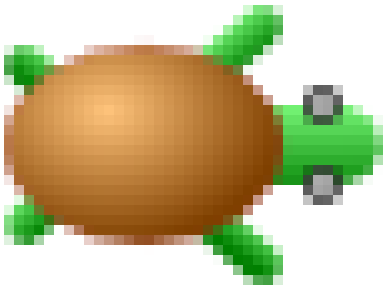
http://irem.univ-reunion.fr/spip.php?article817

Corrigé du sujet de bac ES Pondichery 2015 (algorithme)

- Algorithmique et programmation
- CoffeeScript
- Sofus, ou la programmation visuelle par blocs en collège (et au lycée)
- Sophus ancienne version (sans Blockly)
-



Date de mise en ligne : lundi 20 avril 2015

Le sujet est ici. L'exercice 2 Â« obli/L Â» portait sur une suite arithmético-géométrique :

> Un apiculteur souhaite étendre son activité de production de miel à une nouvelle région. En juillet 2014, il achète 300 colonies d'abeilles qu'il installe dans cette région.
> Après renseignements pris auprès des services spécialisés, il s'attend à perdre 8% des colonies durant l'hiver. Pour maintenir son activité et la développer, il a prévu d'installer 50 nouvelles colonies chaque printemps.

[http://irem.univ-reunion.fr/local/cache-vignettes/L400xH122/api3-fbc38.jpg]

Une nouveauté de ce sujet est qu'il n'est plus nécessaire, pour comprendre l'algorithme, de savoir que la perte de 8% correspond à une multiplication par 0,92 puisque le calcul, tel que décrit dans l'algorithme, est

```
C prend la valeur C   C×0,08 + 50
```

Mais Sophus sait soustraire un pourcentage donc le problème ne se pose pas avec Sophus. L'algorithme de l'énoncé demande de boucler avec cette condition :

```
Tant que C < 400
```

Puisque C désigne visiblement le nombre de colonies, on en déduit que le but de cet algorithme est de savoir au bout de combien d'années l'apiculteur aura dépassé les 400 colonies. Alors, autant faire cette traduction sophusienne :

Nom dans le sujet	*Nom dans Sophus*
n	années
C	colonies

L'algorithme se traduit alors ainsi en Sophus [1] :

colonies devient nouvelle Variable 300
années devient nouvelle Variable 0
Tant que colonies.valeur < 400
 diminuer colonies de 8 pourcents
 augmenter colonies de 50
 incrémenter années
montrer années <a href='http://irem.univ-reunion.fr/local/cache-code/e3a551b64f9f7e83efd90ef7d7d8e529.txt' style='font-family: verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger

Mais l'exercice était plus d'algorithmique Â« papier-crayon Â» que de programmation, alors il y avait un tableau à remplir, dans lequel on devait donner successivement les valeurs des deux variables ainsi que le test de comparaison avec 400. Pour constituer un tel tableau dans Sophus, on va effectuer les opérations suivantes :

[http://irem.univ-reunion.fr/local/cache-vignettes/L185xH155/pileserviettes-bbd98.jpg]

- créer des listes, sous la forme de crochets vides (ce ne sont pas des variables Sophus, juste des tableaux de JavaScript) [2] ;
- à chaque passage dans la boucle, empiler dans ces tableaux les valeurs courantes des variables Sophus [3]. Par exemple le nombre actuel de colonies est *colonies.valeur* et non *colonies* [4] ;
- Tout à la fin, on affiche (Â« montrer Â») toute la pile d'un coup, ce qui évite de multiplier les affichages.

Voici les noms des tableaux :

Nom du tableau	Signification
test	valeurs de vérité de la proposition "C<400"
valeurDeC	nombre actuel de colonies
valeurDeN	temps écoulé, en années

Le script qui donne les trois lignes du tableau :

```
test devient [ ]
valeurDeC devient [ ]
valeurDeN devient [ ]
colonies devient nouvelle Variable 300
années devient nouvelle Variable 0
Tant que colonies.valeur < 400
 diminuer colonies de 8 pourcents
 augmenter colonies de 50
 incrémenter années
 test.empiler colonies.valeur<400
 valeurDeC.empiler colonies.valeur
 valeurDeN.empiler années.valeur
montrer test
montrer valeurDeC
montrer valeurDeN<div class='code_download'   style='text-align: right;'> <a
href='http://irem.univ-reunion.fr/local/cache-code/8b772c5e46722751e1b5dafe6dd50498.txt'   style='font-family:
verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger
```

