ pour le coin supérieur gauche et $(160\,;-115)$ pour le coin inférieur droit.

Le L : retour au point de départ !

Résumé : dans cette activité, l'élève doit déterminer, dans un triangle rectangle la mesure d'un angle ainsi que la longueur du troisième côté du triangle afin de revenir au point de départ. L'élève commence par chercher des pistes à l'aide d'un programme fourni, puis raisonner pour démontrer mathématiquement son résultat.

Mots-clés : trigonométrie ; chercher ; raisonner ; bibliothèque `turtle` ; théorème de Pythagore

Compétences visées

Chercher : « observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels » avec ici un résultat qui s'observe directement par la construction de la figure.

Modéliser : « utiliser, comprendre, élaborer une simulation numérique ». Il s'agit en particulier ici de placer ce problème dans un contexte mathématique.

Calculer : « contrôler les calculs » ; ici la construction permet de savoir si la réponse est juste ou pas.

Situation déclenchante

Un bras robot doit graver un **L** sur un objet au laser. Pour cela, il commence à marquer un premier segment vertical de 100 mm, puis un deuxième segment, cette fois horizontal, de 80 mm.

Avant que l'objet suivant se présente pour être gravé, le robot doit reprendre sa position initiale en parcourant le chemin le plus court possible.

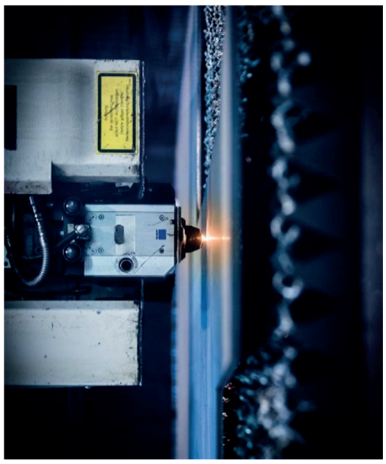


Image libre de droits d'après Pixabay

Problématique

Quelles instructions donner au robot pour revenir le plus rapidement à son point de départ ?

Le L : retour au point de départ !

Scénario pédagogique

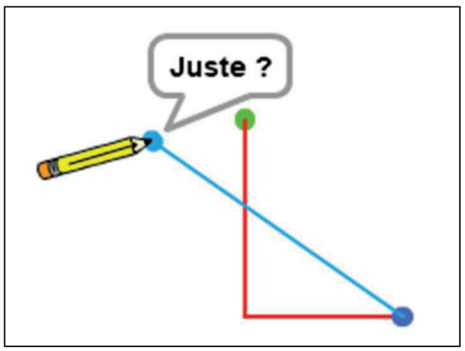
- **Avec la classe :** explicitation du problème par un court échange à l'oral avec la classe.
- **Tests :** les élèves testent individuellement le script pour s'approprier le problème.

  Utilisation d'un code :

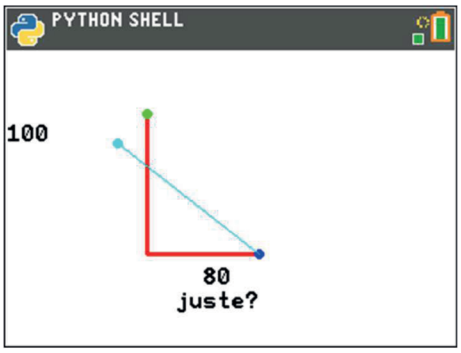
  - Dans cette activité, l'outil informatique fournit une visualisation du résultat obtenu après calculs de l'élève et lui permet de se rendre compte par lui-même de la justesse de ses résultats.
  - Le code est fourni à l'élève : charge à lui de saisir les valeurs numériques de l'angle et de la longueur de l'hypoténuse pour observer le résultat et réagir en conséquence.
- **Mise en commun** (éventuellement au sein de petits groupes) : quels outils mathématiques sont en jeu ? Le problème mathématique posé est double ; quel est l'angle au sommet du triangle qui sera formé après le retour, et quelle est la longueur de l'hypoténuse ?
- **Preuve :** les élèves doivent utiliser le théorème de Pythagore pour déterminer la longueur du 3e côté et la trigonométrie pour déterminer la mesure de l'angle de retour.
- **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

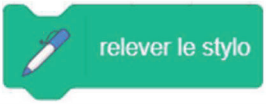
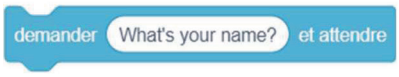
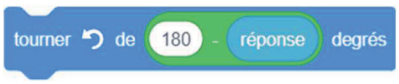


avec la TI-83 Premium CE Edition Python



Le L : retour au point de départ !

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
	Poser le stylo, depuis le centre du lutin par défaut, afin de laisser une trace.	`turtle.pendown()`
	Relever le stylo et donc de déplacer le lutin sans laisser de trace.	`turtle.penup()`
	Demander une valeur à l'utilisateur.	Une fonction Python peut nécessiter la saisie d'aucun, d'un ou plusieurs paramètres.
	Obtenir la réponse de l'utilisateur.	Cette valeur sera rentrée comme paramètre dans une fonction.
	Tourner vers la gauche de l'angle indiqué, ici de la différence entre 180° et la réponse donnée par l'utilisateur.	`turtle.left(180-a)`
	Avancer de 10 pixels selon une direction indiquée précédemment, vers la droite par défaut.	`turtle.forward(10)`

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/

Le L : retour au point de départ !

Avec Python

Le code complet est fourni aux élèves, dont voici quelques extraits.

Ce code est composé de plusieurs fonctions :

- La fonction `gt` de paramètres `x` et `y` qui permet au stylo d'aller directement au point de coordonnées $(x\,;\,y)$ sans écrire.

```
ÉDITEUR : ROBOT
LIGNE DU SCRIPT 0011
from ce_turtl import *
from ti_system import *

turtle.clear()

def gt(x,y):
  turtle.penup()
  turtle.goto(x,y)
  turtle.pendown()

Fns… | a A # | Outils | Exéc | Script
```

- La fonction `cons1` (fonction sans paramètre) qui permet de rédiger une consigne sans l'afficher.
- La fonction `consigne1` (fonction sans paramètre) qui reprend la fonction précédente en ajoutant une aide à la saisie de la réponse.

```
ÉDITEUR : ROBOT
LIGNE DU SCRIPT 0022
def cons1():
  turtle.clear()
  disp_at(4,"100","left")
  disp_at(10,"80","center")
  gt(-60,60)
  turtle.pensize(1)
  turtle.color(255,0,0)
  turtle.goto(-60,-40)
  turtle.goto(20,-40)
  turtle.color(0,0,255)
  turtle.dot(3)
  gt(-60,60)
  turtle.color(0,255,0)
  turtle.dot(3)
Fns… | a A # | Outils | Exéc | Script
```

La fonction `rep1` : elle a pour paramètres un angle exprimé en degrés et une longueur, celle du troisième côté.

