

تعلّم البرمجة مع القط سكراتش



نورا حاتم

تعلم البرمجة مع القط سكراتش

نورا ابراهيم حاتم

هذه الوثيقة متاحة برخصة
المشاع الإبداعي: نسب المصنف -
الترخيص بالمثل، الإصدار 4.0.
مع مراعاة أن كافة الأسماء
والشعارات والعلامات التجارية
الواردة في هذه الوثيقة هي ملك
لأصحابها.
لمزيد من التفاصيل راجع الرابط
التالي:
[CreativeCommons.org/licenses/by-sa/4.0](https://creativecommons.org/licenses/by-sa/4.0)

الفهرس


12 ما هو السكراتش


14 واجهة البرنامج

17 الكائن

23 لبنات الأحداث

23..... 

24..... 

24..... 

25..... 











25..... 

25..... 

25..... 

25..... 

27 لبنات التحكم

27.....	
27.....	
28.....	
28.....	
29.....	
29.....	
29.....	
30.....	
30.....	
31.....	
31.....	

32..... لبنات الحركة.

32..... move steps

32..... turn  degrees

32..... turn  degrees

33..... point in direction

34..... point towards

34..... go to x: y:

34..... go to mouse-pointer

35..... glide secs to x: y:

35..... change x by

35..... set x to

35..... change y by

35..... set y to

36..... if on edge, bounce

36..... set rotation style left-right

37..... x position

37.....	y position
37.....	direction
38.....	لبنات المظاهر
38.....	say Hello! for 2 secs
38.....	say
39.....	think for secs
39.....	think
39.....	show
39.....	hide
40.....	switch costume to
40.....	next costume
40.....	switch backdrop to
40.....	change effect by
41.....	set effect to
41.....	clear graphic effects
42.....	change size by

42.....

42.....

42.....

43.....

43.....

43.....

44..... لبنات العمليات

44.....

44.....

44.....

44.....

44.....

45.....

45.....

45.....

46.....

46.....

46.....	not 
48.....	join 
49.....	length of 
50.....	 mod 
50.....	round 
50.....	sqrt  of 
53.....	لبينات التحسس
53.....	touching  ?
53.....	touching color  ?
54.....	color  is touching  ?
55.....	distance to 
56.....	ask  and wait
56.....	answer
56.....	key  space  pressed?
57.....	mouse down?
57.....	mouse x
57.....	mouse y
58.....	loudness

58..... video motion on Stage

59..... turn video on

59..... set video transparency to 50 %

59..... timer

60..... reset timer

60..... x position of

61..... current minute

62..... days since 2000

62..... username

63..... لبنات القلم

63..... clear

63..... stamp

64..... pen down

64..... pen up

66..... set pen color to

66..... change pen color by



74.....	
75.....	
75.....	
76.....	لبينات البيانات
78.....	
79.....	
80.....	
80.....	
81.....	
82.....	
83.....	
84.....	
84.....	
85.....	
85.....	
86.....	

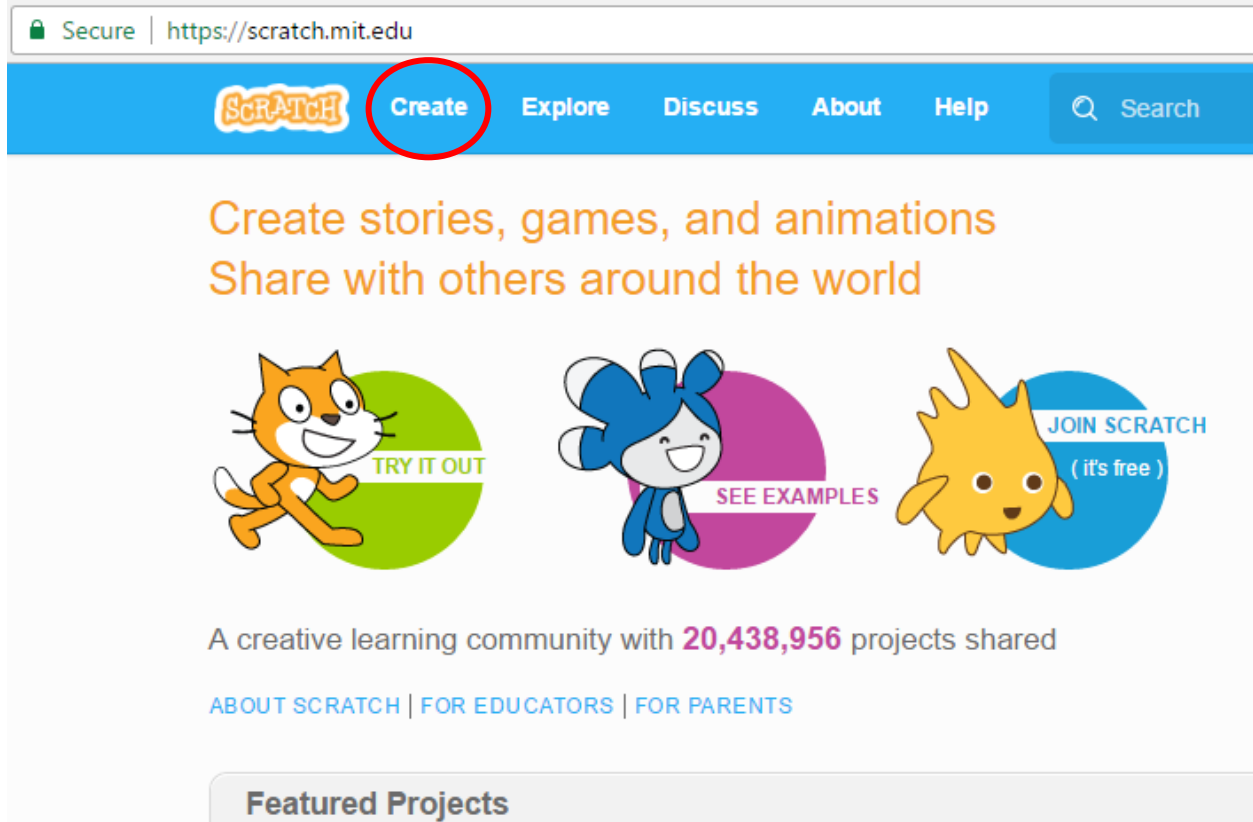
- 86.....
- 87.....المثال الأول: حركة كائن.
- 87.....المثال الثاني: رسم مربع.
- 88.....المثال الثالث: رسم مثلث متساوي الأضلاع.
- 90.....المثال الرابع: رسم دائرة.
- 91.....المثال الخامس: الساعة.
- 95.....المثال السادس: الآلة الحاسبة.
- 96.....المثال السابع: أعماق البحار.
- 100.....المثال الثامن: زوجي أم فردي؟
- 101.....المثال التاسع: الأعداد الزوجية في مجال.
- 102.....المثال العاشر: خمن الرقم الذي اختاره القط.
- 105.....المثال الحادي عشر: الأعداد الزوجية ضمن مجموعة.
- 106.....المثال الثالث عشر: إطعام الدب الجائع.
- 111.....المثال الرابع عشر: العامل $n!$.
- 111.....المثال الخامس عشر: مقياس الحب.

