L'inscription et l'utilisation sont gratuites. Il suffit pour cela d'utiliser un ordinateur disposant de flash (Scratch ne fonctionne pas sur les tablettes ni sur les mobiles).

1) Rendez-vous sur le site suivant pour accéder à l'outil SCRATCH : https://scratch.mit.edu
2) Cliquez sur « commencer à créer »
3) Découvrez l'outil en suivant les tutoriels de prise en main , dans la rubrique en haut « tutoriels »

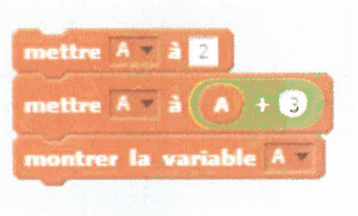
Un algorithme est composé de 3 grandes parties :

1. **Les informations de départ**
2. **Les instructions**
3. **La réponse**

Exemple :

Dans l'algorithme ci-contre, nous avons :

- *L'information de départ*
- *Les instructions*
- *La réponse finale*



Programmation

Dans le langage de programmation, les 3 grandes parties de l'algorithme sont traduites de cette façon :

Informations de départ ⇔ Entrée ou lecture

Instructions ⇔ Traitement

Réponse ⇔ Sortie ou écriture

Par exemple :

Dans sa recette de cake pour 10 personnes, Paola doit utiliser 5 œufs. Mais ce soir, elle ne reçoit que 7 personnes (ils seront 8 à table). Combien d'œufs doit-elle utiliser alors ?

Il faut faire une règle de 3 :

5 œufs pour 10

? œufs pour 8

$$? = \frac{5 \times 8}{10}$$

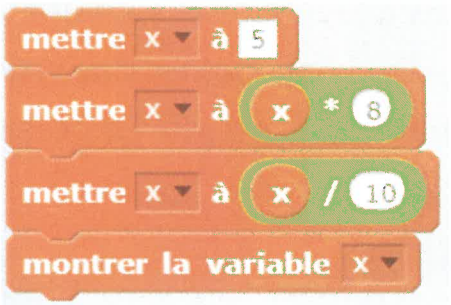
Nous pouvons utiliser un algorithme pour résoudre ce calcul

Attribuer à x la valeur de 5

Diviser par 10

Multiplier par 8

Afficher le résultat



Notez que l'algorithme fonctionnera pour chaque ingrédient, passant les doses de 10 personnes à 8 personnes, ce qui évitera à Paola de faire ses calculs pour sa farine, son sucre et son beurre. **Elle gagne du temps.**

L'ensemble des instructions forme un **script**. En langage de programmation, un ordinateur n'est capable de comprendre que quatre instructions :

- La lecture / écriture
- L'affectation de variables
- Les tests
- Les boucles

C'est donc autour de ces instructions que s'articulera le cours d'algorithmique et de programmation.

Exercices du livre :

Exercices 3, 4 p. 358

Les Variables

Définition

Dans un script, **une variable** est une **boîte** étiquetée dans laquelle des données sont rangées. Une variable peut être un **caractère**, un **mot**, un **nombre**, etc, qui peuvent évoluer en fonction des **instructions** du script.

La variable porte donc un nom et une valeur lui est affectée.

La variable est le premier élément à créer lors de la construction de l'algorithme. Il faut créer la boîte et lui donner un nom. On appelle cette étape **la déclaration des variables**.

L'affectation des variables

La variable ne sera utilisée dans l'algorithme que d'une façon : elle se verra **attribuer une valeur**, qui pourra évoluer selon les instructions du script. Il s'agit alors de **remplir la boîte**.

Les variables peuvent donc avoir plusieurs formes :

- **numérique** (nombre, date, montant monétaire...)
- **alphanumérique** (nom, mot...)
- **booléen** (oui/non, vrai/faux, 0/1...) *(du nom du mathématicien George Boole)*

Scratch est conçu de telle sorte que l'utilisateur ne puisse pas se tromper sur les types de valeur et leur manipulation.

Dans certains scripts il est nécessaire de construire plusieurs variables.

Concrètement, dans Scratch, on attribue une valeur à une variable de cette façon :

On attribue la valeur 8 à la variable A. On retiendra que c'est la variable située à gauche qui est affectée.

La valeur de la variable N est 10 (puisque A =8). Seule la variable située à gauche, N, est affectée, la variable A n'est pas modifiée.