Cette fonction reprend une partie de la consigne et affiche en sortie le segment construit à partir des données saisies par l'élève ainsi que le texte "juste?" centré à la onzième ligne du code.

```
ÉDITEUR : ROBOT
LIGNE DU SCRIPT 0033
def rep1(a,l):
  cons1()
  turtle.color(0,255,255)
  turtle.pensize(0)
  gt(20,-40)
  turtle.setheading(0)
  turtle.left(180-a)
  turtle.forward(l)
  turtle.dot(3)
  disp_at(11,"juste?","center")
  turtle.show()
Fns… | a A # | Outils | Exéc | Script
```

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

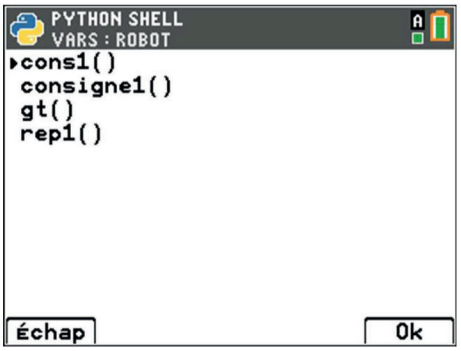
Le L : retour au point de départ !

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 valider la fonction `consigne1`. L'élève voit apparaître la consigne. Il sort de cet écran en appuyant sur la touche [on].

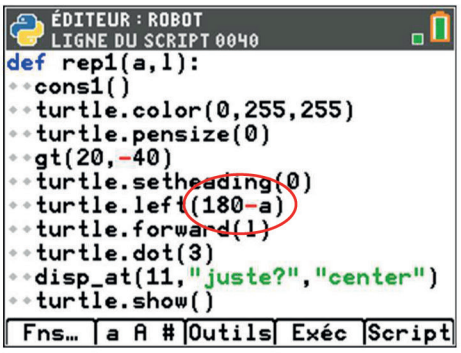
Il va chercher la fonction `rep1` dans laquelle il saisit l'angle et la longueur qu'il avait déterminée : par exemple `rep1(28.6,125)` ; attention à la signification de la virgule en Python, qui permet de séparer des paramètres, et du point. Dans le système anglo-saxon, le point décimal joue le rôle de la virgule décimale dans un nombre.



Prolongements possibles

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

- comprendre pourquoi l'angle à saisir pour tourner à gauche est `180-a` et non pas `a` ;
- demander à modifier le code pour effectuer une gravure en **L** avec d'autres dimensions ;
- générer un point de départ aléatoire pour éviter la modification facile du code « aller à $(-60\,;60)$ » en fin de script par des élèves ;
- demander de coder d'autres lettres simples comme le H, le T ou le X, voire plus complexes avec des lettres arrondies comme le S.



Pour mieux lire le code Python

Il faut bien comprendre que dès que le script rencontre l'instruction `turtle.show`, l'écran propose l'image codée précédemment. Pour quitter cet écran, il faut appuyer sur la touche [on].

C'est pourquoi on a ici proposé une fonction `consigne` et la fonction `rep`.

Ces deux fonctions appellent la fonction `cons1`, ce qui permet d'éviter de recopier plusieurs fois le même texte.

Bien avoir en tête les coordonnées des extrémités de l'écran : $(-160\,;115)$ pour le coin supérieur gauche et $(160\,;-115)$ pour le coin inférieur droit.

Thème : géométrie

Réciproque du théorème de Pythagore

TI-83 Premium CE

python™

Résumé : cette fiche porte sur la programmation de la réciproque du théorème de Pythagore.

Mots-clés : bibliothèque `turtle` ; réciproque du théorème de Pythagore

Compétences visées

Chercher : « Observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels ».

Modéliser : « Utiliser, comprendre, élaborer une simulation numérique » en partant d'une situation usuelle rencontrée fréquemment en mathématiques.

Représenter : « passer d'un mode de représentation à un autre » en transférant des calculs sur feuille à un code.

Calculer : « mettre en œuvre des algorithmes simples ».

Situation déclenchante

Il s'agit d'une séance interne aux mathématiques dans laquelle les élèves doivent programmer la réciproque au théorème de Pythagore.

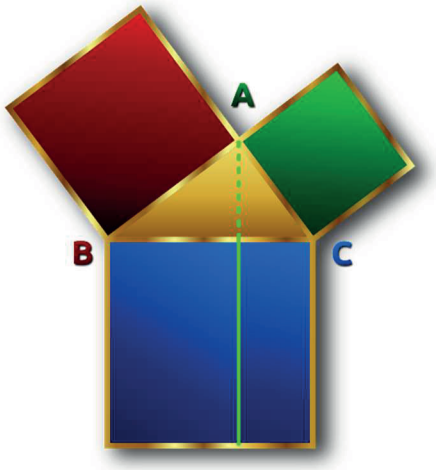


Image libre de droits d'après Pixabay

Problématique

Quel programme pour indiquer si un triangle est rectangle ou non, connaissant ses trois longueurs ?

Réciproque du théorème de Pythagore

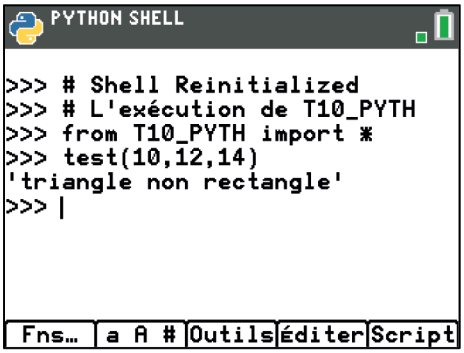
Scénario pédagogique

- **Avec la classe :** on pourra proposer beaucoup de situations impliquant la problématique, comme par exemple : « Le triangle 10 ; 12 ; 14 est-il rectangle ? », ce qui devrait faire émerger l'intérêt d'un script permettant d'indiquer si un triangle est rectangle ou non.
- **En groupe de 2 à 4 élèves pendant quelques minutes :** les élèves cherchent des stratégies pour répondre à la problématique. Beaucoup veulent passer par un aspect graphique (voir la fiche n°10, Le L). La réciproque du théorème de Pythagore n'émerge pas de suite pour certains groupes.
- **Avec la classe :** faire un point en listant les différentes propositions, en demandant finalement quelles sont les entrées (les trois longueurs du triangle), ce qui, par conséquence, est possible ou non. Par exemple, l'utilisation d'un dessin nécessite la trigonométrie et par conséquent de savoir que le triangle est rectangle.
- **En groupe de 2 à 4 élèves :** les élèves doivent faire une ébauche d'algorithme de la réciproque du théorème de Pythagore, avant de passer à l'implantation dans un langage ou un autre, après validation du professeur. Cette partie nécessite un temps assez long pour que les élèves s'approprient le script.
- **Mise en commun :** les élèves expliquent leurs démarches.
- **Pour les élèves les plus en avance :** il est possible de leur proposer un ou plusieurs prolongements possibles, décrit en fin de fiche.
- **Difficultés rencontrées :**
    - prévoir qu'une donnée soit celle du côté le plus grand ;
    - mettre les données au carré.