Et le tableau obtenu [5] :

Il y a moins de 400 colonies	c'est vrai	c'est vrai	c'est vrai	c'est vrai	c'est faux
nombre de colonies	326	349,92	371,926	392,172	410,799
années passées	1	2	3	4	5

Représentation graphique de la suite

Sophus n'a pas de capacité graphique mais alcoffeethmique, si ; voici la représentation graphique de la suite en question :

Le script CoffeeScript pour avoir ce graphique

```
c=300
suite=[c]
for n in [1..50]
 c*=.92
 c+=50
 suite.push c
dessineSuite suite, 50, 300, 700, 2, "red"<div class='code_download'   style='text-align: right;'> <a
href='http://irem.univ-reunion.fr/local/cache-code/e1e8dea967660095c98b82ee8662f4fc.txt'   style='font-family:
verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger
```

0102030405030040050060070 0

Question 3

L'apiculteur espère doubler son nombre initial de colonies. Il voudrait savoir combien d'années il lui faudra pour atteindre cet objectif.

Pour répondre à cette question, il suffit de changer la valeur du seuil dans l'algorithme du début. Au cas où on aurait du mal à savoir quel est le double de 300, on peut tenter cet algorithme :

```
seuil = nouvelle Variable 300
doubler seuil
montrer seuil<div class='code_download'   style='text-align: right;'> <a
href='http://irem.univ-reunion.fr/local/cache-code/0bcdd235b31e40bd34bfe23fb2758cf9.txt'   style='font-family:
verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger
```

On sait maintenant comment modifier l'algorithme du début :

colonies devient nouvelle Variable 300
années devient nouvelle Variable 0
Tant que colonies.valeur < 600
 diminuer colonies de 8 pourcents
 augmenter colonies de 50
 incrémenter années
montrer années <a href='http://irem.univ-reunion.fr/local/cache-code/df1079a9731e0858686e902da29f4149.txt' style='font-family: verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger

interpréteur en ligne

Pour tester les scripts, les copier-coller ci-dessous puis expérimenter avec :
sophus @font-face { font-family: 'Sofia'; font-style: normal; font-weight: 400; src: local('Sofia'), local('Sofia-Regular'), url(http://fonts.gstatic.com/s/sofia/v5/f2JBzUtFBEg-kZNlFQ7f-Q.woff2) format('woff2'), url(http://fonts.gstatic.com/s/sofia/v5/eNNA2vlZnUPtgq9g__E3cA.woff) format('woff'); } #scriptCoff0 { font-family: "Arial", "Helvetica", sans-serif; width: 90%; height: 90%; border: 5px ridge lightGray; background-color: Black; color: LightGreen; } #academy { width: 90%; height: 90%; border: 3px ridge lightGray; background-color: lightyellow; font-family: 'Sofia', cursive; color: brown; } var Variable, aLaPuissance, aMoinsQue, arrondir, auCarré, auCube, augmenter, carré, centupler, cinq, cinquième, cinquièmes, cos, cosinus, cube, dans, de, demi, demis, deux, diminuer, diviser, dix, dixième, dixièmes, doubler, décimales, décrémenter, décupler, entrer, estTableau, extraireLaRacineDe, huit, huitième, huitièmes, incrémenter, inverser, mettre, mettreDans, montrer, multiplier, neuf, neuvième, neuvièmes, octupler, par, pourcent, pourcents, près, quadrupler, quart, quarts, quatre, quintupler, racine, ref, ref1, ref2, sept, septième, septièmes, sextupler, si, sin, sinon, sinus, six, sixième, sixièmes, taille, tan, tangente, tiers, tripler, trois, tronquer, un, x, à, élever, éleverAuCarré, éleverAuCube, À; Boolean.prototype.toLocaleString = function() { if (this.valueOf()) { return " c'est vrai "; } else { return " c'est faux "; } }; Number.prototype.foisFaire = function(fn) { var i, j, ref, results; results = []; for (i = j = 0, ref = this; 0 <= ref ? j < ref : j > ref; i = 0 <= ref ? ++j : --j) { results.push(fn()); } return results; }; Number.prototype.tantQuePlusGrandQue = function(variable, fn) { var results; results = []; while (variable.valeur < this) { results.push(fn()); } return results; }; Number.prototype.tantQuePlusPetitQue = function(variable, fn) { var results; results = []; while (variable.valeur > this) { results.push(fn()); } return results; }; Array.prototype.empiler = function(machin) { return this.push(machin); }; Array.prototype.toLocaleString = "[" + ((function() { var j, len, results; results = []; for (j = 0, len = this.length; j < len; j++) { x = this[j]; results.push(x.toLocaleString()); } return results; }).call(this)) + "]"; estTableau = function(o) { if ((o.valeur != null) && Array.isArray(o.valeur)) { return true; } else { return Array.isArray(o); } }; taille = function(o) { if (o.valeur != null) { return o.valeur.length; } else { return o.length; } }; décimales = "décimales"; près = "près"; à = "à"; de = "de"; par = "par"; dans = "dans"; sinon = "sinon"; pourcents = "pourcents"; pourcent = "pourcents"; auCarré = "au carré"; auCube = "au cube"; aLaPuissance = "à la puissance"; ref = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], un = ref[0], deux = ref[1], trois = ref[2], quatre = ref[3], cinq = ref[4], six = ref[5], sept = ref[6], huit = ref[7], neuf = ref[8], dix = ref[9]; ref1 = ["demis", "tiers", "quarts", "cinquièmes", "sixièmes", "septièmes", "huitièmes", "neuvièmes", "dixièmes"], demis = ref1[0], tiers = ref1[1], quarts = ref1[2], cinquièmes = ref1[3], sixièmes = ref1[4], septièmes = ref1[5], huitièmes = ref1[6], neuvièmes = ref1[7], dixièmes = ref1[8]; ref2 = ["demis", "tiers", "quarts", "cinquièmes", "sixièmes", "septièmes", "huitièmes", "neuvièmes", "dixièmes"], demi = ref2[0], tiers = ref2[1], quart = ref2[2], cinquième = ref2[3], sixième = ref2[4], septième = ref2[5], huitième = ref2[6], neuvième = ref2[7], dixième = ref2[8]; À = Math.PI; montrer = (function(_this) { return function(o) { if (o.valeur != null) { if (Array.isArray(o.valeur)) { return alert("[" + ((function() { var j, len, ref3, results; ref3 = o.valeur; results = []; for (j = 0, len = ref3.length; j < len; j++) { x = ref3[j]; results.push(" " + x.toLocaleString() + " "); } return results; })()) + "]"); } else { return alert(o.valeur.toLocaleString()); } } else { if (((function() { var j, len, results; results = []; for (j = 0, len = o.length; j < len; j++) { x = o[j]; results.push(x); } return results; })()).length > 1) { return alert("[" + ((function() { var j, len, results; results = []; for (j = 0, len = o.length; j < len; j++) { x = o[j]; results.push(" " + x.toLocaleString() + " "); } return results; })()) + "]"); } else { return

```
alert(o.toLocaleString()); } } }; })(this); entrer = (function(_this) { return function(o) { return o.valeur = prompt("Quelle valeur donner à cette variable ?"); }; })(this); Variable = (function() { function Variable(valeur) { this.valeur = valeur != null ? valeur : 0; } Variable.prototype.toString = function() { return this.valeur.toLocaleString(); }; Variable.prototype.estPositif = function() { return this.valeur > 0; }; Variable.prototype.estNégatif = function() { return this.valeur < 0; }; Variable.prototype.estNul = function() { return this.valeur === 0; }; Variable.prototype.estPair = function() { return this.valeur % 2 === 0; }; Variable.prototype.estImpair = function() { return this.valeur % 2 === 1; }; return Variable; })(); mettreDans = (function(_this) { return function(o, bidule) { var ref3; o.valeur = (ref3 = bidule.valeur) != null ? ref3 : bidule; return null; }; })(this); mettre = (function(_this) { return function(bidule, dans, o) { var ref3; if (dans == null) { dans = "dans"; } o.valeur = (ref3 = bidule.valeur) != null ? ref3 : bidule; return null; }; })(this); carré = function(nombre) { return nombre * nombre; }; cube = function(nombre) { return carré(nombre) * nombre; }; racine = function(nombre) { return Math.sqrt(nombre); }; sinus = function(nombre) { return Math.sin(nombre * À / 180); }; cosinus = function(nombre) { return Math.cos(nombre * À / 180); }; tangente = function(nombre) { return sinus(nombre) / cosinus(nombre); }; sin = function(x) { return Math.sin(x); }; cos = function(x) { return Math.cos(x); }; tan = function(x) { return Math.tan(x); }; éleverAuCarré = (function(_this) { return function(o) { return o.valeur *= o.valeur; }; })(this); éleverAuCube = (function(_this) { return function(o) { return o.valeur *= o.valeur * o.valeur; }; })(this); inverser = (function(_this) { return function(o) { return o.valeur = 1 / o.valeur; }; })(this); extraireLaRacineDe = (function(_this) { return function(o) { return o.valeur = Math.sqrt(o.valeur); }; })(this); élever = (function(_this) { return function(o, a, exposant) { var ref3; if (exposant == null) { exposant = 2; } switch (a) { case "à la puissance": return o.valeur = Math.pow(o.valeur, (ref3 = exposant.valeur) != null ? ref3 : exposant); case "au carré": return o.valeur *= o.valeur; case "au cube": return o.valeur *= o.valeur * o.valeur; default: return alert("Je veux bien élever cette variable mais à quelle puissance ?"); } }; })(this); arrondir = (function(_this) { return function(o, a, epsilon, ordre) { if (a == null) { a = ""; } if (epsilon == null) { epsilon = 1; } if (ordre == null) { ordre = "décimales"; } if (a === "") { o.valeur = Math.round(o.valeur); } if (a === "à") { if (ordre === "décimales") { o.valeur = Math.round(o.valeur * Math.pow(10, epsilon)) / Math.pow(10, epsilon); } if (ordre === "près") { return o.valeur = Math.round(o.valeur / epsilon) * epsilon; } } }; })(this); tronquer = (function(_this) { return function(o, a, epsilon, ordre) { if (a == null) { a = ""; } if (epsilon == null) { epsilon = 1; } if (ordre == null) { ordre = "décimales"; } if (a === "") { o.valeur = Math.floor(o.valeur); } if (a === "à") { if (ordre === "décimales") { o.valeur = Math.floor(o.valeur * Math.pow(10, epsilon)) / Math.pow(10, epsilon); } if (ordre === "près") { return o.valeur = Math.floor(o.valeur / epsilon) * epsilon; } } }; })(this); doubler = (function(_this) { return function(o) { if (Array.isArray(o.valeur)) { return o.valeur = (function() { var j, len, ref3, results; ref3 = o.valeur; results = []; for (j = 0, len = ref3.length; j < len; j++) { x = ref3[j]; results.push(2 * x); } return results; })(); } else { return o.valeur *= 2; } }; })(this); tripler = (function(_this) { return function(o) { if (Array.isArray(o.valeur)) { return o.valeur = (function() { var j, len, ref3, results; ref3 = o.valeur; results = []; for (j = 0, len = ref3.length; j < len; j++) { x = ref3[j]; results.push(3 * x); } return results; })(); } else { return o.valeur *= 3; } }; })(this); quadrupler = (function(_this) { return function(o) { if (Array.isArray(o.valeur)) { return o.valeur = (function() { var j, len, ref3, results; ref3 = o.valeur; results = []; for (j = 0, len = ref3.length; j < len; j++) { x = ref3[j]; results.push(4 * x); } return results; })(); } else { return o.valeur *= 4; } }; })(this); quintupler = (function(_this) { return function(o) { if (Array.isArray(o.valeur)) { return o.valeur = (function() { var j, len, ref3, results; ref3 = o.valeur; results = []; for (j = 0, len = ref3.length; j < len; j++) { x = ref3[j]; results.push(5 * x); } return results; })(); } else { return o.valeur *= 5; } }; })(this); sextupler = (function(_this) { return function(o) { if (Array.isArray(o.valeur)) { return o.valeur = (function() { var j, len, ref3, results; ref3 = o.valeur; results = []; for (j = 0, len = ref3.length; j < len; j++) { x = ref3[j]; results.push(6 * x); } return results; })(); } else { return o.valeur *= 6; } }; })(this); octupler = (function(_this) { return function(o) { if (Array.isArray(o.valeur)) { return o.valeur = (function() { var j, len, ref3, results; ref3 = o.valeur; results = []; for (j = 0, len = ref3.length; j < len; j++) { x = ref3[j]; results.push(8 * x); } return results; })(); } else { return o.valeur *= 8; } }; })(this); décupler = (function(_this) { return function(o) { if (Array.isArray(o.valeur)) { return o.valeur =
```

```
(function() {  var j, len, ref3, results;  ref3 = o.valeur;  results = [];  for (j = 0, len = ref3.length; j < len; j++) {  x = ref3[j];  results.push(10 * x);  }  return results;  })();  } else {  return o.valeur *= 10;  }  };  })(this);  centupler = (function(_this) {  return function(o) {  if (Array.isArray(o.valeur)) {  return o.valeur = (function() {  var j, len, ref3, results;  ref3 = o.valeur;  results = [];  for (j = 0, len = ref3.length; j < len; j++) {  x = ref3[j];  results.push(100 * x);  }  return results;  })();  } else {  return o.valeur *= 100;  }  };  })(this);  incrémenter = (function(_this) {  return function(o) {  o.valeur += 1;  return null;  };  })(this);  décrémenter = (function(_this) {  return function(o) {  o.valeur -= 1;  return null;  };  })(this);  augmenter = (function(_this) {  return function(o, de, chouia, mode) {  var ref10, ref11, ref12, ref13, ref3, ref4, ref5, ref6, ref7, ref8, ref9;  if (mode == null) {  mode = "";  }  if (estTableau(o)) {  if (mode === "") {  if (estTableau(chouia)) {  if ((taille(chouia)) === (taille(o))) {  o.valeur = o.valeur.map(function(courant, place) {  var ref3;  return courant + ((ref3 = chouia.valeur) != null ? ref3 : chouia)[place];  });  } else {  alert("erreur de dimension");  }  }  }  } else {  if (de === "de") {  switch (mode) {  case "":  o.valeur += (ref3 = chouia.valeur) != null ? ref3 : chouia;  break;  case "demis":  o.valeur *= 1 + ((ref4 = chouia.valeur) != null ? ref4 : chouia) / 2;  break;  case "tiers":  o.valeur *= 1 + ((ref5 = chouia.valeur) != null ? ref5 : chouia) / 3;  