ما هو السكراش Scratch ؟

سكراش Scratch هو لغة برمجة سهلة ومبسطة تستهدف فئة هواة البرمجة غير المختصين والأطفال المتعطشين للتعلّم والسير على طريق الإبداع. يتيح لنا برنامج السكراش تصميم الألعاب والقصص التفاعلية. وتأتي شهرة السكراش وانتشاره ولا سيّما بين الفتيات والفتيان لسهولة استعماله؛ بحيث يقضي على الصعوبة التي يواجهها الطلاب عادة في مجال البرمجة، فهو -على عكس معظم لغات البرمجة التي تحتاج إلى كتابة أكواد برمجية وحفظ تعليمات- يوفر لمستخدميه بيئة سهلة وواضحة وتعليمات جاهزة مقبولة فيما يسمّى لبنات Blocks، ويمكننا من إدراج صور ومقاطع صوتية من المكتبة الخاصة به أو من حواسيبنا. وتعليم البرمجة للأطفال لا يقتصر على الراغبين بدخول عالم البرمجة في المستقبل؛ بل البرمجة وسيلة تساعد في فهم وإدراك ما حولهم وتحليل المشكلات التي تواجههم في مختلف مناحي الحياة ومعرفة أسبابها وإيجاد حلول لها.

يتوافر برنامج السكراش المفتوح المصدر مجاناً على موقع السكراش: <https://scratch.mit.edu/>

بإمكانك أن تقوم بالبرمجة أونلاين بالضغط على Create كما موضح في الشكل:



أو بإمكانك تنزيله من الرابط التالي واستخدامه دون الحاجة للاتصال بالإنترنت:

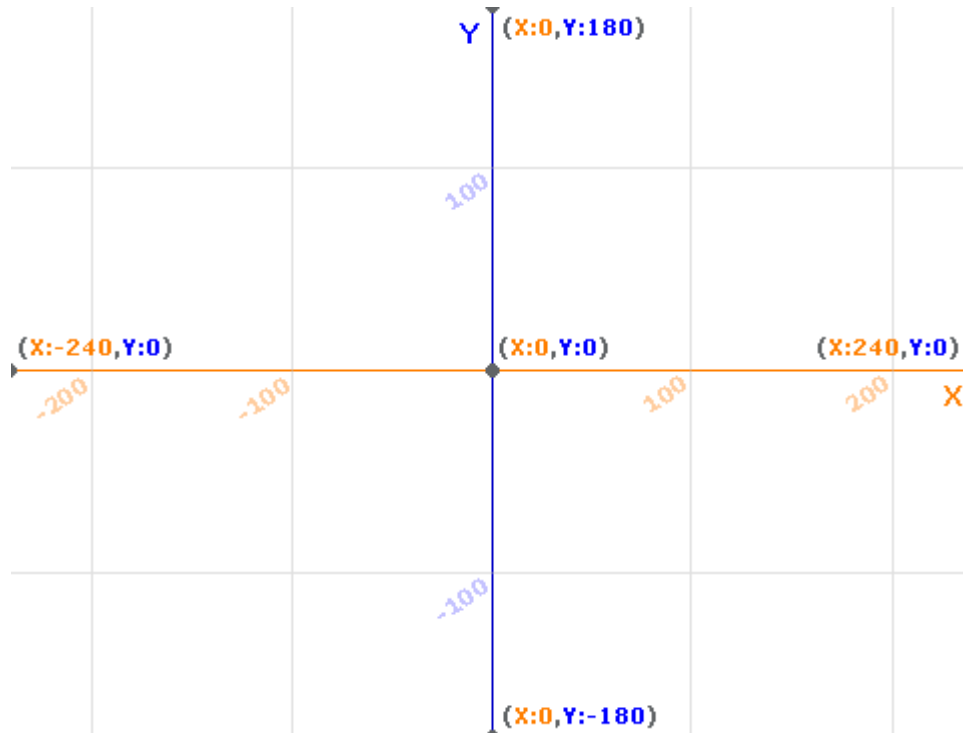
<https://scratch.mit.edu/scratch2download/>

بإمكانك إنشاء حساب على موقع السكراتش وبذلك يتيح لك مشاركة مشاريعك مع برمجين آخرين وكذلك يمكنك من الاطلاع على مشاريعهم.

بعض المفاهيم والمفردات التي سيتم ورودها في هذا الكتاب:

- المنصة stage: مكان حدوث البرنامج حيث يمكن للمستخدم مشاهدة التنفيذ.
- كائن sprite: وهو الذي ينفذ المهام التي يطلبها منه المبرمج على المنصة.
- لبنة block: وتعني تعليمة برمجية تأمر الكائن بالقيام بفعل ما.
- كدسة برمجية: مجموعة من اللبنة.
- محور الفواصل: والمقصود به المحور 'xx'.
- محور الترتيب: والمقصود به المحور 'yy'.
- المترجم: ويُقصد به برنامج السكراتش الذي يقوم بتحويل لبناتك إلى أوامر يفهمها الحاسوب وينفذها مباشرة.

يشكّل محوري الفواصل والترتيب معلماً متجانساً؛ أي محور الفواصل يعامد محور الترتيب وطويلة شعاع الوحدة تساوي الواحد. تمتد المنصة بين الإحداثيات: $x:-240,240$, $y:-180,180$.

