https://ww2.ac-poitiers.fr/math/spip.php?article1069 - Auteur : Raphaël Nivelle

Algorithmique en seconde : aborder la notion de fonction

publié le 27/12/2019 - mis à jour le 03/01/2020

De Scratch vers Python

Descriptif :

Cet article présente une séquence permettant l'introduction du langage Python en seconde, à partir des connaissances des élèves en Scratch. Elle s'appuie sur la notion de distance entre deux réels

Sommaire :

Définition mathématique de la distance entre deux réels et conversion en algorithme

La notion de distance entre deux réels figure au programme de seconde depuis la rentrée 2019. Sa définition pratique présente un cas intéressant de disjonction de cas :

Définition :

La distance entre deux réels a et b, notée $d(a;b)$, est la différence entre le plus grand de ces deux réels et le plus petit. La valeur de cette distance dépend donc de **l'ordre** de a et de b :

- si $a > b$ alors $d(a;b) = a - b$;
- si $a \leqslant b$ alors $d(a;b) = b - a$;

Sans changer la définition mathématique, on peut remplacer les deux conditions par une instruction conditionnelle étendue de la forme Si ... alors : .... Sinon : .... On obtient alors une version algorithmique de la distance, avec entrée et sortie :

Algorithme en pseudo-langage

Algorithme du calcul de la distance

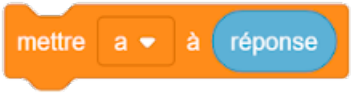
```
Variables
    a (réel), b (réel)
Début
1   | a ← Saisir("Valeur de a ? ")
2   | b ← Saisir("Valeur de b ? ")
3   | Si a > b alors :
4   |     | d ← a − b
5   | Sinon :
6   |     | d ← b − a
7   | Fin Si
8   | Afficher("La distance vaut : ", d)
9 Fin
```

Une fois que cette forme a été établie en classe, les élèves ont été invités à convertir cet algorithme en un programme Scratch. Étant en début d'année, c'était l'occasion de mesurer les compétences acquises à l'issue du cycle 4, en terme de programmation et de manipulation du logiciel Scratch.

La plupart d'entre eux a réussi à convertir correctement la structure conditionnelle en un bloc Scratch adéquat mais les difficultés ont été plus nombreuses en ce qui concerne la gestion des **variables** :

- statut particulier de la variable 

: cette variable étant associée au bloc capteur *demander et attendre*


: elle stocke temporairement la réponse de l'utilisateur après l'utilisation d'un tel bloc. Dans le cas d'utilisations successives de ces blocs, les réponses sont écrasées et c'est la dernière valeur saisie qui est conservée.
- nécessité de créer des variables `a`, `b`, `d` pour stocker les valeurs saisies par l'utilisateur et les réutiliser pour le calcul de la distance, laquelle doit aussi être enregistrée dans une variable.
- certains élèves ont été gênés pour retrouver l'affectation de variable : l'affectation en langage naturel s'écrit dans le sens *variable reçoit valeur* (`a <- réponse`) alors que le bloc d'affectation de Scratch et son verbe *mettre* peut faire penser à une affectation dans l'autre sens (mettre une valeur dans une variable) :


Une majorité d'entre eux a tout de même réussi à obtenir un script ressemblant à celui-ci :

Image d'un script Scratch implémentant le calcul de la distance entre deux réels

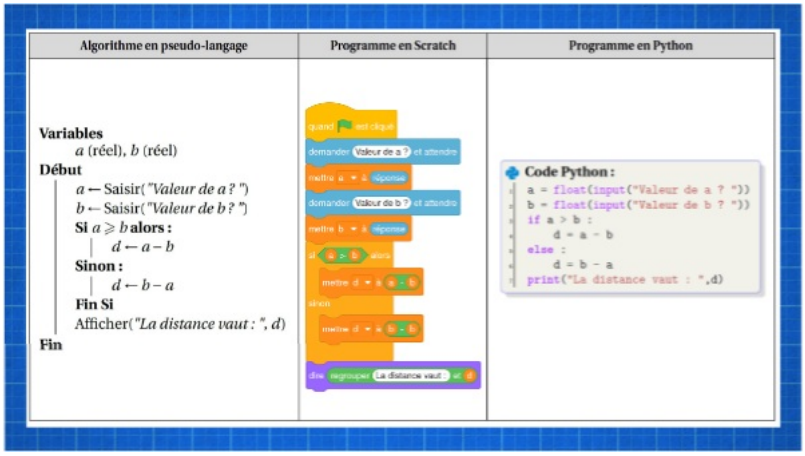
Une version en ligne peut être testée directement ci-dessous :