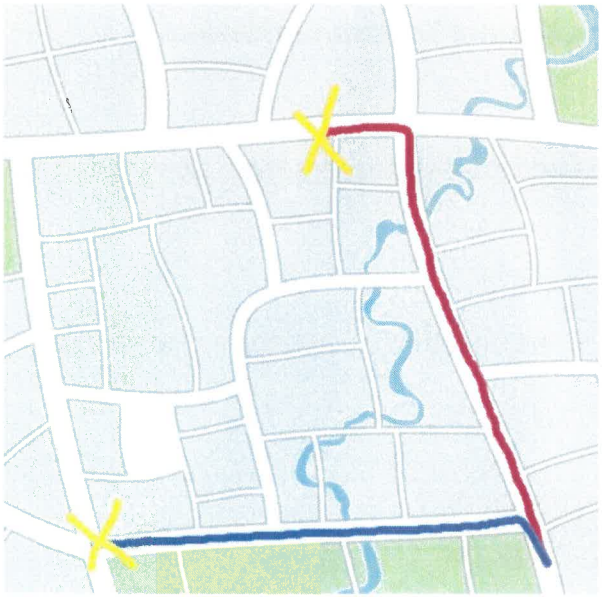
Ordre des instructions

L'ordre dans lequel sont données les instructions joue un rôle essentiel dans le résultat final.

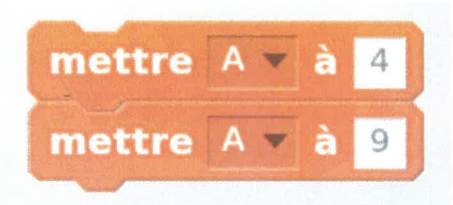
Par exemple : on donne à deux personnes ces indications :

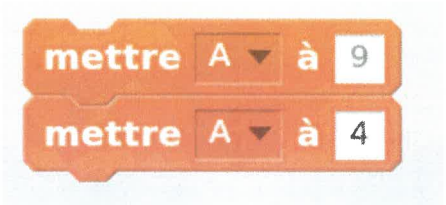
- *Prenez la prochaine à gauche, roulez 2 km*
- *Roulez 2 km, prenez la prochaine à gauche.*

Il y a peu de chances que ces personnes se retrouvent au même endroit à l'arrivée.



Comparons ces deux algorithmes :



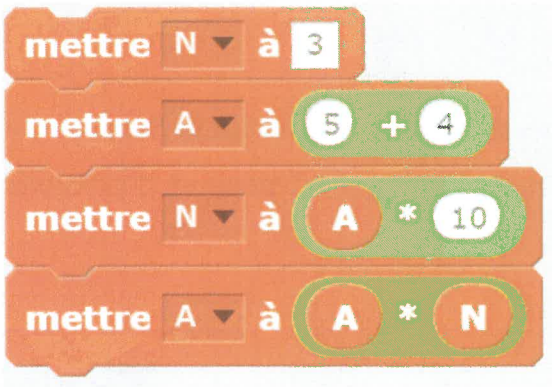


Dans le premier exemple A =9 et dans le second A =4.

Expressions et opérateurs

On a vu que la variable était située à gauche dans l'instruction d'affection. La partie située à droite est appelée l'expression.

Une expression est un ensemble de valeurs, reliées par des opérateurs, et équivalent à une seule valeur.



Voici des exemples d'expressions.

On utilise plusieurs sortes d'opérateurs :

- Les classiques : + - × ÷
- Les logiques (ou booléens) : **ET, OU, NON...**

Dans Scratch on peut également trouver le bloc 

qui donne le reste d'une division non entière.

Par exemple :

donnera en résultat 

puisque $\frac{16}{5} = 5 + \frac{1}{16}$ *ou* $5\ R\ 1$.

aura pour résultat 

puisque $\frac{23}{6} = 18 + \frac{5}{6}$ *ou* $18\ R\ 5$.

Un autre bloc peut être utilisé dans des programmes de calcul Scratch est l'arrondi . 

Comme son nom l'indique, il permet d'arrondir une valeur à l'entier le plus proche.