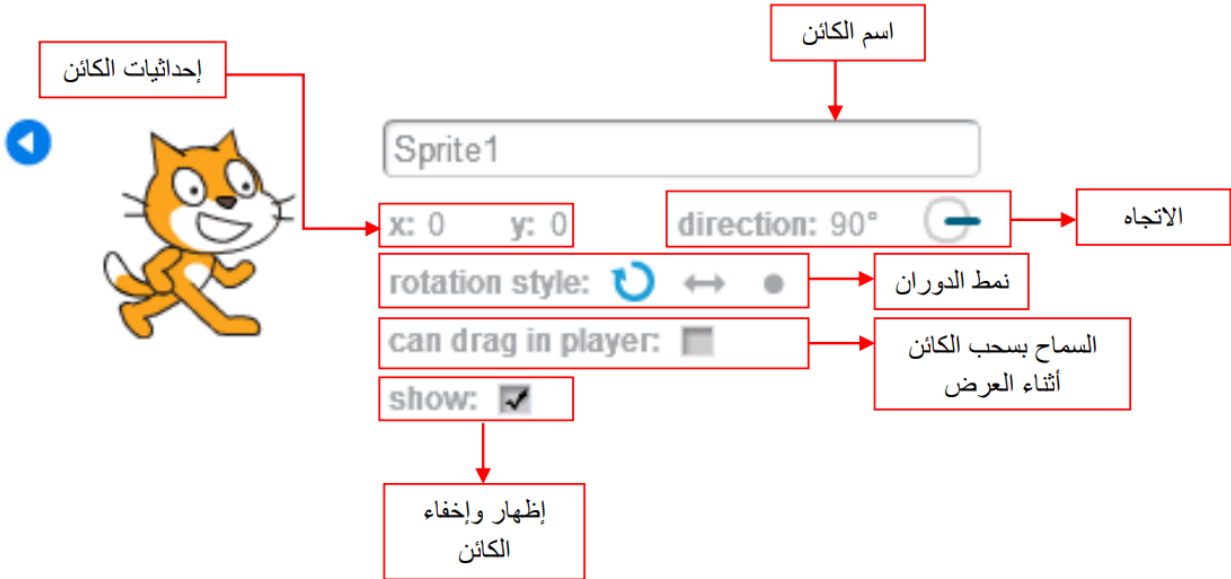


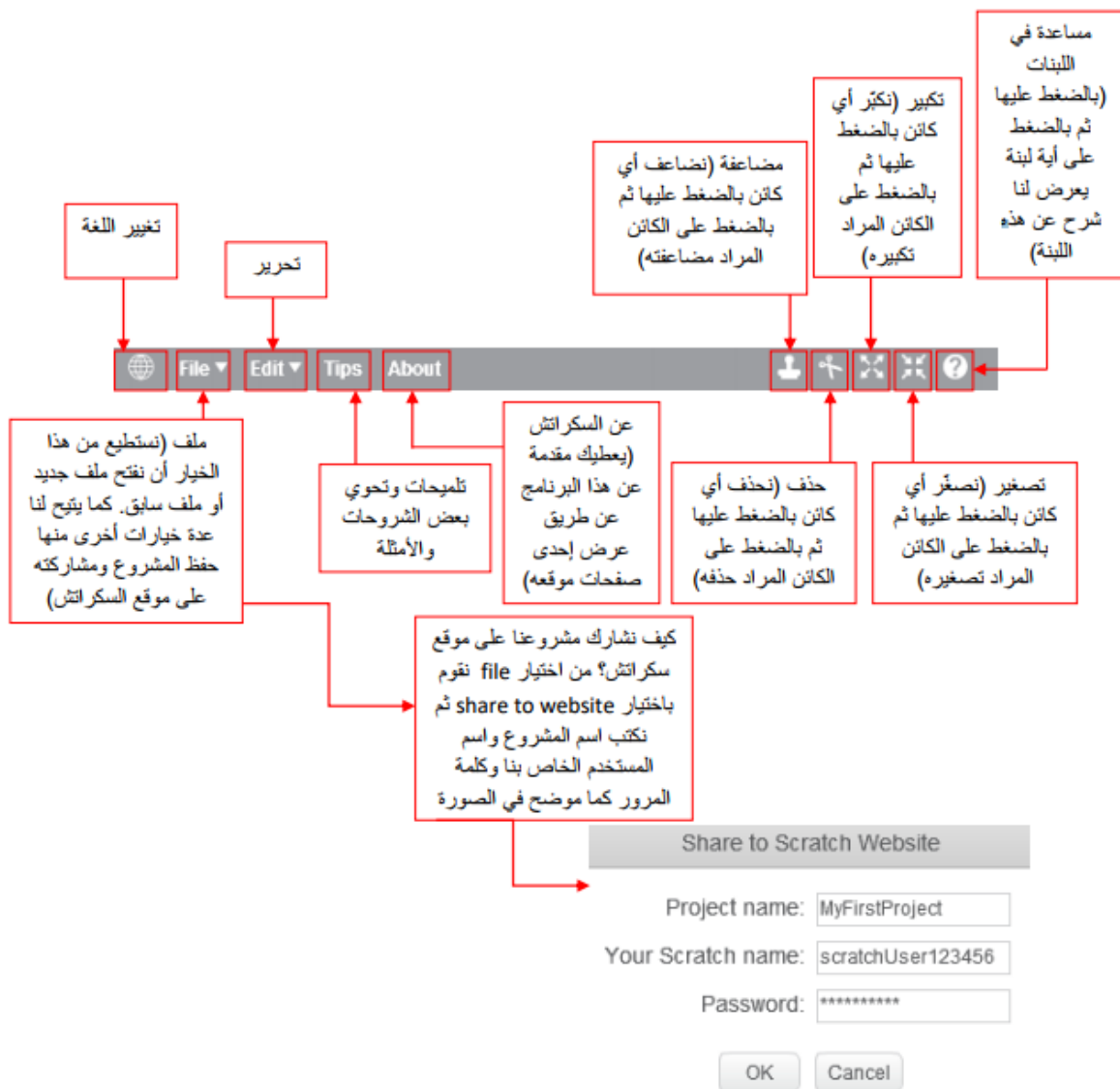
يجب لفت الانتباه أن كلّ نقطة على المنصة لها إحداثيات خاصة بها.