Exemple de script scratch pour le calcul de la distance de deux réels, avec entrée et sortie (Scratch)

Intégration d'un script Scratch pour le calcul de la distance de deux réels.

● Le saut vers Python

La suite de l'activité a consisté à traduire le script Scratch en script Python. Étant totalement novices dans ce langage, les élèves ont commencé par recopier le script proposé afin de se familiariser avec les premiers éléments de ce langage et repérer les premières analogies avec le langage naturel et le langage Scratch :



L'algorithme de la distance en langage naturel et son implémentation en Python et en Scratch (Genially)
Diaporama illustrant la traduction de l'algorithme de la distance en Python et en Scratch

Les difficultés rencontrées lors de la saisie tiennent aux spécificités du langage Python :

- utilisation des guillemets pour les chaînes de caractères ;
- en-tête des blocs se terminant par les deux points " :" ;
- **indentation** des instructions à l'intérieur d'un bloc ;
- absence de symbole de fin de bloc autre que le retour à l'indentation initiale

Lors de cette simple séquence de recopie, les élèves ont pu appréhender les exigences d'un langage informatique textuel et se confronter à la précision et la rigidité d'une syntaxe proche de celle des expressions mathématiques.

Une version en ligne peut être testée directement ci-dessous :

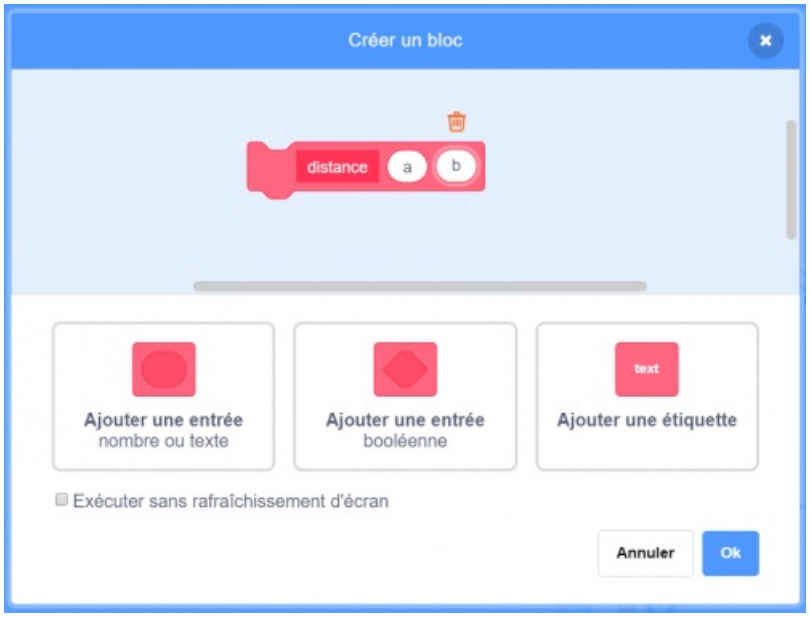
Exemple de script Python pour le calcul de la distance de deux réels, avec entrée et sortie (Genially)
Intégration d'un script Python pour le calcul de la distance de deux réels, avec entrée et sortie, à partir du site replit

● Approche fonctionnelle : le bloc de fonction de Scratch et la fonction Python

Les notions d'entrées-sorties (fonctions demander et attendre-dire en Scratch et input-print en Python) évoquées plus haut ne relèvent pas de la pensée algorithmique et l'accent mis par le programme sur la notion de **fonction** permet de s'en libérer complètement (cf document d'accompagnement "Algorithmique et programmation" (mai 2017).