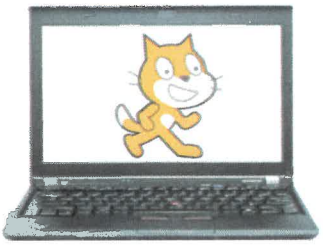
Par exemple :

L'arrondi de π (Pi) 

vaut 

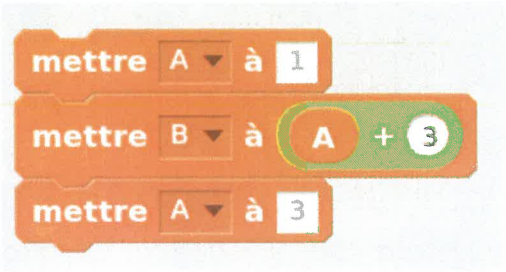
L'arrondi de la racine carrée de 2 

est 

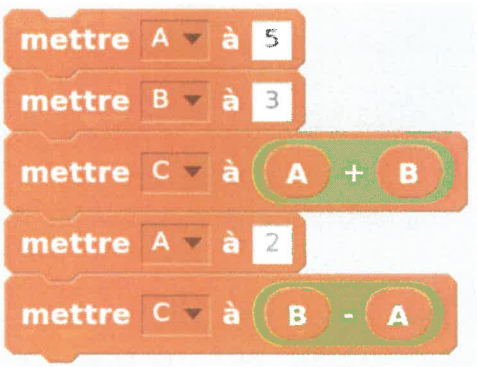


Exercices

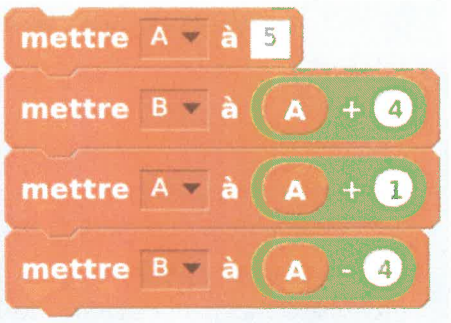
1) Quelles seront les valeurs des variables A et B après les instructions suivantes ?



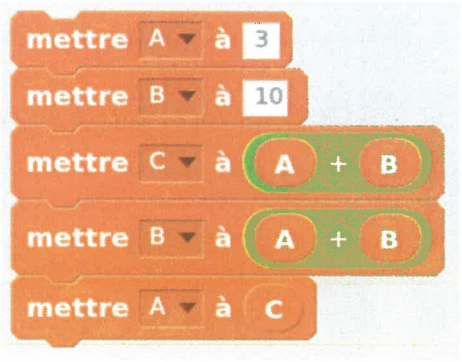
2) Quelles seront les valeurs des variables A, B et C après les instructions suivantes ?



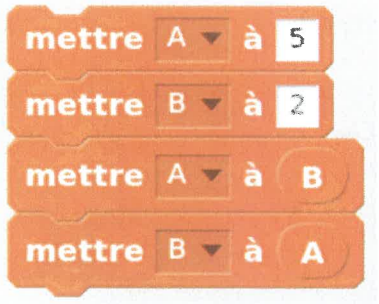
3) Quelles seront les valeurs des variables A et B après les instructions suivantes ?



4) Quelles seront les valeurs des variables A, B et C après les instructions suivantes ?

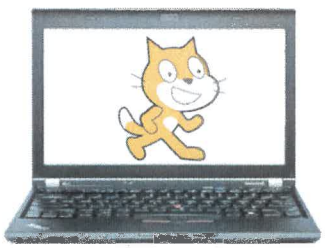


5) Quelles seront les valeurs des variables A et B après les instructions suivantes ? Permettent-elles d'échanger les valeurs de A et de B ? Quel résultat obtient-on en inversant les instructions ?



6) Écrivez un algorithme permettant d'échanger les valeurs de deux variables A et B, quel que soit leur contenu.

7) On dispose de trois variables A, B et C. Écrivez un algorithme permettant de transférer la valeur A dans B, B dans C et C dans A, quelle que soit leur valeur.



Semaine 2

MARDI

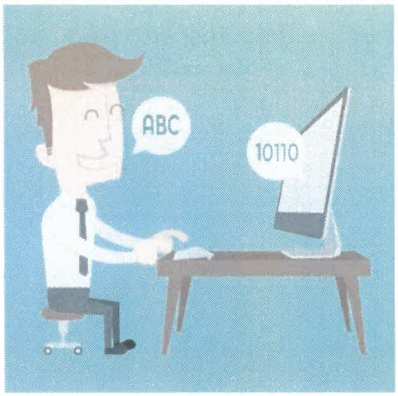
Algorithmique et programmation

La lecture et l'écriture d'un algorithme

Rappelez-vous, construire un algorithme va servir à créer un programme qui sera **exécuté par un ordinateur**. Il faut donc se mettre du point de vue de la machine pour comprendre les notions de lecture et d'écriture.

