Introduction à Scratch et à l'algorithmique

Formation préparatoire au CRPE - INSPE Lyon

Valentin ROUSSEL

Laboratoire S2HEP, Université de Lyon, France

Résumé du cours - L'algorithmique est l'étude et la production de règles et techniques impliquées dans la conception d'algorithmes. Un algorithme est un ensemble de règles et d'opérations successives qui permet de résoudre un problème énoncé. Il peut être traduit, grâce à un langage de programmation, en un programme exécutable par un ordinateur. L'algorithmique est donc l'étude de la production d'algorithmes en vue de répondre à des problèmes. Ce cours propose une brève introduction à l'algorithmique, par la production et l'étude de courts programmes basiques en scratch. Scratch un langage de programmation exécutable par le logiciel de même nom. L'étude de ces programmes sera l'occasion d'aborder les notions de *variable*, de *boucle*, de *condition* et d'*interaction homme-machine* au travers de 2 types d'exercices : l'un portant sur les programmes de calcul, l'autre sur la construction de figures.

Les exercices de niveau * correspondent au niveau attendu pour l'épreuve du CRPE**. Les exercices de niveaux supérieurs correspondent à un niveau plus avancé, garantissant toutefois une maîtrise amplement suffisante de scratch pour la préparation au concours.

1 Programmes de calcul

En scratch, un programme est une suite de blocs. Chaque bloc réalise une et unique tâche. Le programme principal commence toujours par le bloc de départ :

les blocs de type :

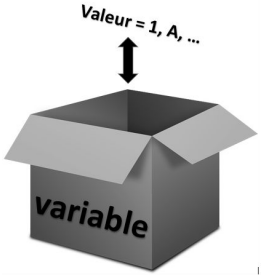
- **mouvement**, permettent de déplacer et orienter le lutin dans la scène.
- **apparence**, permettent de modifier l'aspect du lutin et le faire parler.
- **son**, permettent de jouer des sons et gérer le volume.
- **événement**, gèrent les interactions entre l'homme et la machine. Le bloc de départ permet par exemple à l'utilisateur d'indiquer à la machine d'exécuter le programme scratch.
- **contrôle**, permettent de répéter ou conditionner les actions des blocs qui sont à l'intérieur.
- **capteur**, gèrent également les interactions entre l'homme et la machine mais ont en plus la capacité de garder une information en mémoire.
- **opérateur**, gèrent les opérations classiques : addition, multiplication, soustraction division, comparaisons logiques etc...
- **variable**, permettent de déclarer des variables et de gérer le nom et la valeur de ces variables.

1.1 Variable

Définition - Une variable possède :

- Un nom
- Une valeur

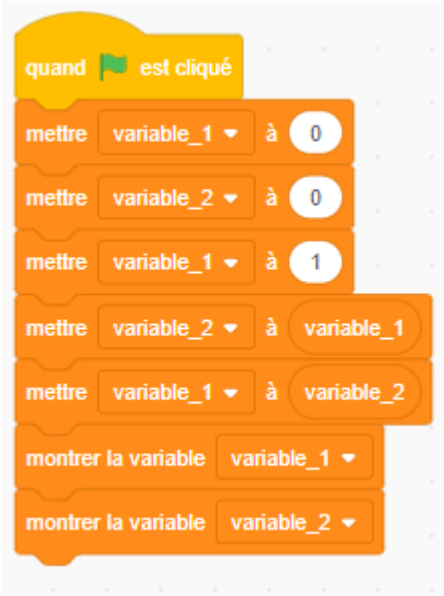
On peut représenter une variable comme une étiquette collée sur une boîte qui contient une valeur qui peut changer au fil du temps.



Exercice 1.1*
A chaque étape (6) du programme ci-dessous, indiquer la valeur de la variable « ma variable ».

Exercice 1.2*
A chaque étape (7) du programme ci-dessous, indiquer la valeur de la variable « score ».

Exercice 1.3*
A chaque étape (8) du programme ci-dessous, indiquer les valeurs des variables « $variable_1$ » et « $variable_2$ » .