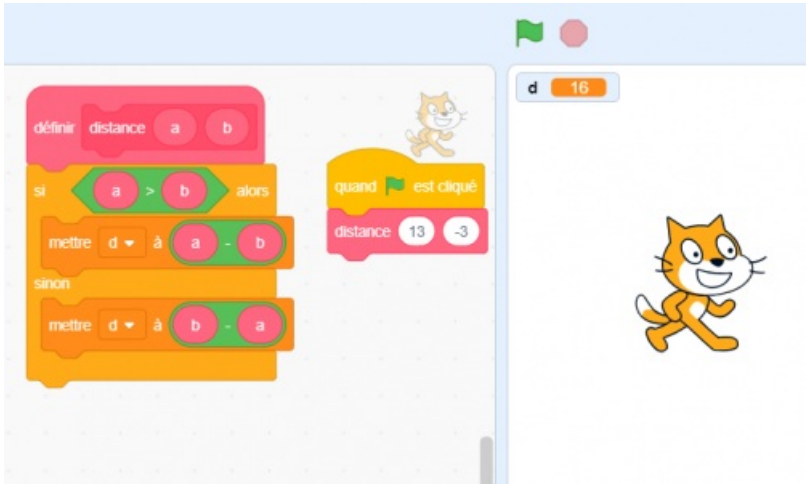
Les blocs de fonctions de Scratch permettent de construire des blocs personnalisés d'instructions, avec paramètres

d’entrée :



Copie d’écran illustrant la création d’un bloc personnalisé sous Scratch

Ainsi, les élèves ont pu construire un bloc de fonction `distance` qui fait le calcul de la distance et stocke ce calcul dans la variable `d` :



Copie d’écran du bloc de fonction distance avec Scracth

L’appel de la fonction est réalisé par l’insertion d’un bloc sous un bloc chapeau et les paramètres d’entrée sont spécifiés directement dans le script avant le clic sur le drapeau vert.

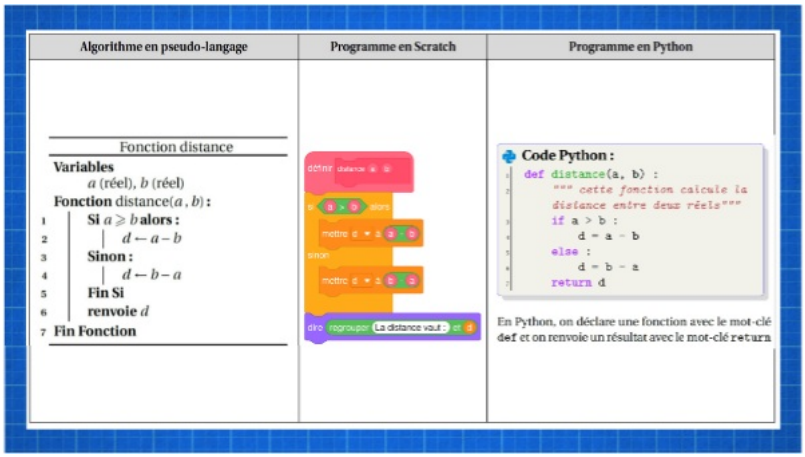
Avec Python, ils ont pu mesurer l’analogie de la construction avec l’en-tête `def distance(a, b):` qui joue le même rôle que le bloc 

et le bloc d’instructions quasiment identique à celui utilisé avec entrées-sorties :

Correspondance entre code Scratch et code Python pour la fonction distance (Genially)
Diaporama Genially illustrant la correspondance entre les deux langages

Toutefois, l'analogie entre ces deux constructions a ses limites :

- dans les deux cas, les variables `a` et `b` sont des variables locales, internes à la fonction mais la variable `d` est une variable globale en Scratch alors qu'elle reste interne à la fonction en Python. La possibilité de restreindre la variable à un lutin ne l'empêche pas d'être modifiée au cours d'un script extérieur à la fonction, ce qui peut poser un problème d'intégrité de la variable.
- En Scratch, il n'existe pas de mot clé `return` qui permet de renvoyer la valeur de la fonction. Celle-ci est stockée dans la variable globale `d` avec les risques évoqués plus haut : elle n'est pas "protégée" comme le serait le résultat d'un appel de fonction avec Python.