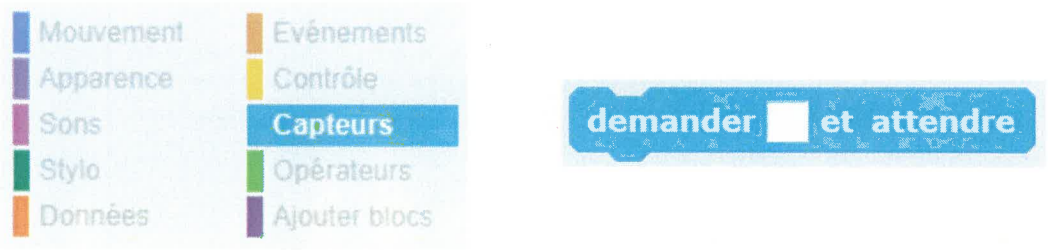
Les instructions permettant à l'utilisateur de saisir des données qui seront ensuite utilisées par le programme, sont **la lecture** (le programme lit les données saisies).

Les instructions qui permettent au programme de donner le résultat à l'utilisateur du programme sont **l'écriture** (elles sont écrites par le programme).



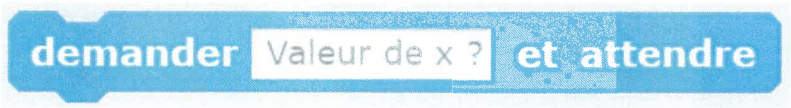
Lecture

Dans Scratch, les instructions de lecture sont dans le menu Capteurs :

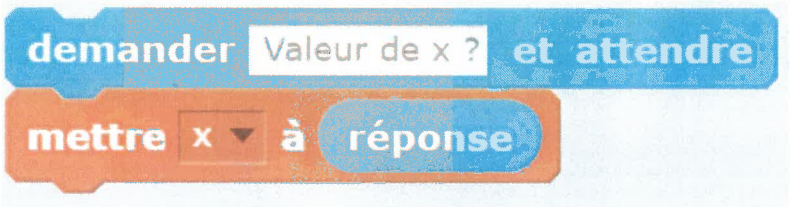


Le programme demande une donnée à l'utilisateur, et attend la suite des instructions. Il est recommandé de **donner un libellé**, c'est-à-dire de préciser la nature de ce qui est demandé pour faciliter l'utilisation du programme.

Par exemple :



Le programme suit ensuite les instructions du programme. Par exemple, ici, le programme demande à l'usager une valeur, il lit la réponse, puis il l'affecte à la variable x.

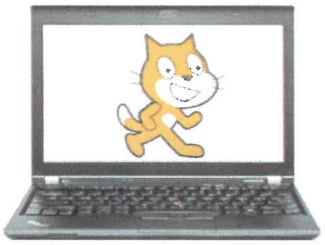


Écriture

Pour obtenir le résultat de l'algorithme, l'exécutant doit recevoir une instruction d'écriture. Dans Scratch, il s'agit par exemple de :

ou encore

Exercices



1) On veut réaliser un algorithme correspondant au programme de calcul suivant :

   - *Choisir un nombre*
   - *Multiplier ce nombre par 5*
   - *Retrancher 1 au résultat.*

Qu'affiche le lutin si on choisit la valeur 8 ?

2) Construisez l'algorithme suivant :

   - *Choisir un nombre*
   - *Ajouter 3 à ce nombre*
   - *Multiplier le résultat par le nombre de départ*

Expliquez pourquoi il est impossible de créer ce script avec une seule variable ?

Achevez votre programme en ajoutant une instruction d'écriture.

3) Écrivez un algorithme utilisant des variables alphanumériques et affichant trois variantes possibles du célèbre vers de Du Bellay *« Heureux qui, comme Ulysse, a fait un beau voyage »*. Ne tenez compte ni des majuscules, ni de la ponctuation.

Exercice du livre :

Exercice 11 p.358

Entraînement sur internet :
se reporter à la liste des sites conseillés Thème programme et algorithme SCRATCH : programme 3ème

Semaine 2

JEUDI

Algorithmique et programmation

Les tests

Le test en algorithme consiste **à poser une condition** dans le programme, qui se trouve alors devant **une alternative**. En effet, soit la condition est remplie, soit elle ne l'est pas.

Heureusement, l'ordinateur est tout à fait capable **d'analyser une situation** et de dire si une condition est remplie ou si elle ne l'est pas.

Il existe **deux formes** de tests en algorithmique :

- une forme plus simple : « **Si… Alors…** »
- une forme plus complexe : « **Si… Alors… Sinon…** »

Mais avant de développer ce chapitre, il faut définir ce qu'est exactement une condition au sens algorithmique.

Une **condition** est une **comparaison**.