واجهة البرنامج



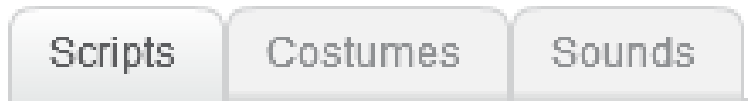
بالضغط على خيارات الكائن نجد مايلي:





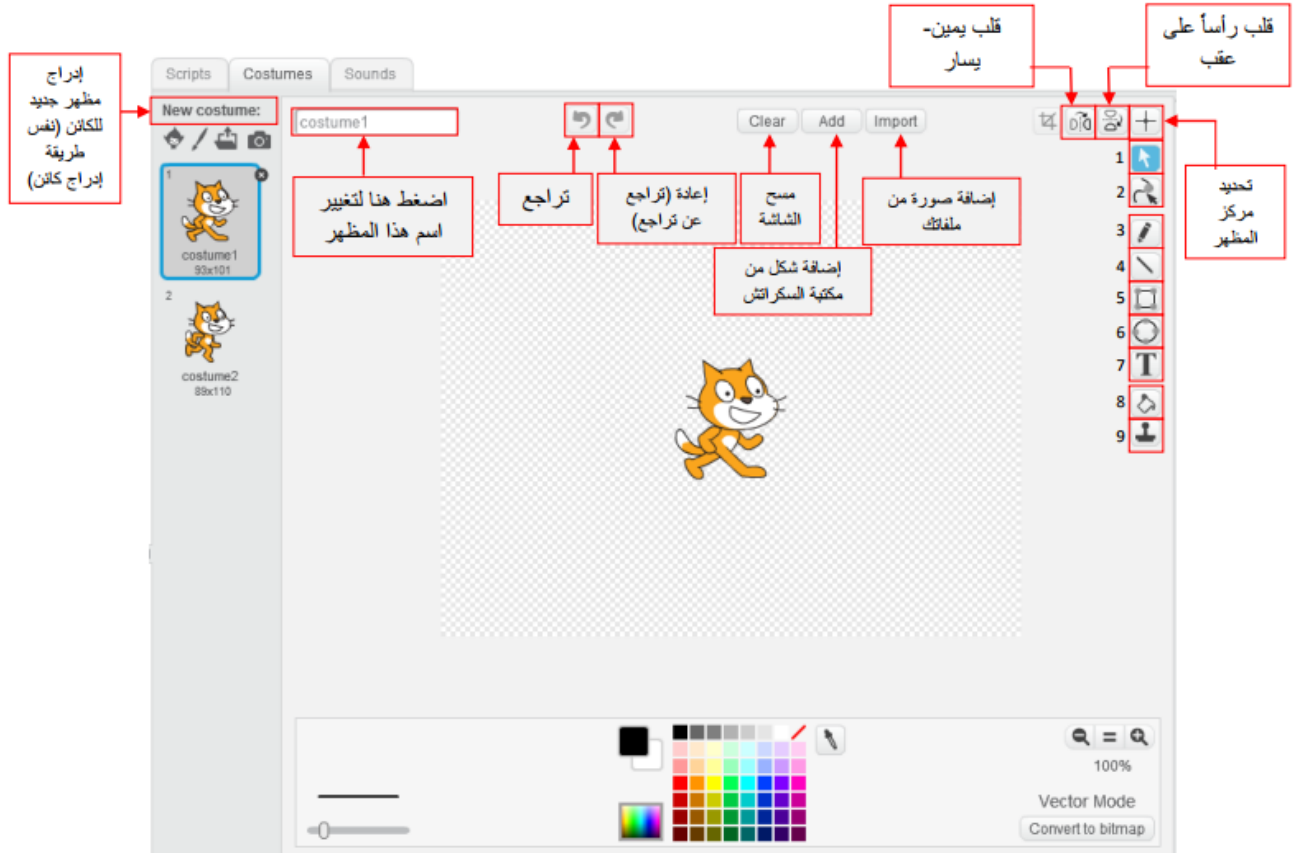
الكائن

نجد لدى أي كائن هذه القائمة التي تظهر في أعلى البرنامج والتي تحوي ثلاثة خيارات. الخيار الأول Scripts ويعني المقاطع البرمجية وهي مسؤولة عن برمجة هذا الكائن. ولدينا الخيار الثاني Costumes ويعني المظاهر. والخيار الثالث Sounds ويعني الصوت وهو مسؤول عن الأصوات الخاصة في هذا الكائن والتي يستطيع أن يشغلها.



ملاحظة: لبرمجة أي كائن، تُسحب اللبانات من مكانها وتوضع في حقل Scripts للكائن المراد برمجته.

أولاً: المظاهر Costumes



1: تحديد الشكل.

2: إعادة تشكيل المظهر؛ بالضغط عليها ثم الضغط على المظهر تظهر نقاط لمعالم المظهر، حرّك النقاط ليأخذ المظهر الشكل الذي تريده.

3: القلم.

4: خط مستقيم.

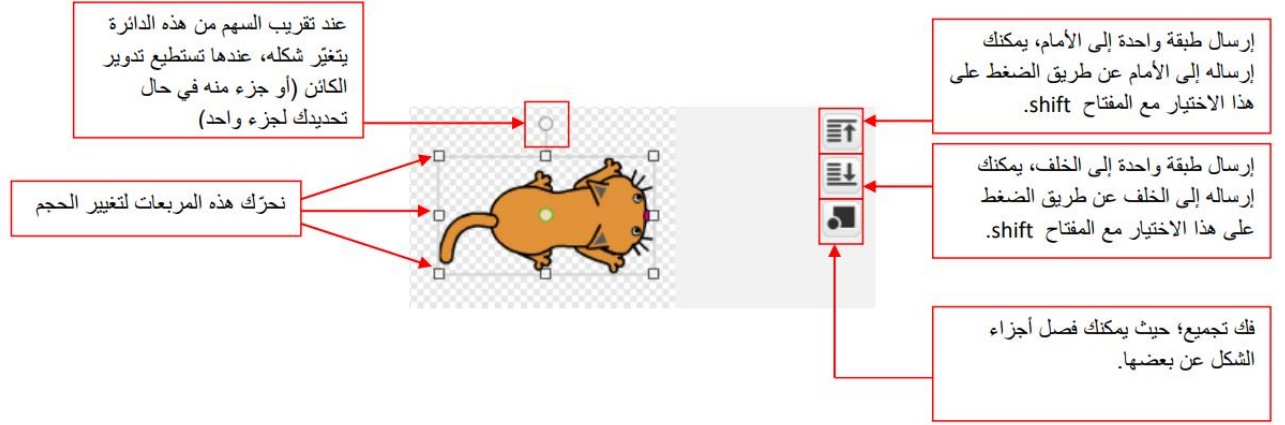
5: مستطيل، يمكنك رسم مربع من خلال الضغط على المفتاح shift أثناء الرسم.