La fonction distance en langage naturel et son implémentation en Python et en Scratch (Genially)
Animation illustrant la traduction de la fonction distance en Python et en Scratch

● Réutilisation d'une fonction : centre et rayon d'un intervalle

Un des intérêts des fonctions en Python est la possibilité d'utiliser la valeur qu'elles renvoient (quand c'est prévu) dans un calcul ou une autre fonction du même script comme les $f(x)$ en mathématiques. De la même manière que l'on peut calculer : $2f(3)+5$ en mathématiques, on peut aussi calculer : `2*distance(a,b)+5` en Python, puisque la fonction `distance` renvoie un nombre.

La suite de la séquence consistait donc à demander aux élèves de construire eux-mêmes une fonction `centre_rayon(a,b)`, à partir du schéma ci-dessous :

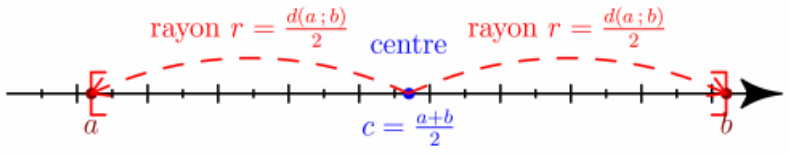
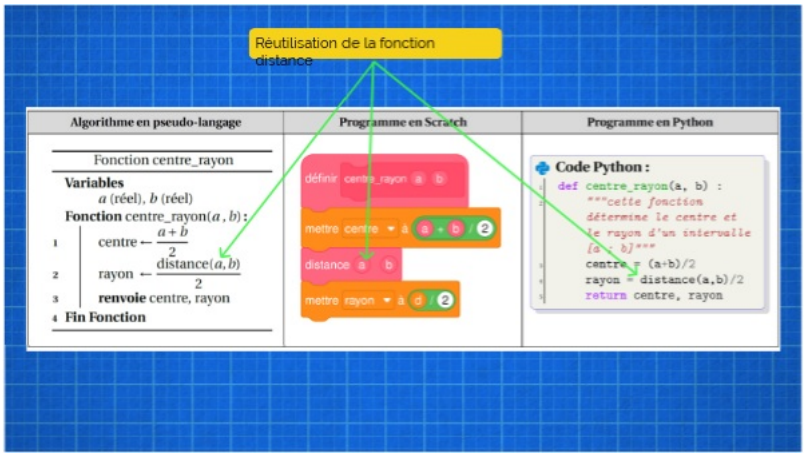


Schéma illustrant la notion de centre et de rayon d'un intervalle [a ; b]

La difficulté portait davantage sur leur capacité à réinvestir les structures que sur les contenus mathématiques. L'appel de la fonction distance à l'intérieur de la fonction centre_rayon a été mis en avant dans les consignes.



Fonction centre_rayon utilisant la fonction distance (Genially)
Diaporama illustrant la traduction de la fonction centre_rayon (utilisant la fonction distance) en Scratch et en Python

La fin de l'activité demandait aux élèves de déterminer des fonctions permettant de résoudre des équations de la forme $|x-a|=r$ et des inéquations de la forme $|x-a|[\leqslant?]r$. Là encore, la dimension algorithmique/programmation primait sur les mathématiques et il s'agissait surtout de réinvestir la notion de fonction.

- Fiche élève (PDF de 251.2 ko)
  Fiche élève récapitulative de la séance sur la distance entre deux réels
- Script Python de l'activité sur la distance (Zip de 435 octets)
  Script Python des fonctions construites au cours de l'activité sur la distance
- Sources de la fiche élève (Zip de 402.8 ko)
  Archive contenant les sources tex, les figures et les scripts informatiques utilisés dans la fiche élève et pendant l'activité