Elle est composée d'une valeur, d'un comparatif et d'une autre valeur.

Ainsi, les valeurs peuvent être comparées de différentes façons :

- Égal à...
- Différent de...
- Supérieur à... (strictement ou non)
- Inférieur à... (strictement ou non)

Le test « Si... Alors... »

Dans Scratch, les briques utilisées pour les conditions se trouvent dans le menu Contrôle :

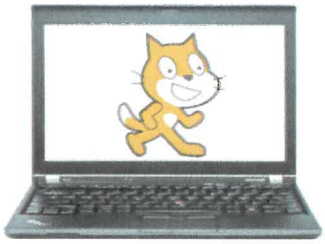
Si la condition est vraie, alors le programme réalise l'instruction demandée avant de passer à la suite. Si la condition est fausse, le programme passe directement à l'instruction suivante.

Par exemple :

Ce script indique à l'utilisateur s'il est mineur ou majeur, selon l'âge qu'il indique.

Construis-le et teste-le dans Scratch.

Exercices



1) Construisez un algorithme permettant d'indiquer si un nombre est positif ou négatif. On ne tiendra pas compte du zéro pour le moment.

2) Construisez un algorithme permettant de comparer deux nombres et de donner le plus grand des deux.

Le test « Si... Alors... Sinon... »

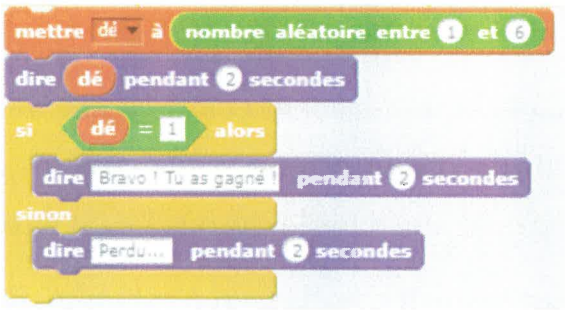
Ce test est identique au test précédent et fonctionne de la même façon. La différence est que **deux instructions** sont données : une dans le cas où la condition est **vraie**, et une dans le cas où elle est **fausse**.

Elle se présente dans Scratch sous la forme :

Il faut se rappeler que le rédacteur du script a dès lors la possibilité de truquer –volontairement ou non– une alternative, en paramétrant le programme de façon à ce que la condition soit remplie, ou non.

Par exemple, on souhaite créer un programme pour un jeu dont le but est de faire un as. Le programme doit jeter le dé (virtuellement, il s'agit en fait de choisir un chiffre au hasard entre 1 et 6), puis annoncer au joueur s'il a gagné, ou s'il a perdu.

Voici la construction dans Scratch :



Construis le script et teste-le dans Scratch.

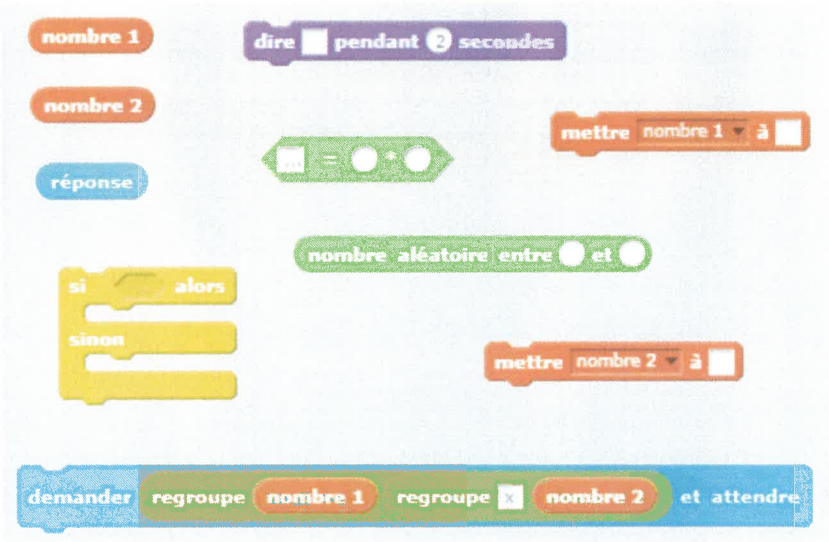
Exercices du livre :