1.2 Condition

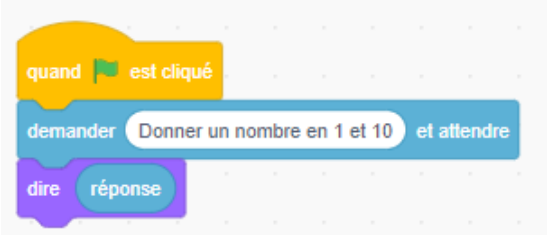
Définition - Une instruction conditionnelle permet d'effectuer différents calculs ou actions en fonction de l'évaluation d'une *condition*.

Exercice 1.4**
En fin d'exécution, quel résultat retournera ce programme ?

1.3 IHM

Définition - Les IHM sont les moyens mis en œuvre afin qu'un utilisateur puisse contrôler et communiquer avec une machine.

Exercice 1.5*
Dans l'interface de Scratch, reproduire le script suivant et décrire ce que fait le programme.



Exercice 1.6**
Décrire, étape par étape ce que fait le programme suivant.

Exercice 1.7**
Ecrire un programme qui :

1. Demande à l'utilisateur un nombre
2. Multiplie ce nombre par lui-même
3. Soustrait 1 au résultat
4. Affiche le résultat

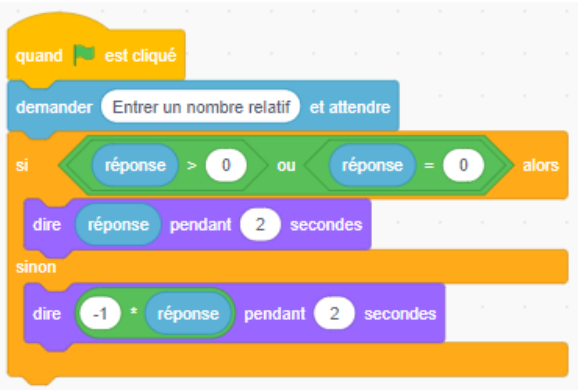
Exercice 1.8***
Ecrire un programme qui :

1. Demande à l'utilisateur un nombre
2. Ajoute 1 à ce nombre
3. Multiplie le résultat par le nombre de départ
4. Soustrait le nombre de départ au résultat
5. Soustrait 1 au résultat
6. Affiche le résultat

Exercice 1.9***
Ecrire un programme qui :

1. Demande à l'utilisateur un nombre
2. Ajoute 1 à ce nombre
3. Multiplie le résultat par lui-même
4. Soustrait au résultat le carré du nombre de départ
5. Soustrait 2 au résultat
6. Affiche le résultat

Exercice 1.10***
Décrire le programme suivant

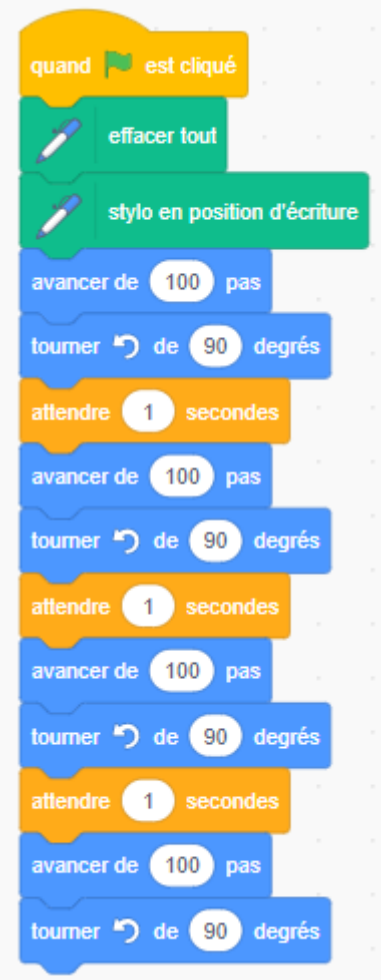


Exercice 1.11***
Décrire le programme suivant

2 Construction de figures

2.1 L'outil stylo

Exercice 2.1*
Dans l'interface de Scratch, reproduire le script suivant et décrire ce que fait le programme.



Exercice 2.2 ** (Début) à * (fin)**
Que fait ce programme ? Que remarque-t-on ? Comment optimiser (réécrire) ce programme ?