break;  case "quarts":  o.valeur *= 1 + ((ref6 = chouia.valeur) != null ? ref6 : chouia) / 4;  break;  case "cinquièmes":  o.valeur *= 1 + ((ref7 = chouia.valeur) != null ? ref7 : chouia) / 5;  break;  case "sixièmes":  o.valeur *= 1 + ((ref8 = chouia.valeur) != null ? ref8 : chouia) / 6;  break;  case "septièmes":  o.valeur *= 1 + ((ref9 = chouia.valeur) != null ? ref9 : chouia) / 7;  break;  case "huitièmes":  o.valeur *= 1 + ((ref10 = chouia.valeur) != null ? ref10 : chouia) / 8;  break;  case "neuvièmes":  o.valeur *= 1 + ((ref11 = chouia.valeur) != null ? ref11 : chouia) / 9;  break;  case "dixièmes":  o.valeur *= 1 + ((ref12 = chouia.valeur) != null ? ref12 : chouia) / 10;  break;  case "pourcents":  o.valeur *= 1 + ((ref13 = chouia.valeur) != null ? ref13 : chouia) / 100;  }  } else {  alert("Je veux bien augmenter cette variable mais de combien ?");  }  }  return null;  };  })(this);  diminuer = (function(_this) {  return function(o, de, chouia, mode) {  var ref10, ref11, ref12, ref13, ref3, ref4, ref5, ref6, ref7, ref8, ref9;  if (mode == null) {  mode = "";  }  if (estTableau(o)) {  if (mode === "") {  if (estTableau(chouia)) {  if ((taille(chouia)) === (taille(o))) {  o.valeur = o.valeur.map(function(courant, place) {  var ref3;  return courant - ((ref3 = chouia.valeur) != null ? ref3 : chouia)[place];  });  } else {  alert("erreur de dimension");  }  }  } else {  if (de === "de") {  switch (mode) {  case "":  o.valeur -= (ref3 = chouia.valeur) != null ? ref3 : chouia;  break;  case "demis":  o.valeur *= 1 - ((ref4 = chouia.valeur) != null ? ref4 : chouia) / 2;  break;  case "tiers":  o.valeur *= 1 - ((ref5 = chouia.valeur) != null ? ref5 : chouia) / 3;  break;  case "quarts":  o.valeur *= 1 - ((ref6 = chouia.valeur) != null ? ref6 : chouia) / 4;  break;  case "cinquièmes":  o.valeur *= 1 - ((ref7 = chouia.valeur) != null ? ref7 : chouia) / 5;  break;  case "sixièmes":  o.valeur *= 1 - ((ref8 = chouia.valeur) != null ? ref8 : chouia) / 6;  break;  case "septièmes":  o.valeur *= 1 - ((ref9 = chouia.valeur) != null ? ref9 : chouia) / 7;  break;  case "huitièmes":  o.valeur *= 1 - ((ref10 = chouia.valeur) != null ? ref10 : chouia) / 8;  break;  case "neuvièmes":  o.valeur *= 1 - ((ref11 = chouia.valeur) != null ? ref11 : chouia) / 9;  break;  case "dixièmes":  o.valeur *= 1 - ((ref12 = chouia.valeur) != null ? ref12 : chouia) / 10;  break;  case "pourcents":  o.valeur *= 1 - ((ref13 = chouia.valeur) != null ? ref13 : chouia) / 100;  }  } else {  alert("Je veux bien diminuer cette variable mais de combien ?");  }  }  return null;  };  })(this);  multiplier = (function(_this) {  return function(o, par, facteur, mode) {  var ref10, ref11, ref12, ref13, ref3, ref4, ref5, ref6, ref7, ref8, ref9;  if (par == null) {  par = "par";  }  if (mode == null) {  mode = "";  }  if (estTableau(o)) {  return o.valeur = o.valeur.map(function(courant) {  var ref3;  return courant *= (ref3 = facteur.valeur) != null ? ref3 : facteur;  });  } else {  if (par === "par") {  switch (mode) {  case "":  return o.valeur *= (ref3 = facteur.valeur) != null ? ref3 : facteur;  case "demis":  return o.valeur *= ((ref4 = facteur.valeur) != null ? ref4 : facteur) / 2;  case "tiers":  return o.valeur *= ((ref5 = facteur.valeur) != null ? ref5 : facteur) / 3;  case "quarts":  return o.valeur *= ((ref6 = facteur.valeur) != null ? ref6 : facteur) / 4;  case "cinquièmes":  return o.valeur *= ((ref7 = facteur.valeur) != null ? ref7 : facteur) / 5;  case "sixièmes":  return o.valeur *= ((ref8 = facteur.valeur) != null ? ref8 : facteur) / 6;  case "septièmes":  return o.valeur *= ((ref9 = facteur.valeur) != null ? ref9 : facteur) / 7;  case "huitièmes":  return o.valeur *= ((ref10 = facteur.valeur) != null ? ref10 : facteur) / 8;  case "neuvièmes":  return o.valeur *= ((ref11 =
```