Voici les visuels à l'issue des programmes :

en Scratch

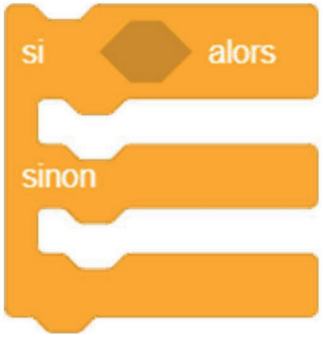
avec la TI-83 Premium CE Edition Python



Réciproque du théorème de Pythagore

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
	Affecter une valeur à la variable a, 0 ici ;	`a=0`
	Mettre au carré la réponse demandée à l'utilisateur ; il n'existe pas de fonction puissance dans Scratch ;	`a**2` `**` désigne la séquence puissance dans Python

	Créer une structure de contrôle conditionnelle.	`if condition:`
		`··instructions si vraie`
		`else:`
		`··instructions si faux`
		
		L'indentation avec `··` est nécessaire pour définir les instructions à exécuter dans un bloc du code Python.

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

Réciproque du théorème de Pythagore

Avec Python

Le code complet est à construire par les élèves.

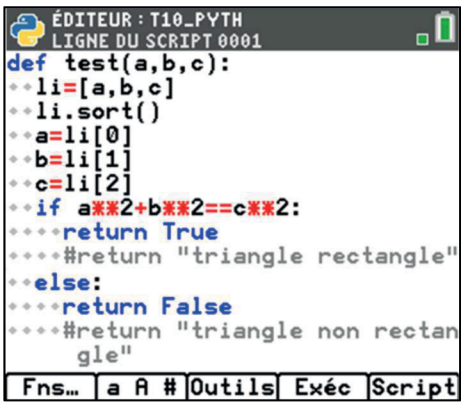
Ce code est composé de deux fonctions, seule la première est attendue des élèves, la seconde permet une ouverture vers les triplets pythagoriciens :

- La fonction `test` de paramètres `a` ; `b` et `c` qui sont les trois longueurs du triangle à tester.

  Les paramètres sont insérés dans la liste `li`, ce qui permet de les ordonner de façon croissante avec la fonction `li.sort()`.

  Deux options pour le retour à l'utilisateur : soit par un message texte, soit par un retour sous forme de booléen `True`/`False`.

  Cette deuxième option permet le réinvestissement de la fonction `test` dans la fonction suivante.

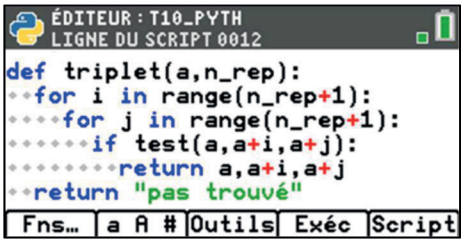


- La fonction `triplet` de paramètres `a` et `n_rep` qui sont respectivement la longueur de départ et l'étendue à tester.

  Ainsi, à partir d'une valeur (longueur) déterminée, tous les triplets entiers sont testés depuis (`a` ; `a` ; `a`) jusqu'à (`a` ; `a+n_rep` ; `a+n_rep`). Le script s'arrête à la première valeur trouvée ou indique qu'il n'y a pas de tels triplets sur l'intervalle testé.

  Cela est rendu possible grâce à la fonction `test`. Il est pratique de renvoyer un booléen.

  A noter que dans la boucle `for`, il faut aller jusqu'à `n_rep+1` puisque la boucle commence à 0 et s'arrête à l'entier qui précède `n_rep+1`.



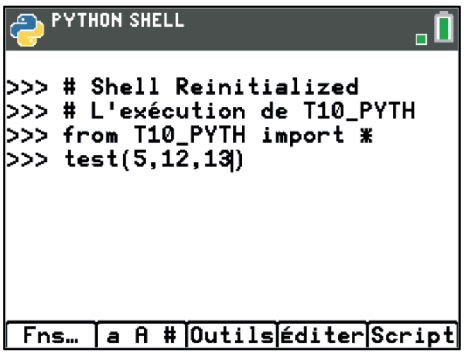
Une programmation possible est disponible sur le site TI : education.ti.com/fr/scratch-python

Réciproque du théorème de Pythagore

Mode opératoire

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

Par les flèches directionnelles, il faut sélectionner la fonction `test`, valider par `Ok`, puis rajouter les trois longueurs du triangle à tester en séparant les paramètres par une virgule.



Prolongements possibles

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

- décliner un programme permettant d'indiquer en quel point le triangle est rectangle ;
- faire un algorithme de tri ;
- déterminer un triplet pythagoricien à partir d'un nombre et d'une étendue (script `triplet`) ;
- déterminer la mesure des angles dans le cas d'un triangle rectangle.

171 pas

Résumé : il s'agit d'un travail collaboratif dans lequel les élèves cherchent à trouver des emplacements possibles de coffres au trésor. Il est préférable d'avoir fait la fiche précédente sur la réciproque du théorème de Pythagore.

Mots-clés : bibliothèque `math` ; théorème de Pythagore

Compétences visées

Chercher : « Observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels ».

Modéliser : « Utiliser, comprendre, élaborer une simulation numérique ».

Représenter : « passer d'un mode de représentation à un autre »

Calculer : « mettre en œuvre des algorithmes simples ».

Situation déclenchante

Mathurin le pirate a caché des trésors et il a laissé des instructions bien étranges pour les retrouver.

Dans un grand désert bien plat, il a caché des trésors selon une méthode bien précise :

« Depuis le point de départ en noir sur la carte ci-contre, il faut faire exactement 171 pas vers le nord, puis faire un nombre entier de pas vers l'est au moins égal à 171 pas. S'il est possible de revenir au point de départ par un nombre entier de pas, alors il faut creuser ! »

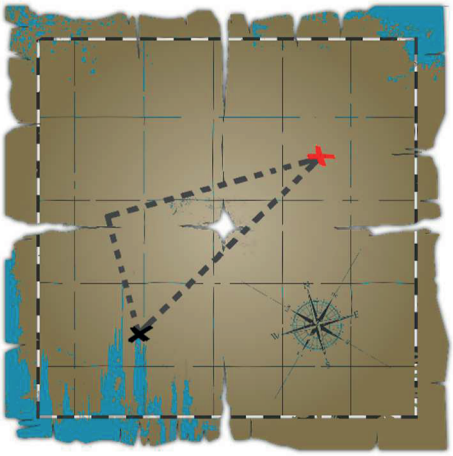


Image libre de droits d'après Pixabay

Problématique

Quels sont tous les emplacements des trésors ?