2.2 Les blocs

Définition - Les blocs sont des sous-programmes, déclarés en dehors du programme principal et pouvant être appelés dans le programme principal.

Exercice 2.3 ***
Dans l'interface de Scratch, reproduire le script suivant et décrire ce que fait le programme.

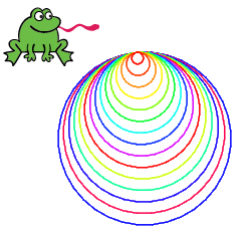
Exercice 2.4 **
Que fait le programme suivant... ?

3 Préparation au CRPE - Niveau avancé

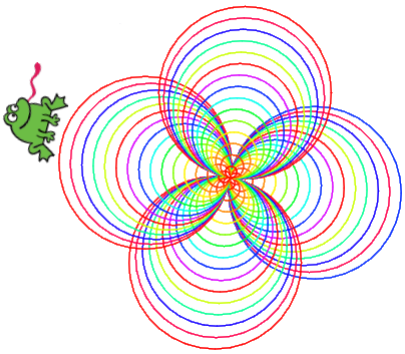
Les exercices suivants sont proposés par assurer une préparation optimale aux épreuves du CRPE. Ils mobilisent les connaissances acquises dans les exercices précédents, mais nécessitent un degré de complexité algorithmique supplémentaire.

Exercice 3.1 ****
Ecrire un programme qui permet de tracer la figure suivante :

Exercice 3.2 ****
Ecrire un programme qui permet de tracer la figure suivante :



Exercice 3.3 *****
Ecrire un programme qui permet de tracer la figure suivante :



Exercice 3.4 *****
Voir dernière page après les corrections

4 Corrections

Exercice 1.1

1. ma variable= 5
2. ma variabl= 10
3. ma variabl= 15
4. ma variabl= 20
5. ma variabl= 25

Exercice 1.2

1. $score = 5$
2. $score = 10$
3. $score = 0$
4. $score = 2$
5. $score = 7$
6. $score = 7$

Exercice 1.3

1. $variable_1 = 0$
2. $variable_2 = 0$
3. $variable_1 = 1$
4. $variable_2 = variable_1 = 1$
5. $variable_1 = variable_2 = 1$
6. $variable_1 = 1$
7. $variable_2 = 1$

Exercice 1.4

"Variable 1 \= Variable 2" car variable 1 = 1 et variable 2=2

Exercice 1.5

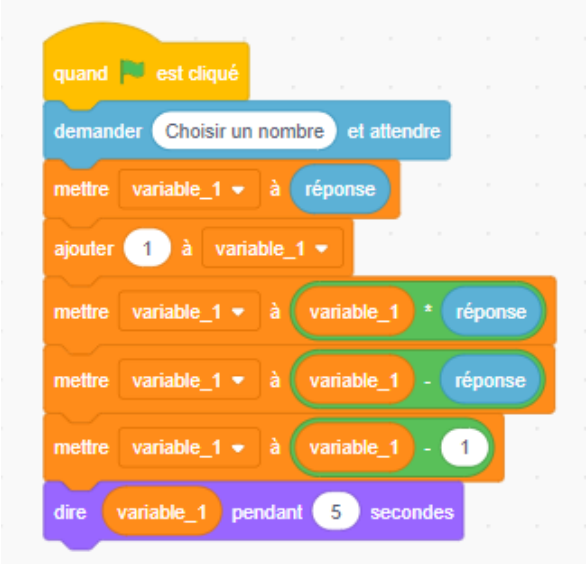
Le programme demande un nombre à l'utilisateur. A la fin du programme, le lutin dit le nombre que l'utilisateur a choisi.