Exercices 19 et 22 p.359

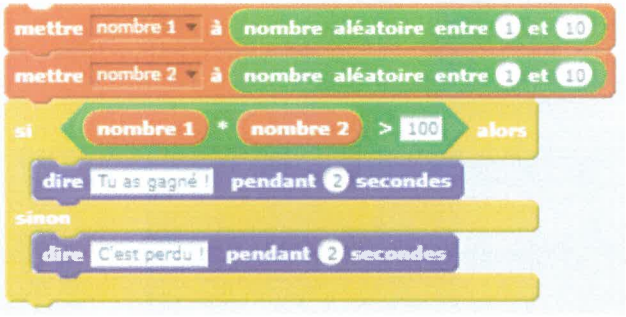
Exercices :

1) À l'aide des blocs ci-dessous (attention, il pourrait en manquer certains !), construis un algorithme permettant de jouer à ce jeu :

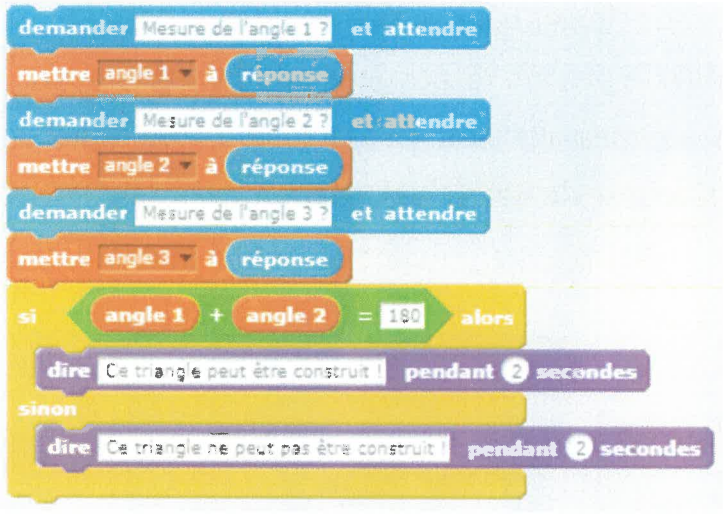
- L'ordinateur choisit deux nombres entre 1 et 10
- Il demande le produit de ces deux nombres au joueur
- Il affiche un message pour dire au joueur si sa réponse est correcte.
- Enfin il propose de refaire une partie.



2) Explique quelle est l'erreur dans le script suivant :



3) Corrige l'algorithme ci-dessous qui calcule si un triangle est construisible ou non après avoir demandé la mesure de ses trois angles.



Semaine 2

VENDREDI

Algorithmique et programmation

• Les conditions composées

Nous avons vu hier les deux types de tests possibles en programmation :

- *« Si... Alors... »*
- *« Si... Alors... Sinon... »*

Or, pour certains problèmes, ces conditions ne peuvent suffire. En effet, comment traduire une condition telle que : *« Si la variable N est comprise entre 10 et 20... »* ? Ou bien : *« Si la variable M n'est pas supérieure à 15... »* ? Ou encore *« Si la variable P est inférieure ou égale à 100... »* ?

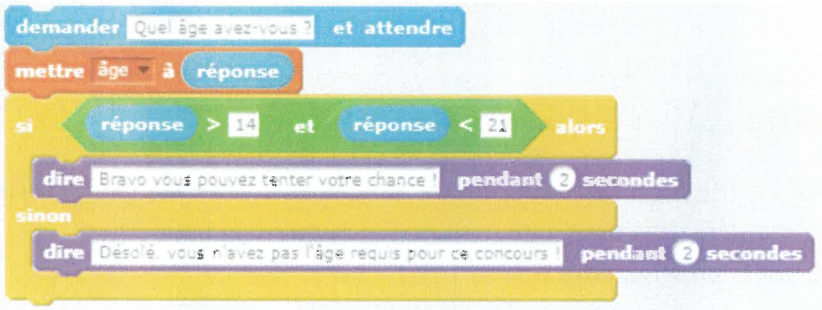
Il s'agit là en réalité de **conditions composées**, grâce à des **opérateurs logiques**. Ils sont au nombre de quatre en algorithmique, mais seulement trois sont disponibles dans Scratch.

L'opérateur logique « ...ET... »

L'opérateur logique ET est le plus simple. Il s'agit d'associer deux conditions, et que les deux soient vérifiées, pour que la condition composée soit remplie.

Par exemple :

Imaginons qu'un programme gère les inscriptions des participants à un concours réservé aux 15-20 ans.