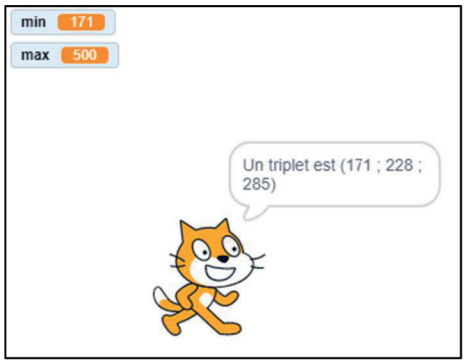
171 pas

Scénario pédagogique

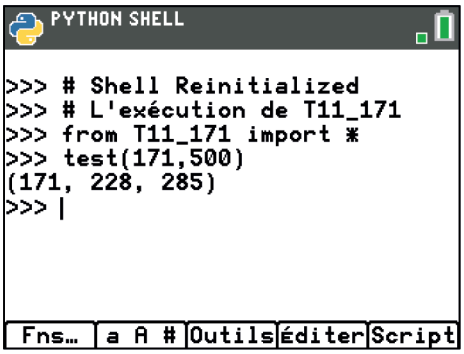
- **Avec la classe :** proposer le problème précédent et s'assurer que tous les groupes ont compris l'utilisation du théorème de Pythagore pour cette activité. Cela pourra être facilitée par l'utilisation préalable de la fiche précédente sur le théorème de Pythagore.
- **En groupe de 2 à 4 élèves :** les élèves doivent faire une ébauche d'algorithme permettant la résolution du problème proposé, avant de passer à l'implantation dans un langage ou un autre, après validation du professeur. Cette partie peut nécessiter un temps assez long pour que les élèves s'approprient le programme, notamment par la maîtrise de fonctions dédiées : boucle `while` par exemple.
- **Mise en commun :** les élèves expliquent leurs démarches.
- **Preuve :** les élèves peuvent montrer qu'il existe un nombre fini de solutions.
- **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

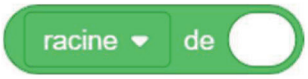


avec la TI-83 Premium CE Edition Python

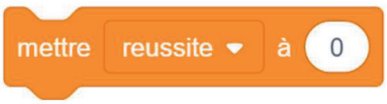
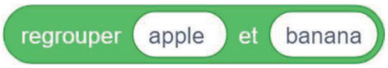


171 pas

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
	Prendre la racine carrée du nombre en paramètre.	`sqrt()` C'est l'abréviation de ***square root***, signifiant « racine carrée » en anglais.

	Répéter les instructions contenues dans la boucle jusqu'à ce que la condition se réalise.
	Il n'y a pas de boucle « répéter tant que » dans Scratch, et pas de boucle « répéter jusqu'à ce que » dans le langage Python.
	Attention donc à l'adaptation en langage Python.
		`while condition inverse :`
		`••instructions`
		
		L'indentation avec •• est nécessaire pour définir les instructions à exécuter d'une structure dans le code Python.

	Créer une condition dans laquelle il est testé si la partie entière (« plancher ») de la valeur de la variable c est égale à elle-même. Le mot « plancher » est une traduction de *floor* en anglais. C'est une condition souvent utilisée.	`int(c)==c` `int` est l'abréviation de ***integer***, signifiant « entier » en anglais. Ici, il s'agit de la fonction partie entière. A noter que pour une condition dans Python, il faut mettre un double égal, le simple égal signifiant une affectation.
	S'assurer de savoir s'il y a au moins une solution, de sorte qu'un message puisse être transmis.	La fonction `return` arrête le programme, par conséquent, il n'est pas utile d'avoir une variable supplémentaire.
	Regrouper du texte et des variables pour les afficher dans un message texte.	L'utilisation de la fonction `return` limite les interactions avec l'utilisateur. Voir la partie « Pour mieux lire le code Python ».

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

171 pas

Avec Python

Le code complet est à construire par les élèves.

Ce code est composé d'une seule fonction :

- La fonction `test` de paramètres `min` et `max` qui sont respectivement le nombre de départ et le nombre maximal auquel finir.

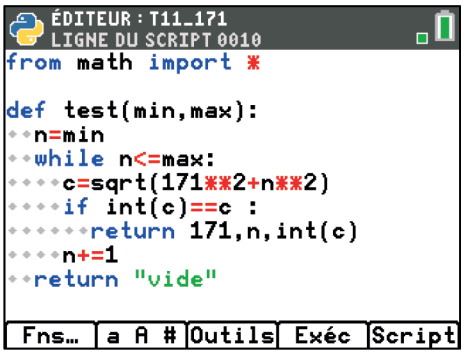
  Etant donné que le nombre de répétitions n'est pas connu, il faut utiliser une boucle `while`, qui nécessite une condition. Ici, la variable `n` doit rester inférieure ou égale au maximum pour que les instructions de la boucle se poursuivent.

  Il aurait été tout à fait possible d'utiliser une boucle `For`.

  Un test est réalisé pour chaque triplet potentiel pour savoir si la valeur de l'hypoténuse est bien un nombre entier.

  A noter l'utilisation de `int(c)` dans le `return` : voir la partie « Pour mieux lire le code Python ».

  Si aucun triplet n'a été trouvé dans l'étendue, alors le script renvoie le message `"vide"`.



Il faudra répéter plusieurs fois ce programme pour s'assurer d'obtenir toutes les valeurs d'emplacements des coffres à trésors, ce qui nécessite une coordination au sein des groupes et l'implémentation des programmes dans chaque calculatrice.

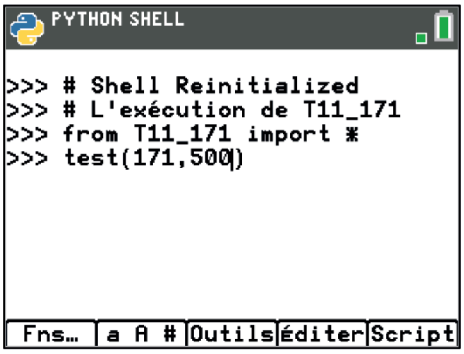
Une programmation possible est disponible sur le site TI : education.ti.com/fr/scratch-python

171 pas

Mode opératoire

Une fois le script exécuté, il faut appuyer sur la touche `[var]` : la fonction définie dans le script apparait.

Par les flèches directionnelles, il faut sélectionner la fonction `test`, valider par `Ok`, puis ajouter la valeur de départ du deuxième côté du triangle rectangle et la valeur finale à laquelle finir, en séparant les paramètres par une virgule.