6: إهليلج (شكل بيضوي)، يمكنك رسم دائرة من خلال الضغط على المفتاح shift أثناء الرسم.

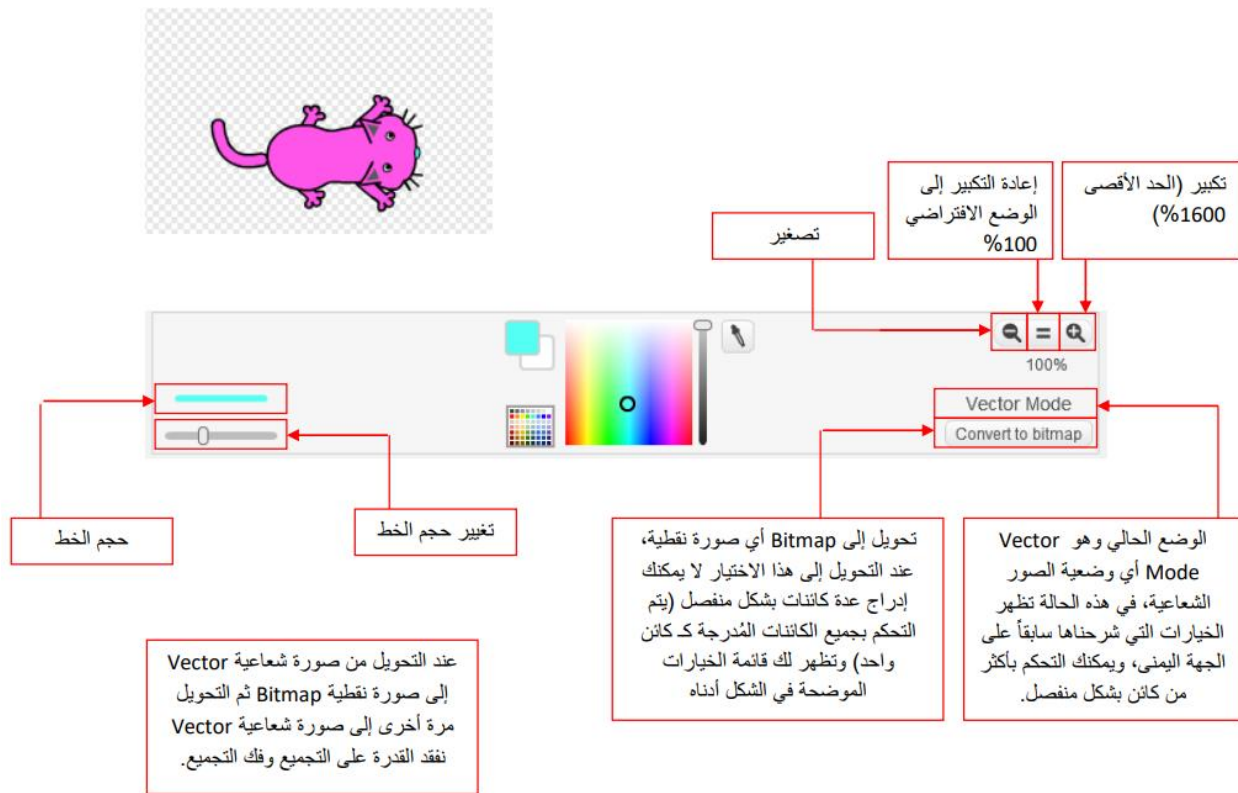
7: نص.

8: تلوين الشكل.

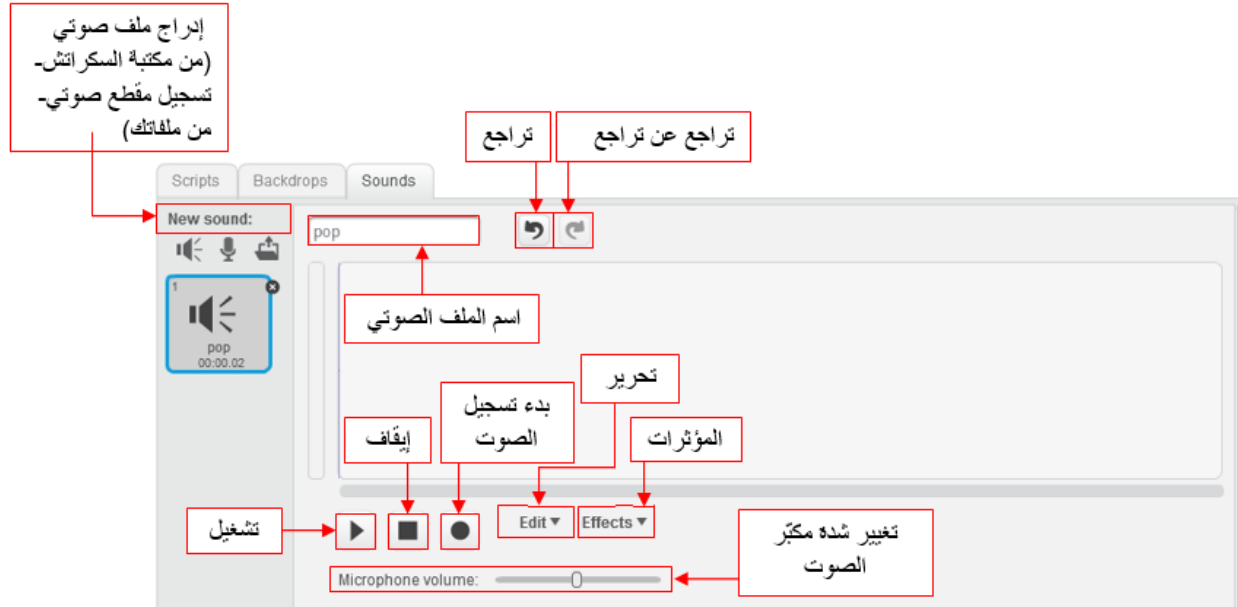
إذا ضغطنا على كائن مُدرج في هذا المحرر نلاحظ ظهور عدة خيارات جديدة، لنتعرّف عليها.