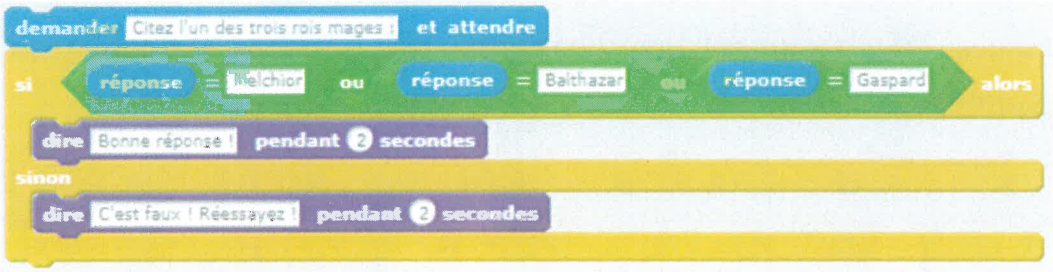
L'algorithme devra être construit avec une condition composée et l'opérateur logique ET, puisque les participants devront avoir **à la fois** plus de 14 ans et moins de 21. Les deux conditions doivent être remplies pour valider le test.

Construis le script et teste-le dans Scratch.

L'opérateur logique « ...OU... » :

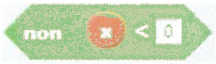
Dans le cas où un test doit remplir soit une condition, soit une autre, pour être validé, c'est l'opérateur « ...OU... » qui sera utilisé. Dans ce cas de figure, il suffit que l'une des deux conditions soit vérifiée pour que le test soit validé.

Dans cet algorithme, on demande au joueur de citer le nom d'un des trois rois mages :



Pour valider le test, le joueur peut citer n'importe lequel des trois noms. Si aucune des trois conditions n'est remplie, alors le test échoue et l'instruction « Sinon… » s'applique.

L'opérateur logique « NON… »

Le NON inverse tout simplement une condition. Admettons que l'on recherche des nombres positifs ou nuls (supérieurs ou égaux à 0), il est possible de poser la condition suivante : 

En effet, dans Scratch, les signes ≤ et ≥ n'existent pas.

L'opérateur logique « …XOR… »

Cet opérateur n'est pas présent dans Scratch, pour autant il existe bel et bien en algorithmique. Il s'agit d'un opérateur OU exclusif (*eXclusive OR* en anglais), c'est-à-dire que pour que le test soit validé, il faut que l'une et seulement l'une des conditions soit remplie.

Par exemple, pour valider une commande dans un restaurant qui propose des menus, on pourra choisir entre deux entrées, deux plats et deux desserts, mais pas prendre les six propositions. La condition sera validée une fois qu'une entrée, un plat et un dessert auront été choisis.

Les tables de vérité

Les tables de vérité sont souvent utilisées pour illustrer les opérateurs logiques. Elles résument bien l'ensemble des possibilités que l'on peut rencontrer.

A	B	A ET B	A OU B	NON A	A XOR B
Vrai	Vrai	Vrai	Vrai	Faux	Faux
Vrai	Faux	Faux	Vrai	Faux	Vrai
Faux	Vrai	Faux	Vrai	Vrai	Vrai
Faux	Faux	Faux	Faux	Vrai	Faux

Exercice :

Quelles valeurs vérifient les conditions suivantes ?

a. 

b. 

c. 

d. 

e. 

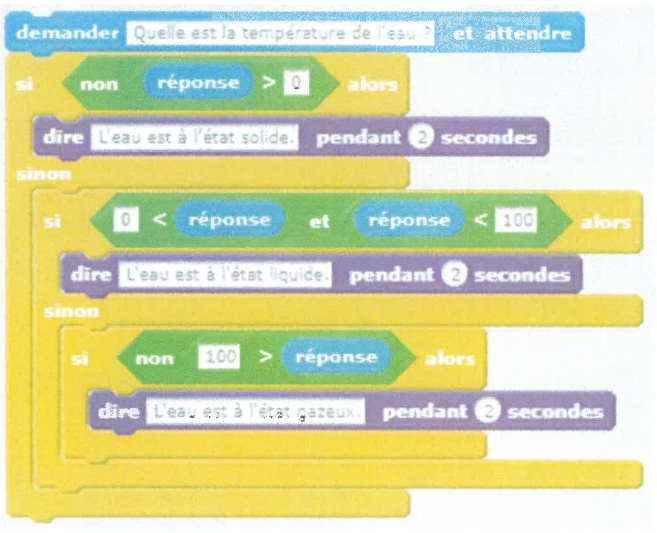
• Les tests imbriqués

Les tests contenus dans un algorithme peuvent donc être composés, mais également imbriqués.

Les tests imbriqués servent à **associer** plusieurs combinaisons entre elles et à **combiner** plusieurs solutions possibles.