Prolongements possibles

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

- programmer un test pour s'assurer que l'utilisateur a bien mis un nombre supérieur à 171 ;
- mettre les solutions trouvées dans une liste pour pouvoir être utilisée ensuite ;
- tester avec un nombre plus important de pas au départ comme 1 771.

Pour mieux lire le code Python

Il faut faire attention avec les boucles « répéter tant que » et « répéter jusqu'à ce que ». En effet, sans une condition d'arrêt, ces boucles peuvent créer une erreur en s'exécutant de façon infinie.

Il faudra appuyer sur la touche `[on]` pour arrêter le programme.

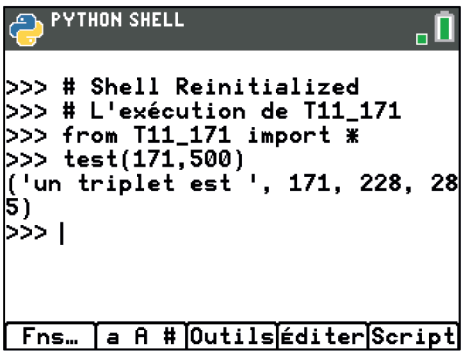
Il faut importer ici la bibliothèque `math` afin d'avoir la fonction racine carrée : `sqrt()`.

Concernant les interactions avec l'utilisateur, il convient d'indiquer que Scratch est un langage de programmation pour faire progresser la pensée algorithmique, très visuel et davantage poussé pour l'interaction avec un utilisateur. Le langage Python se veut plus général et permet de faire des mathématiques plus poussées, avec un réinvestissement des renvois d'une fonction pour une autre. C'est une raison à l'utilisation de la fonction `return` plutôt que la fonction `print`.

Il est cependant toujours possible d'écrire par exemple :

```
return "un triplet est ",171,n,int(c)
```

Le résultat est ci-contre.



Marche aléatoire

Résumé : une situation « pseudo réelle » est proposée sur un modèle de marche aléatoire : à quelle valeur peut-on estimer la probabilité de réussite de cette marche aléatoire ?

Mots-clés : bibliothèque `turtle` ; probabilités ; statistique ; marche aléatoire ; loi des grands nombres

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 en effectuant des simulations.

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

Situation déclenchante

Un cycliste débutant doit traverser un pont de 6 m de large et 10 m de long. On modélise son avancement de la manière suivante : à chaque fois qu'il avance d'un mètre, ce cycliste va vers la droite ou vers la gauche de manière aléatoire ; son écart (vers la droite ou vers la gauche) est égal à 1 mètre.

Au départ, le cycliste est situé au début du pont, à 3 m de chaque bord : quelles sont ses chances de réussite ?

Image libre de droits d'après Pixabay

Problématique

Comment estimer la probabilité de réussite de ce cycliste dans sa traversée du pont ?

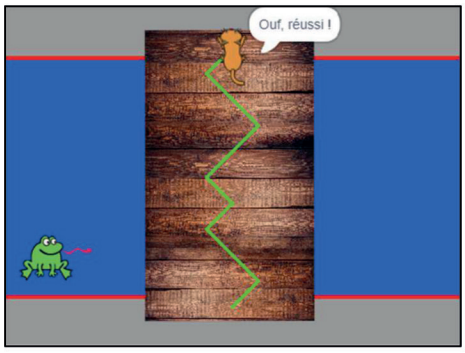
Marche aléatoire

Scénario pédagogique

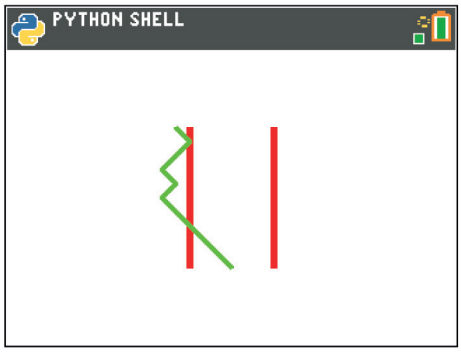
- **Avec la classe :** pour que chacun comprenne bien le sujet, il est intéressant de faire faire ce travail « à la main » à l'aide d'une pièce (pile ou face) sur une feuille de papier.
- **Approche algorithmique :** un travail par groupe peut être pertinent pour la mise en place d'un algorithme permettant de modéliser cette situation.
- **Utilisation d'un code :** un code peut être fourni à l'élève. La partie graphique sera laissée telle quelle et une partie du code, la fonction correspondant à la marche aléatoire en elle-même, peut être à la charge des élèves.
- **Mise en commun** (éventuellement au sein de petits groupes) : on observe que le passage est réussi ou pas à l'aide du graphique... mais est-ce suffisant ?
- **Approche fréquentiste :** mise en place en commun d'un algorithme permettant de donner la fréquence de réussite de la traversée du pont. Les fonctions en lien peuvent être fournies en partie aux élèves selon les choix de l'enseignant.
- **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

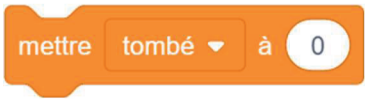
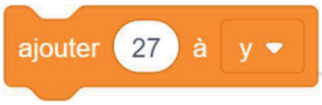


avec la TI-83 Premium CE Edition Python



Marche aléatoire

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
	Affecter la valeur de la variable « alea » avec un nombre aléatoire qui est soit 0, soit 1, l'instruction ne traitant que les nombres entiers.	`a=randint(0,1)`
	Affecter la valeur de la variable « tombé » à 0 ; cette variable contrôle si le personnage tombe ou le nombre de fois où le personnage tombe lors de plusieurs simulations.	Cette variable n'est pas utile en l'état dans Python du fait du renvoi d'un booléen True/False.
	Ajouter à la valeur de la variable « x » la quantité « -27 ».	`x-=10` Cette instruction comporte un opérateur d'affectation augmenté. Il est aussi possible d'écrire `x=x-10`. Les valeurs sont différentes entre Scratch et Python car les tailles d'écran sont différentes : voir « Pour mieux lire le code Python ».
	Ajouter à la valeur de la variable « y » la quantité « 27 ».	`y+=10`
	Déplacer le lutin aux coordonnées indiquées, ici $(x\,;y)$.	`turtle.goto(x,y)`
	Effectuer le quotient de la valeur de la variable « reussite » par celle de « n ».	`c/n`

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/

Marche aléatoire

Avec Python

Le code est fourni tout ou partie aux élèves, code dont voici quelques extraits.