Exercice 1.6

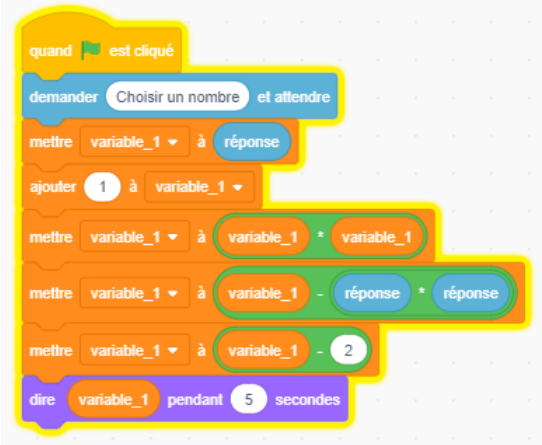
1. Le lutin annonce ce que va faire le programme
2. Le programme demande un nombre à l'utilisateur
3. Le programme déclare une variable "ma variable" et lui donne pour valeur 0.
4. Le programme donne à la variable "ma variable" la valeur "réponse + réponse", qui correspond au double du nombre donné par l'utilisateur.
5. Le programme affiche la valeur de la variable "ma variable", ce qui correspond au résultat.

Exercice 1.7

Exercice 1.8



Exercice 1.9



Exercice 1.10

1. Le programme demande à l'utilisateur un nombre relatif.
2. Si l'utilisateur donne un nombre supérieur ou égal à 0 le programme affiche ce nombre.
3. Sinon, si le nombre et inférieur à 0 le programme transforme ce nombre en un nombre positif et l'affiche

Ainsi, ce programme retourne la valeur absolue d'un nombre.

Exercice 1.11

1. Le programme demande à l'utilisateur un nombre entre 10 et 20.
2. Le programme déclare une variable $variable_1$ et lui donne pour valeur le nombre indiqué par l'utilisateur.
3. Le programme déclare une variable $variable_2$ et lui donne pour valeur le nombre indiqué par l'utilisateur.
4. Jusqu'à ce la $variable_1$ atteigne 0, le programme répète :
   - Ajoute - 1 à $variable_1$ (il décrémente la valeur de la variable).
   - Double la valeur de $variable_2$.
5. Affiche la valeur définitive de $variable_2$.

Ainsi, ce programme retourne la valeur absolue d'un nombre.

Exercice 2.1

Ce programme trace un carré de côté 100 pas.

Exercice 2.2

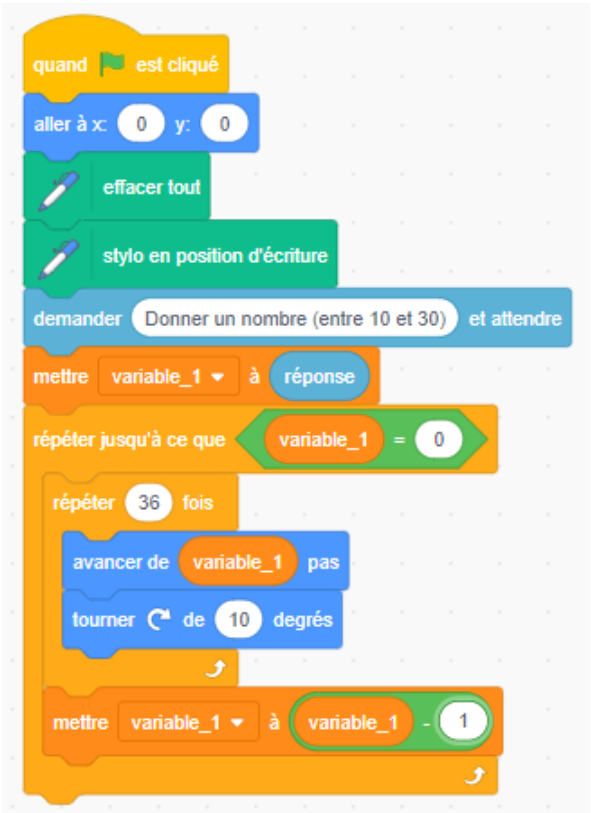
Ce programme trace une croix suisse. On remarque que des séries de blocs se répètent. Pour optimiser le programme, on peut le réécrire à l'aide de blocs de contrôle.