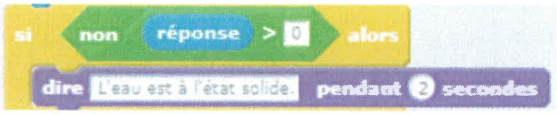
Par exemple, un programme peut avoir à donner l'état de l'eau en fonction de sa température. Pour cela, trois tests différents doivent être imbriqués.

Le voici dans Scratch :



Dans ce script, le programme demande à l'utilisateur d'indiquer la température de l'eau. La première condition est *« NON réponse > 0 »*, autrement dit *« si la température est inférieure ou égale à zéro° »*. Si cette

condition est vraie, alors l'instruction donnée est d'écrire *« L'eau est à l'état solide. »*

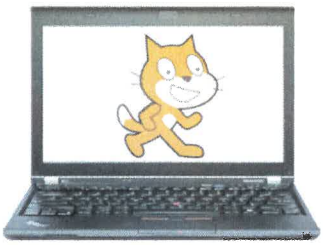


Si la condition est fausse, le programme teste une autre condition, à savoir si la température est comprise strictement entre zéro et cent degrés. Si cette deuxième condition est vraie, le programme affiche *« L'eau est à l'état liquide. »*.

Si cette deuxième condition est fausse, alors le programme teste une troisième condition, *« NON réponse < 100 »*, c'est-à-dire *« si la température est supérieure ou égale à 100° »*. Si cette troisième et dernière condition est vraie, le programme affiche *« L'eau est à l'état gazeux »*.

Construis le script et teste-le dans Scratch.

Exercices



1) Construisez un algorithme qui demande un nombre à l'utilisateur, puis qui lui dit si ce nombre est positif, négatif, ou nul.

2) Proposez un algorithme utilisant le théorème de Pythagore qui, après avoir demandé la longueur des trois côtés du triangle ABC, est capable de dire si le triangle est rectangle et en quel sommet.

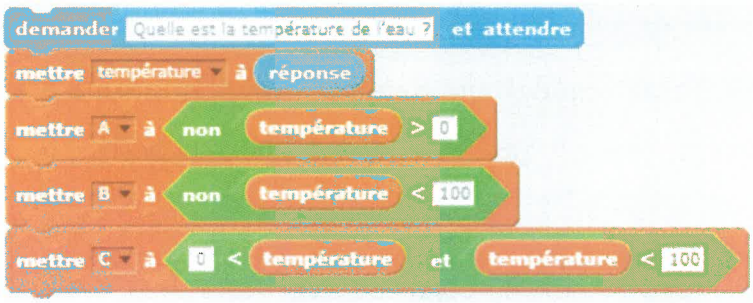
3) Écrire un algorithme qui détermine la catégorie des utilisateurs selon leur âge (entre 6 et 15 ans) :

- 6 et 7 ans : « Poussin »
- 8 et 9 ans : « Pupille »
- 10 et 11 ans : « Minime »
- Plus de 12 ans : « Cadet »

• Les variables booléennes

En algorithmique **les variables booléennes** sont du type **binaire**, vrai ou faux. Le terme booléen vient du nom du mathématicien George Boole qui les a inventés.

Reprenons notre exemple sur l'état de l'eau en utilisant des variables booléennes :



La réponse est affectée à la variable température, puis cette variable est testée par l'algorithme. La réponse rendue est **de la forme 0/1, oui/non, vrai/faux...** Vous noterez que dans Scratch le résultat est soit *true*, soit *false*.

La réponse 10 donnera un résultat du type :

Les conditions A et B sont fausses.

La condition C est vérifiée : la température est en effet comprise entre 0 et 100.

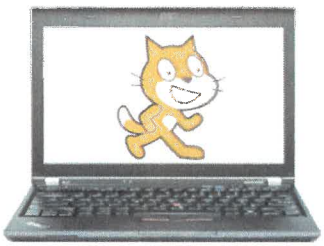
La réponse -50 donnera ce résultat :

La condition A est vraie : la température de l'eau est négative.

Les conditions B et C sont fausses.

Construis le script et teste-le dans Scratch.

Exercices :



1) Les habitants du pays Boole sont imposés de la façon suivante :
   - Les hommes sont imposés dès 25 ans
   - Les femmes sont imposées entre 20 et 35 ans seulement
   - Les autres ne sont pas imposés.

   Construisez un algorithme booléen qui permet d'indiquer aux habitants s'ils sont imposables ou non.

2) Construisez un algorithme booléen qui demande deux nombres A et B à l'utilisateur, puis annonce lequel des deux est le plus petit.