Ce code est composé de plusieurs fonctions :

- La fonction `gt` de paramètres `x` et `y` qui permet au stylo d'aller directement au point de coordonnées $(x\,;y)$ sans écrire.
- La fonction `bord` de paramètres `x1` ; `y1` ; `x2`et `y2` qui trace des segments verts en trait épais joignant les points de coordonnées $(x1\,;y1)$ et $(x2\,;y2)$.
- La fonction `marche_aléatoire` (fonction sans paramètre) qui visualise une expérience de marche aléatoire.

  Cette fonction peut être fournie aux élèves en effaçant certaines lignes, celles qui concernent la condition par exemple.

  Il est important de bien maîtriser les syntaxes du type `x+=10` ; celle-ci signifie que la valeur de `x` augmente de 10.

  L'instruction `turtle.show()` permet de lancer la visualisation à la fin de l'appel à cette fonction.

- La fonction `reussi` : elle est sans paramètre et renvoie un booléen : `True` si la traversée du pont est faite, `False` en cas d'échec.

  On n'a pas de visualisation graphique avec cette fonction.

  La fonction `freq` ; elle a pour paramètre `n`, le nombre de répétitions d'essais de traversée de pont.

  Elle renvoie la fréquence de réussite.

  Elle est très simple dans sa syntaxe et utilise efficacement la fonction `reussi` précédemment décrite.

ÉDITEUR : T12_MAR1
LIGNE DU SCRIPT 0018

```
def bord(x1,y1,x2,y2):
  turtle.color(255,0,0)
  turtle.pensize(2)
  gt(x1,y1)
  turtle.goto(x2,y2)
  turtle.color(0,0,0)
  turtle.pensize(0)
  #turtle.show()
```

Fns… | a A # | Outils | Exéc | Script

ÉDITEUR : T12_MAR1
LIGNE DU SCRIPT 0029

```
from random import *
def marche_aléatoire():
  turtle.clear()
  bord(-30,-50,-30,50)
  bord(30,-50,30,50)
  gt(0,-50)
  x,y=0,-50
  turtle.color(0,255,0)
  turtle.pensize(1)
  for i in range(10):
    a=randint(0,1)
    if a==0:
      x+=10
    else:
      x-=10
    y+=10
    turtle.goto(x,y)
  turtle.show()
```

Fns… | a A # | Outils | Exéc | Script

ÉDITEUR : T12_MAR1
LIGNE DU SCRIPT 0039

```
def reussi():
  x=0
  for i in range(10):
    a=randint(0,1)
    if a==0:
      x+=1
    else:
      x-=1
    if x>3 or x<-3:
      return False
  return True
```

Fns… | a A # | Outils | Exéc | Script

ÉDITEUR : T12_MAR1
LIGNE DU SCRIPT 0060

```
def freq(n):
  c=0
  for i in range(n):
    if reussi():
      c+=1
  return c/n
```

Fns… | a A # | Outils | Exéc | Script

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

Marche aléatoire

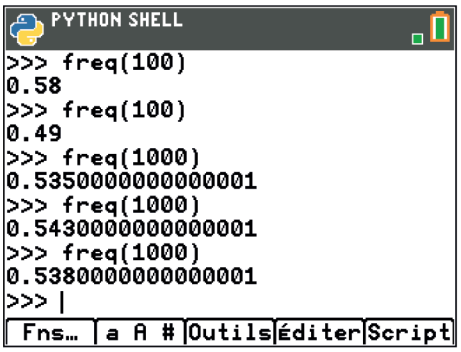
Mode opératoire

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 `marche_aléatoire` qui permettra de visualiser une situation. Il faut alors appuyer sur la touche [on] pour sortir de cette visualisation graphique.

En sélectionnant la fonction `reussi`, on verra s'afficher `True` ou `False` mais il est difficile de se faire une idée de la fréquence de réussite.

C'est la fonction `freq` qui permettra de le faire : on verra à ce moment-là que les valeurs peuvent différer pour une même valeur de n, et que ces valeurs de fréquence semblent se stabiliser à mesure que n augmente.



Prolongements possibles

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

- ajouter dans le graphique une phrase indiquant que la traversée est réussie ou non ;
- dans le graphique représentant la traversée, stopper le tracé dès que l'on bascule du pont ;
- modifier les valeurs numériques du problème de départ et adapter le code en conséquence.

Pour mieux lire le code Python

Il faut bien comprendre que dès que le script rencontre l'instruction `turtle.show`, l'écran propose une image : le dessin demandé par le script. Pour quitter cet écran, il faut appuyer sur la touche [on].

Cependant, il peut être intéressant de tester certaines fonctions graphiques, comme la fonction bord de ce code. Ainsi, on écrit `turtle.show()` à la fin de la fonction bord pour la tester et la valider et par la suite, on commente cette ligne par un `#,` en tapant sur la touche [2nde] suivie de [3] par exemple. Si cette fonction est appelée dans une autre fonction, les instructions se dérouleront sans se « stopper » à l'instruction `turtle.show()`. C'est ce qui a été fait dans le code proposé.

Pour information, sur Scratch, la taille d'écran est de 480 pixels par 360 pixels. Sur la calculatrice, elle est de 320 pixels par 240 pixels. Ces différences nécessitent donc des aménagements pour passer d'une programmation à l'autre, occasionnant des problèmes mathématiques qu'il peut être intéressant à développer pour les élèves les plus en avance.

Bien avoir en tête les coordonnées des extrémités de l'écran de la calculatrice : $(-160\,;115)$ pour le coin supérieur gauche et $(160\,;-115)$ pour le coin inférieur droit.

Multiples

Résumé : dans cette activité, les élèves doivent programmer afin d'obtenir les multiples d'un nombre dans une liste.

Mots-clés : multiples ; fractions ; liste ; boucles itératives

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.

Calculer : « effectuer un calcul automatisable à la main ou à l'aide d'un instrument (calculatrice, logiciel) » ; la recherche systématique d'un dénominateur commun passe par des calculs automatisés.

Situation déclenchante

Les élèves ont des difficultés sur le calcul numérique, en particulier sur les fractions. La somme ou différence de deux fractions de dénominateurs différents impliquent de connaître les multiples de ceux-ci afin de pouvoir faire le calcul demandé. Certains élèves n'ont pas acquis l'optimisation de recherche du plus petit multiple commun (ppcm ou ppmc).

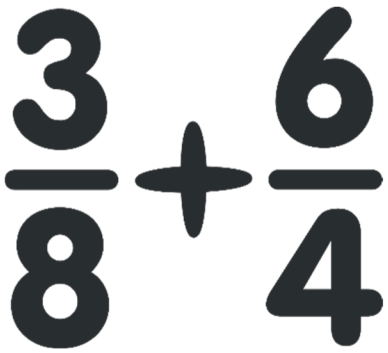


Image libre de droits d'après Pixabay

Problématique

Comment obtenir, par programmation, le ppcm ou ppmc de deux nombres entiers donnés ?