في الشكل الآتي قمنا بتغيير شكل القطعة السابقة عن طريق مجموعة من التعديلات: غيّرنا لونها، وعكسنا اتجاه ذيلها ، ودوّرنا أطرافها، وغيّرنا موضع العينين.



ثانياً: الأصوات Sounds



ماذا يوفر لك خيار المؤثرات Effects؟

- تلاشي الصوت في البداية.
 - تلاشي الصوت في النهاية.
 - زيادة شدة الصوت.
 - تخفيض شدة الصوت.
 - صمت (سكون).
 - عكس المقطع الصوتي (يبدأ من النهاية وينتهي في البداية).
- أما خيار تحرير Edit فهو يتيح لك خيارات كالتقص والنسخ واللصق والحذف و...

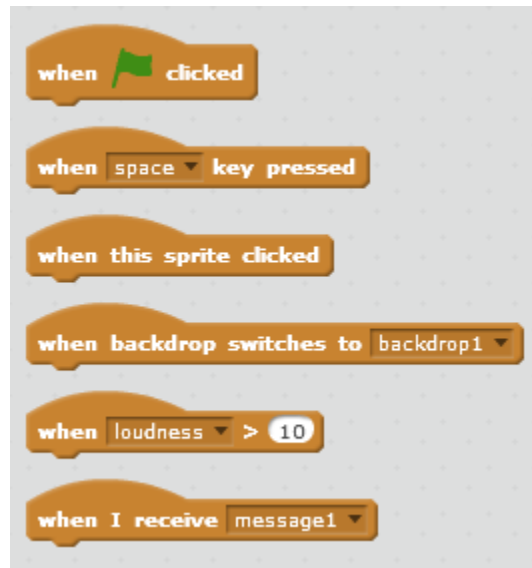
المنصة

التعامل مع المنصة يشبه التعامل مع الكائنات، توجد بعض الاختلافات في اللبانات سنذكرها لاحقاً، ،
لنتعرّف الآن على طريقة إدراج منصة.



لبّات الأحداث Events Blocks

تعدّ لبّات هذه الفئة اللبّات الأكثر ضرورة في كل مشروع سكراتش لأنها المسؤولة عن بدء تنفيذ التعليمات وبالتالي بدء البرنامج.



نلاحظ أن هذه التعليمات تأخذ شكل القمة من الأعلى حيث لا توضع إلا في بداية البرنامج ولا يمكن وضع أي تعليمة قبلها. وفي كل كدسة برمجية لا يمكن أن يوجد إلا تعليمة واحدة منها.

لنبدأ بالتعرّف على التعليمات:



تنفّذ هذه التعليمة اللبّات البرمجية التي تليها عند الضغط على العلم الأخضر المتوضع فوق المنصّة.



تنفّذ هذه التعليمات اللبّات البرمجية التي تليها عند الضغط على الزر المختار من قبل مصمم المشروع. بإمكانك أن تختار أحد الأحرف أو الأرقام المتوضّعة على لوحة المفاتيح إضافة إلى الأسهم و زر المسافة.



تنفّذ هذه التعليمات اللبّات البرمجية التي تليها عند الضغط على الكائن الخاص بها.

when backdrop switches to

أثناء تصميم مشروعي لابدّ من حاجتك إلى العديد من الخلفيات، ولا بدّ من تغيير تصرف كائناتك في كل خلفية. فمثلاً أثناء تصميمك للعبة لا شك من وجود خلفية للترحيب تختلف موسيقاها وكائناتها عن الخلفية التي ستجري فوقها اللعبة. لذلك تنفّذ هذه التعليمات اللبّات البرمجية التي تليها عندما تتغير الخلفية إلى الخلفية المحددة من قبل المصمم (للتعرّف على كيفية تغيير الخلفية اطلع على فصل Looks).

when loudness >

تنفّذ هذه التعليمات اللبّات البرمجية التي تليها عندما تصبح القيمة المختارة من قبل المصمم (شدة الصوت- المؤقت- حركة الفيديو) أكبر من قيمة معينة. نلاحظ أن السكراتش يمكنك من جعل مشروعي تفاعلي ؛ فمثلاً يمكنك استخدام الميكروفون وبذلك يقيس شدة الصوت، ويمكنك أيضاً استخدام كاميرا الويب لرصد حركة الفيديو.



لا بدّ أن جميع من حولك يستعمل وسيلة اتصال للتواصل مع الآخرين، لكن هل تعلم أن الكائنات في السكراتش تفعل ذلك أيضاً؟

يستطيع الكائن في السكراتش إرسال رسالة إلى كائن آخر، وذلك يسهل من عملية تواصل الكائنات مع بعضها، كأن يطلب كائن من آخر أن يختفي مثلاً. ويتم ذلك باستعمال التعليمات الموضحة أعلاه؛ ماعليك إلا اختيار الرسالة الافتراضية أو إنشاء رسالة جديدة.

ولكن كيف سيستقبل الكائن الآخر هذه الرسالة؟

عندما ترسل إلى صديقك رسالة عبر الجوال فيجب أن يكون أحدهما يستطيع الإرسال والآخر يستطيع الاستقبال، وكذلك كائنات السكراتش أحدها يرسل بالتعليمات السابقة والآخر يستقبل بالتعليمات التالية:

when I receive

تنفذ هذه التعليمات اللبنة البرمجية التي تليها فور استقبال رسالة من كائن آخر (فور إرسال كائن رسالة لها).

لكن ماهو الفرق بين الإرسال والإرسال والانتظار؟

في تعليمات broadcast and wait يرسل الكائن الرسالة وينتظر المترجم حتى انتهاء تنفيذ تلك الكدسة البرمجية التي تلي when I receive الخاصة بتلك ال broadcast. بمعنى آخر لا ينتقل المترجم إلى التعليمات التي تلي broadcast and wait إلا عند انتهاء تنفيذ الكدسة البرمجية التي تلي when I receive الخاصة بتلك ال broadcast. أما في حالة broadcast فقط فإنه يرسل الرسالة ويستمر في تنفيذ التعليمات التي تليها بغض النظر عن انتهاء الكدسة البرمجية التي تلي when I receive الخاصة بتلك ال broadcast.