facteur.valeur) != null ? ref11 : facteur) / 9; case "dixièmes": return o.valeur *= ((ref12 = facteur.valeur) != null ? ref12 : facteur) / 10; case "pourcents": return o.valeur *= ((ref13 = facteur.valeur) != null ? ref13 : facteur) / 100; default: return alert("erreur opérateur"); } } else { return alert("Je veux bien multiplier cette variable mais par quoi ?"); } } }; })(this); diviser = (function(_this) { return function(o, par, facteur, mode) { var ref10, ref11, ref12, ref3, ref4, ref5, ref6, ref7, ref8, ref9; if (par == null) { par = "par"; } if (mode == null) { mode = ""; } if (estTableau(o)) { return o.valeur = o.valeur.map(function(courant) { var ref3; return courant /= (ref3 = facteur.valeur) != null ? ref3 : facteur; }); } else { if (par === "par") { switch (mode) { case "": return o.valeur /= (ref3 = facteur.valeur) != null ? ref3 : facteur; case "demis": return o.valeur /= ((ref4 = facteur.valeur) != null ? ref4 : facteur) / 2; case "tiers": return o.valeur /= ((ref5 = facteur.valeur) != null ? ref5 : facteur) / 3; case "quarts": return o.valeur /= ((ref6 = facteur.valeur) != null ? ref6 : facteur) / 4; case "cinquièmes": return o.valeur /= ((ref7 = facteur.valeur) != null ? ref7 : facteur) / 5; case "sixièmes": return o.valeur /= ((ref8 = facteur.valeur) != null ? ref8 : facteur) / 6; case "septièmes": return o.valeur /= ((ref9 = facteur.valeur) != null ? ref9 : facteur) / 7; case "huitièmes": return o.valeur /= ((ref10 = facteur.valeur) != null ? ref10 : facteur) / 8; case "neuvièmes": return o.valeur /= ((ref11 = facteur.valeur) != null ? ref11 : facteur) / 9; case "dixièmes": return o.valeur /= ((ref12 = facteur.valeur) != null ? ref12 : facteur) / 10; default: return alert("erreur opérateur"); } } else { return alert("Je veux bien diviser cette variable mais par quoi ?"); } } }; })(this); si = function(booleen, fonction) { if (booleen) { fonction(); } return null; }; aMoinsQue = function(booleen, fonction) { if (!booleen) { fonction(); } return null; }; var correction, traduction; traduction = function(dutexte) { var texte; texte = dutexte; texte = texte.replace(/\ dans\ /g, " in "); texte = texte.replace(/pour\ /g, "for "); texte = texte.replace(/Pour\ /g, "for "); texte = texte.replace(/Sinon/g, "else"); texte = texte.replace(/sinon/g, "else"); texte = texte.replace(/Si\ /g, "if "); texte = texte.replace(/\ nouvelle\ /g, " new "); texte = texte.replace(/Tant que\ /g, "while "); texte = texte.replace(/tant que\ /g, "while "); texte = texte.replace(/Jusqu'à ce que\ /g, "until "); texte = texte.replace(/jusqu'à ce que\ /g, "until "); texte = texte.replace(/\ +fois\ +faire/g, ".foisFaire -Â»"); texte = texte.replace(/\ +au carré/g, ", auCarré"); texte = texte.replace(/\ +au cube/g, ", auCube"); texte = texte.replace(/\ +à la puissance/g, ", aLaPuissance,"); texte = texte.replace(/extraire la racine de/g, "extraireLaRacineDe "); texte = texte.replace(/([^\,])\ +à\ +([^\,])/g, "$1, à, $2"); texte = texte.replace(/([^\,])\ +de\ +([^\,])/g, "$1, de, $2"); texte = texte.replace(/([^\,])\ +par\ +([^\,])/g, "$1, par, $2"); texte = texte.replace(/([^\,])\ +demi/g, "$1, demi"); texte = texte.replace(/([^\,])\ +tiers/g, "$1, tiers"); texte = texte.replace(/([^\,])\ +quart/g, "$1, quart"); texte = texte.replace(/([^\,])\ +cinquième/g, "$1, cinquième"); texte = texte.replace(/([^\,])\ +sixième/g, "$1, sixième"); texte = texte.replace(/([^\,])\ +septième/g, "$1, septième"); texte = texte.replace(/([^\,])\ +huitième/g, "$1, huitième"); texte = texte.replace(/([^\,])\ +neuvième/g, "$1, neuvième"); texte = texte.replace(/([^\,])\ +dixième/g, "$1, dixième"); texte = texte.replace(/([^\,])\ +pourcent/g, "$1, pourcent"); texte = texte.replace(/([^\,])\ +près/g, "$1, près"); texte = texte.replace(/([^\,])\ +décimale/g, "$1, décimale"); texte = texte.replace(/\ devient\ /g, " = "); texte = texte.replace(/:=/g, "="); return texte; }; correction = function(unAlgo) { var texte; texte = unAlgo; texte = texte.replace(/\.foisFaire\ *-Â»/g, " fois faire"); texte = texte.replace(/-Â»/g, " faire "); texte = texte.replace(/,(\ )*de(\ )*,/g, " de "); texte = texte.replace(/,(\ )*par(\ )*,/g, " par "); texte = texte.replace(/,(\ )*à(\ )*,/g, " à "); texte = texte.replace(/,(\ )*dans(\ )*,/g, " dans "); texte = texte.replace(/,(\ )*aLaPuissance(\ )*,/g, " à la puissance "); texte = texte.replace(/Variable/g, "variable initialisée à "); texte = texte.replace(/==/g, " est égale à "); texte = texte.replace(/^-Â»/g, " est supérieure à "); texte = texte.replace(/>=/g, " est supérieure ou égale à "); texte = texte.replace(/<=/g, " est inférieure ou égale à "); texte = texte.replace(/ Interpréteur Sophus

```
colonies devient nouvelle Variable 300
années devient nouvelle Variable 0
Tant que colonies.valeur < 400
    diminuer colonies de 8 pourcents
    augmenter colonies de 50
    incrémenter années
montrer années
```

Lancer cet algorithme

Copier-coller l'algorithme

Sortie en français