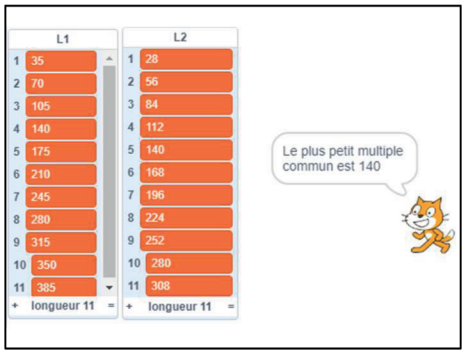
Multiples

Scénario pédagogique

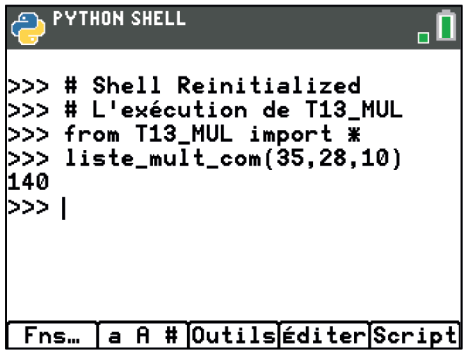
- **Avec la classe :** à l'occasion d'un exercice ou d'une correction de devoir sur le calcul fractionnaire, mettre au défi les élèves de programmer une aide pour la somme ou différence de fractions.
- **En groupe de 2 à 4 élèves :** les élèves doivent faire une ébauche d'algorithme permettant la résolution du problème proposé, avant de passer à l'implantation dans un langage ou un autre, après validation du professeur. Cette partie peut nécessiter un temps assez long pour que les élèves s'approprient le script, notamment par la maîtrise de fonctions dédiées : les listes par exemple, et notamment le parcours des éléments d'une liste dans une boucle `for` ou une structure conditionnelle `if`.
- **Mise en commun :** les élèves expliquent leurs démarches.
- **Preuve :** les élèves peuvent montrer qu'il existe toujours au moins un dénominateur commun.
- **Pour les élèves les plus en avance :** il est possible de leur proposer un ou plusieurs prolongements possibles, décrit en fin de fiche.
- **Difficultés rencontrées :**
    - la gestion des listes.

Voici les visuels à l'issue des programmes :

en Scratch

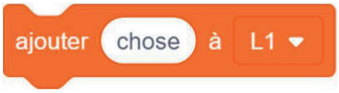


avec la TI-83 Premium CE Edition Python

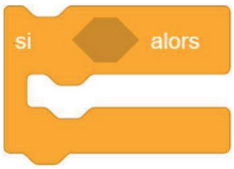


Multiples

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
	Ajouter un élément à une liste préalablement créée.	`L1.append(élément)` *Append* signifie « ajouter » en anglais. « `élément` » est donc ajouté à la suite des autres éléments dans la liste `L1`.

	Créer une boucle répéter.	`for i in range n:`
		`••instructions`
		
		Par défaut, les valeurs de `i` commencent à 0 et augmentent avec un pas de 1.
		L'indentation avec •• est nécessaire pour définir les instructions à exécuter dans un bloc du code Python.

	Créer le i-ième multiple du premier élément de la liste L1	`i*n` `n` doit être défini comme le premier élément de la liste.
	Créer une structure de contrôle conditionnelle.	`if condition:` `••instructions`

	Tester si l'élément i de la liste L1 est égal ou non à l'élément j de la liste L2.	`for élément in L1:`
		`••if élément in L2:`
		`••••instructions`
		
		Cette programmation est expliquée dans la partie suivante « Avec Python ».

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

Multiples

Avec Python

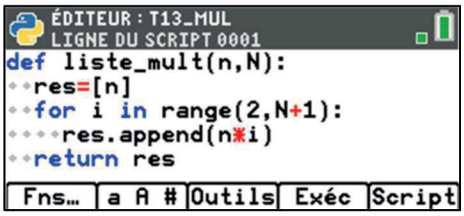
L'un des codes principaux est à construire complétement par les élèves.

Ce code est composé de plusieurs fonctions :

- La fonction `liste_mult` de paramètres `n` et `N` qui sont respectivement le nombre dont il faut trouver les multiples et l'étendue.

  C'est une fonction qui sera utilisée par les scripts principaux pour créer la liste des N multiples du nombre n.

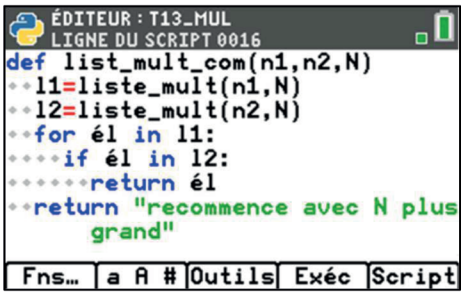
  Il faut donc commencer la boucle `for` à 2 pour la finir à `N+1`.



- La fonction `list_mult_com` de paramètres `n1` ; `n2` et `N` qui sont respectivement les deux nombres pour lesquels il faut déterminer les multiples et l'étendue.

  A noter la programmation des boucles : pour chaque élément de la liste `l1`, celui-ci est comparé à chacun des éléments de la liste `l2`.

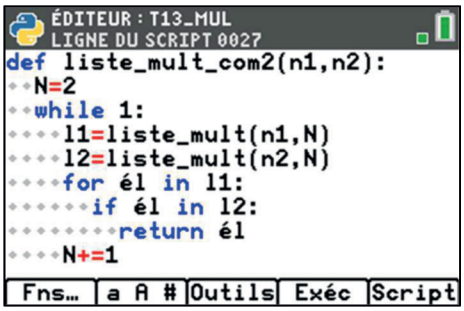
  On trouvera le `é` dans l'onglet `a A #`.



- La fonction `liste_mult_com2` de paramètres les nombres `n1` et `n2`.

  Cette fonction permet de trouver le plus petit multiple commun en recalculant la liste des multiples, en augmentant l'étendue de 1 à chaque itération de la boucle `while`.

  A noter que la boucle `while` se poursuit indéfiniment car la condition est toujours vraie. L'arrêt se produit lorsque le script renvoie un multiple commun, ce qui sera toujours le cas, au moins pour $n1 \times n2$.



- La fonction `ppmc` de paramètres les nombres `n1` et `n2`. Elle est laissée à l'appréciation du lecteur.

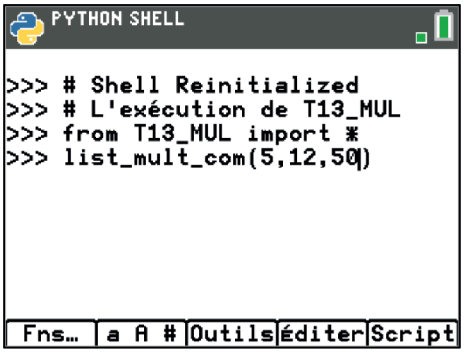
Une programmation possible est disponible sur le site TI : education.ti.com/fr/scratch-python

Multiples

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 `list_mult_com`, valider par `Ok`, puis ajouter les deux nombres pour lesquels on souhaite avoir les multiples, et l'étendue, en séparant les paramètres par une virgule.