نلاحظ عدم وجود اختلاف بين لبنات هذه الفئة عند الكائن ولبناتها عند المنصة سوى في تعليمات when this sprite clicked فهي تعرف بهذا الشكل عند الكائن أما في المنصة فتصبح when stage clicked.

لبّات التّحكّم Control Blocks

لقيادة السيارة لا بدّ من تحكّمك بها، وللبرمجة باستعمال السكراتش لا بدّ من وجود التعليمات التي تتيح لك التّحكّم بالبرنامج الذي تريد تصميمه، تقوم لبّات هذه الفئة بتلك المهمة.



سنتعرّف تعليمة الانتظار:



من الواضح أن هذه التعليمة تجعل المترجم يتوقف عندها بحسب المدة التي حددها مصمم المشروع حيث تتعرّف على القيم بالثواني.

لننتقل الآن إلى الحلقات، ماذا لو أردت تكرار فعل معين يصدر عن كائنك 15 مرة؟ أو أردت تكراره طالما لم يُضغط على زر الإيقاف (الزر الأحمر)؟

هناك ما يدعى بالحلقات؛ وهي نوعان:

- حلقات منتهية؛ تكرّر التعليمات البرمجية التي بداخلها عدد من المرات يحددها المبرمج.
- حلقات غير منتهية؛ تكرّر التعليمات البرمجية التي بداخلها طالما لم يُضغط على زر إيقاف البرنامج (الزر الأحمر).



يبين المثال التالي حلقة منتهية، فعند الضغط على العلم الأخضر يرسل الكائن رسالة وينتظر لمدة ثانيتين، يكرر العمليتين السابقتين عشرة مرات.



أما المثال التالي فيبيّن حلقة غير منتهية، وهنا عند الضغط على العلم الأخضر تكرر الحلقة التعليمات التي بداخلها (إرسال رسالة والانتظار لمدة ثانيتين) إلى أن يُضغط على زر إيقاف البرنامج (الزر الأحمر).



كل خطوة تقوم بها في حياتك اليومية هي جزء من خوارزمية، والذهاب لشراء قطعة قماش هو خوارزمية أيضاً تبدأ بالخروج من المنزل وتنتهي بالعودة إليه وبينهما الذهاب إلى بائع القماش ، لكن ماذا ستفعل إذا كان مغلقاً متجر القماش؟ لابدّ من ذهابك إلى متجر آخر. أي أن إغلاق المتجر غيّر مسار خوارزمتك. فأصبح هناك شرط << إذا كان المتجر مفتوح ستشتري قطعة القماش وتعود إلى المنزل أما إذا كان مغلقاً فستذهب إلى متجر آخر. كذلك لدينا شروط في جميع الخوارزميات، وبما أن لبنات السكراتش المبرمجة من قبل المصمم هي عبارة عن مجموعة من الخوارزميات فلا بدّ من وجود لبنات للشروط، لنعرّف عليها.



تعبّر هذه التعليمة عن الشرط. يوضع الشرط في الفراغ بين if و then فإذا كان هذا الشرط محقق تنفذ التعليمات التي بداخلها.

الآن كيف سيصبح مسار البرنامج إذا لم يتحقق الشرط؟



ببساطة، سيتابع المترجم تنفيذ التعليمات التي تلي الشرط وكأن التعليمات الموجودة داخل الـ if غير موجودة.

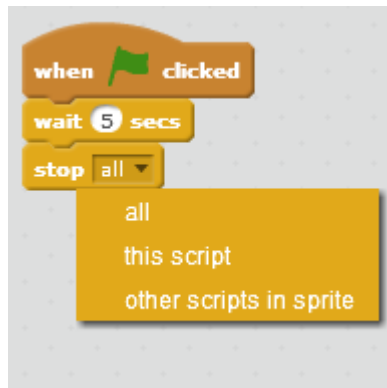
لكن يمكنك باستخدام التعليمة if/else تغيير مسار البرنامج إلى طريق آخر تحدده له؛ إذا كان الشرط محقق تنفذ التعليمات الموجودة داخل if أما إذا لم يكن محققاً ستنفذ التعليمات الموجودة داخل else.



هذه التعليمة تجعل كائنك ينتظر حتى يتحقق الشرط الموضوع في الفراغ.



بينما هذه التعليمة تجعل الكائن يكرّر التعليمات التي بداخلها حتى يتحقق الشرط الموضوع في الفراغ.



- إذا أردت إيقاف التعليمات البرمجية في مشروعك فعليك باستخدام:
- Stop all : لجميع الكائنات بما فيها التعليمات البرمجية للمنصة أيضاً.
 - Stop this script : إيقاف الكائن ذاته.
 - Stop other scripts in sprite : الكدسات البرمجية الأخرى (المغايرة للكدسة الموجودة فيها تعليمية الإيقاف).
- نلاحظ عدم القدرة على وضع تعليمات بعد تعليمة stop في حالتي stop all و stop this script لأن عندما يصل المترجم إليها يوقف قراءة التعليمات ولا ينفذ مايليها.

لننتقل الآن إلى القسم الأخير من تعليمات التحكم:



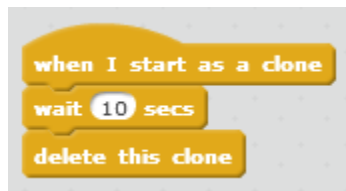
تقوم هذه التعليمات بنسخ الكائن بنفس موقعه وبنفس مظهره ودون إدراجه بين الكائنات، أو بنسخ كائن آخر تقوم أنت بتحديد من خلال الضغط على السهم. لكن كيف ستتم برمجته بعد نسخه؟

when I start as a clone

تمكّنك التعليمات الآتية من برمجة الكائن المنسوخ وذلك بوصل التعليمات البرمجية بها مثل أي تعليمات من تعليمات الـ Events اللواتي يأخذن شكل القمّة.

delete this clone

عندما ينتهي الكائن المنسوخ من تأدية وظيفته، بإمكانك حذفه من خلال تعليمات delete this course.



نلاحظ عدم وجود اختلاف بين لبنات هذه الفئة عند الكائن ولبناتها عند المنصة سوى في تعليمات النسخ. فلا يمكن للمنصة نسخ نفسها، لكن باستطاعتها نسخ أي كائن آخر.