Exemple de solutions utilisant la décrémentation :

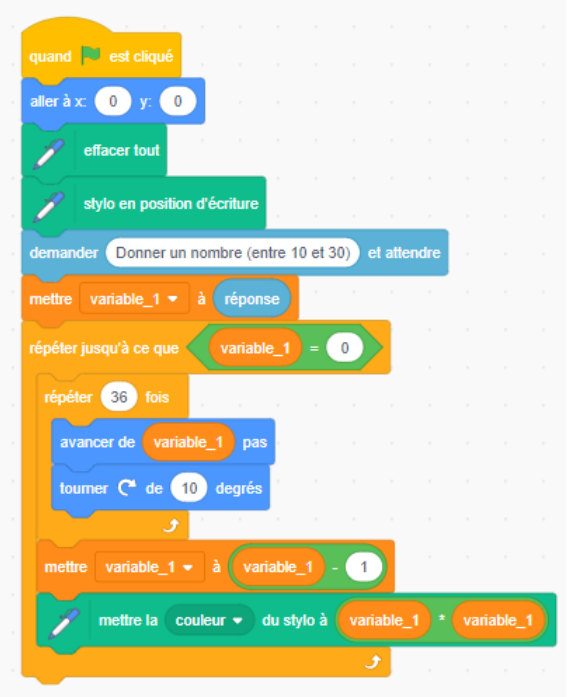
Exercice 2.3
Le programme appelle un sous programme "triangle". Le sous programme "triangle" trace un triangle équilatéral de 50 pas de côté. Le programme principal trace trois triangles alignés dans un ordre particulier : au centre, à droite puis à gauche.

Exercice 2.4
Ce programme trace un cercle.

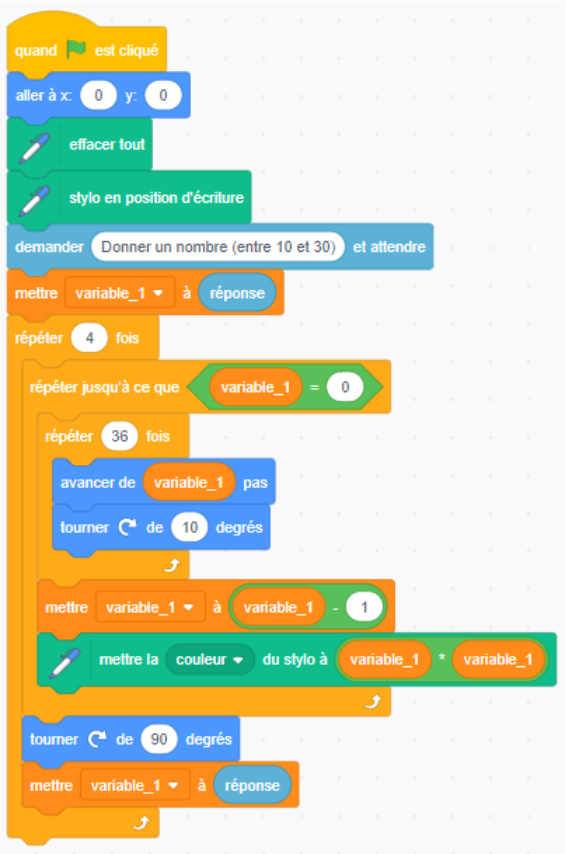
Exercice 3.1



Exercice 3.2



Exercice 3.3



Exercice 3.4 *** (avec correction)**
Ecrire un programme qui :
— Demande à l'utilisateur où déplacer le lutin.
— Une fois à l'emplacement, demande à l'utilisateur s'il souhaite tracer un cercle, un triangle équilatéral, ou un carré, et retourne un message d'erreur si l'utilisateur n'a pas indiqué une réponse correcte.
— Une fois que l'utilisateur a indiqué son choix, demande à l'utilisateur la dimension (en pas), de la figure souhaitée.
— Une fois la figure tracée, recommence à l'étape 1. Le programme devra s'arrêter lorsque la touche « espace » est maintenue.
3 blocs doivent préalablement être déclarés :
— Le bloc « triangle_ équilatéral_ générique », qui demande à l'utilisateur la longueur d'un côté du triangle et trace le triangle.
— Le bloc « carré_ générique », qui demande à l'utilisateur la longueur d'un côté du carré et trace le carré.
— Le bloc « cercle_ générique », qui demande à l'utilisateur la dimension en pas d'un cercle et trace le cercle.