Prolongements possibles

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

- poursuivre la programmation en cherchant quel est le plus petit élément parmi les éléments finaux des listes `l1` et `l2` afin de n'augmenter que la liste qui a ce plus petit élément ;
- écrire un script prenant aussi les numérateurs des fractions afin de renvoyer les fractions avec les représentants de même dénominateurs.

Crible d'Eratosthène

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 ?

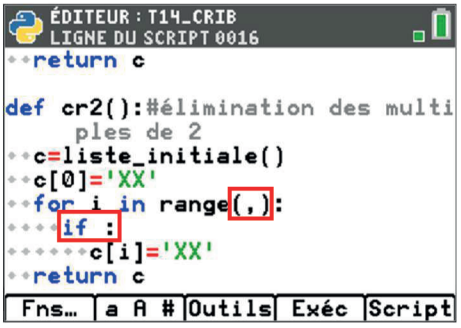
Crible d'Eratosthène

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.

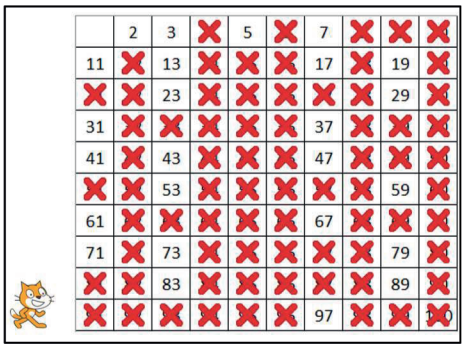
Il est possible que l'exécution du script ne tienne pas compte d'une ligne en la commentant par # qui s'obtient, par exemple, par la touche `[2nde]` suivie de la touche `[(-)]`.

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.

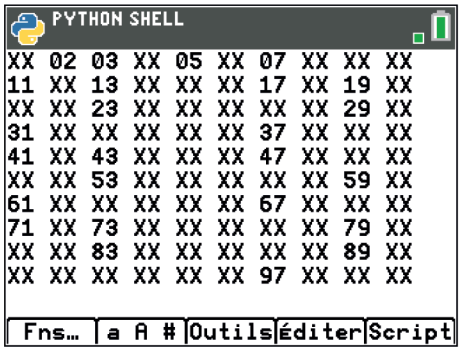
- **Pour les élèves les plus en avance :** il est possible de leur proposer un ou plusieurs prolongements possibles, décrit en fin de fiche.

Voici les visuels à l'issue des programmes :

en Scratch

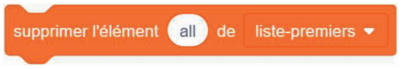
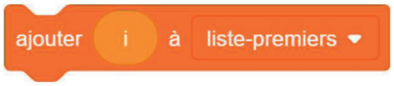
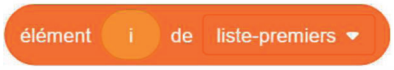
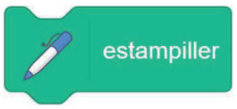


avec la TI-83 Premium CE Edition Python



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/

Crible d'Eratosthène

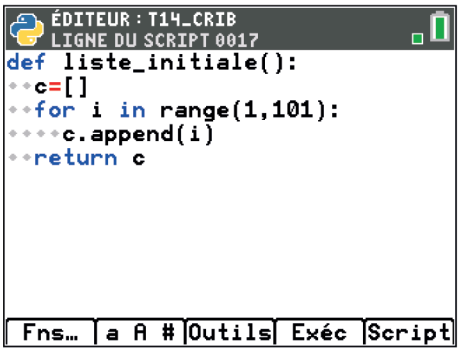
Avec Python

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.

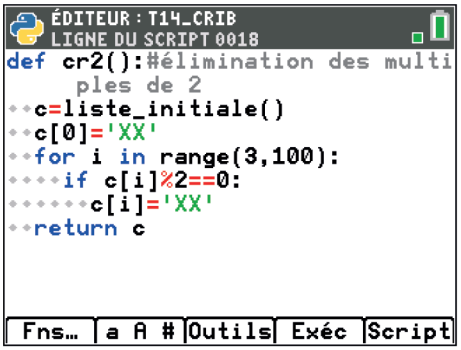
  Bien noter que l'instruction `range(1,101)` permet d'aller de 1 (inclus) à 101 (exclu), donc jusqu'à 100.



- 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.

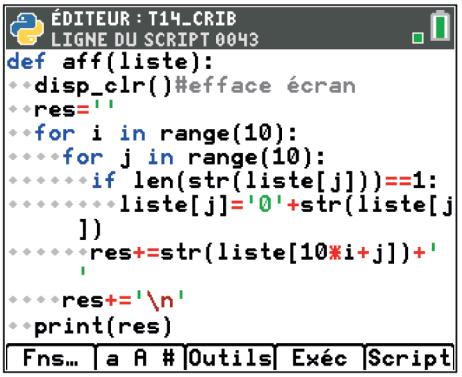
  Il est efficace d'utiliser les copier/coller : on se place sur la ligne à copier puis `Outils` par appui sur la touche [zoom] et enfin [5] ; on se place ensuite sur la ligne où l'on veut effectuer cette copie puis `Outils` et [7].

- 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.

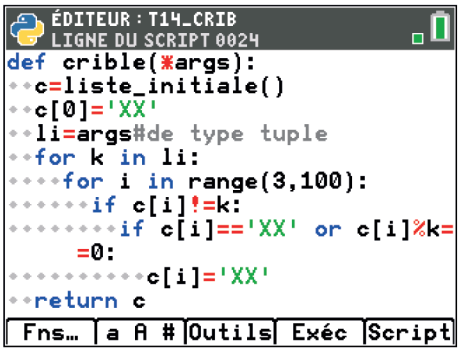
  La syntaxe `'\n'` permet un retour à la ligne.

  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

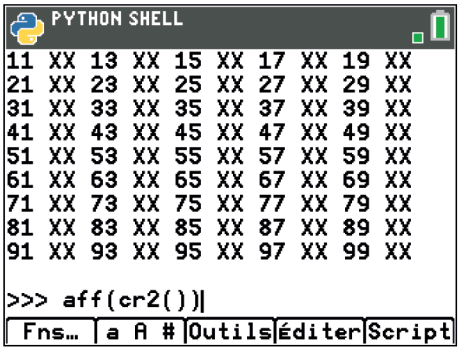
Mode opératoire

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.