لبّات الحركة Motion blocks

لإضافة الحيوية إلى مشروعك في السكراتش لا بدّ من أن تحرّك كائناتك بطريقة ما تختارها بما يناسب المشروع، باستخدام لبّات هذه الفئة يمكنك أن تقوم بذلك وبطرق مختلفة.

هناك عدّة طرق لتحريك الكائنات في السكراتش:

- حركة أفقية.
- حركة عمودية.
- حركة عشوائية.

وإضافة إلى ما سبق، يمكن للكائنات تتبّع مؤشر الفأرة أو تتبّع كائن آخر.

كما بيّنا سابقاً أبعاد منصّة السكراتش:

(x:240,-240 , y:180,-180)



لنبيّن الطريقة التي تتيح للكائن بالتحرك على محور الفواصل:



تعليمية الـ move تمكّن كائناتك من التحرك على محور الفواصل وإذا أردت عكس جهة الحركة فاضرب القيمة الموضوعية بالفراغ بـ (-1).

الآن ماذا تفعل إذا كانت حركة الكائن تتطلب الدوران؟



يمكنك تدوير الكائن عن طريق هذه تعليمية، وفي الواقع هي نوعان؛ النوع الأول يجعل الكائن يتحرك بالاتجاه المباشر (عكس اتجاه حركة عقارب الساعة) والنوع الثاني؛ يجعله يتحرك بالاتجاه غير المباشر (باتجاه حركة عقارب الساعة).



الآن بعدما تعرّفنا على كيفية تدوير الكائن بعدد من الدرجات التي يحددها المبرمج. ماذا تفعل إذا أردت أن يكون اتجاه الكائن للأعلى أو الأسفل أو اليمين أو اليسار وبدقة ؟

point in direction

(0) up

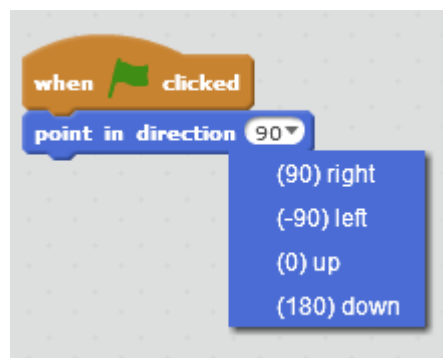
(-90) left



(90) right

(180) down

توضّح الصورة أعلاه الاتجاهات الأربعة للكائن، وللإجابة عن السؤال السابق نستعين بتعليمة الـ point in direction.



يمكنك من خلال الضغط على السهم أن تحدد الاتجاه المناسب للكائن، وبذلك يصبح اتجاهه هو الاتجاه الذي حددته بغض النظر عن اتجاهه السابق . أي : إذا كان اتجاه الكائن (180) down وقمت بتحديد الاتجاه الجديد وهو (90) right فإن اتجاه الكائن يصبح مباشرة (90) right.

بعدما تعلّمنا كيفية تغيير اتجاه كائن ما علينا تعلّم كيفية إظهار القيمة العددية لاتجاهه على المنصّة، للقيام بذلك نبحث في لبنات الحركة Motion Blocks عن تعليمة الـ direction ونضغط على المربع الذي بجانبها كما موضّح في الصورة أدناه :



الآن إذا أردت أن تجعل كائن ما يتجه نحو كائن آخر، ما عليك إلا باستعمال تعليمة point towards.



بإمكانك تحديد إما مؤشر الفأرة أو إحدى الكائنات المدرجة على المنصّة.



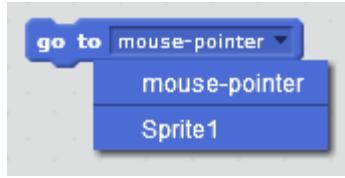
بعد تعلّم المهارات السابقة أصبح بإمكانك تحريك كائناتك بشكل أفقي وتغيير اتجاهها بعدّة طرق، والآن سنستعرض كيفية تحديد نقطة لكائن ما لنجعله يذهب إليها فور وصول المترجم إلى تلك التعليمة وهي go to x: __ y: __.



نتيح لك هذه التعليمة تحديد نقطة تجعل الكائن يذهب إليها عن طريق إعطاء إحداثياتها على محوري الفواصل والترتيب بشرط أن تكون هذه النقطة تقع في حدود منصّة السكراتش.



أمّا إذا كان مشروعك لا يحتاج إلى تحديد نقطة وحسب بل إلى الذهاب لكائن ما أو تتبّعه أو حتى تتبّع مؤشر الفأرة فعليك باللجوء إلى تعليمة go to: __.



كما ذكرنا سابقاً نقوم باختيار إما مؤشر الفأرة أو الكائن المراد الذهاب إليه عن طريق السهم.

هل فكرت مسبقاً في جعل كائنك يذهب إلى نقطة تحددها له بحيث ترى الطريق التي يسلكها أثناء ذهابه إليها؟



لابد أنك تفكر في تعليمة `go to x: __ y: __` وهذا تفكير منطقي، لكن تعليمة الـ `go to x: __ y: __` تنقل الكائن إلى النقطة المطلوبة فور وصول المترجم إليها أما التعليمة البرمجية التي سنتعرفها الآن هي: تعليمة `glide __ secs to x: __ y: __` التي تجعل الكائن يتزحلق إلى النقطة المطلوبة خلال فاصل زمني تحدده لها والذي بدوره يقوم بتحديد سرعة الكائن، ونلاحظ هنا وجود تناسب عكسي بين الزمن والسرعة (كلما زاد الزمن نقصت السرعة).



كما وضحنا سابقاً الفرق بين `glide __ secs to x: __ y: __` و `go to x: __ y: __` كذلك هو الفرق بين تعليمتي `change x by __` و `set x to __` فعند اختيار الأولى `change x by` نرى التغير الذي يطرأ على موقع الكائن بشكل تدريجي أما في التعليمة الثانية `set` فإنها تجعل إحداثيات `x` للكائن هي القيمة المعطاة فور وصول المترجم إليها.

كما يجب التوضيح أن تعليمة `change x by __` تحرك الكائن نحو اليمين إذا كانت القيمة المحددة من قبل المبرمج موجبة وتحركه نحو اليسار إذا كانت تلك القيمة سالبة.



عندما تريد تحريك كائناتك على محور الترتيب فيمكنك (بنفس الطريقة الموضحة سابقاً في تعليمتي
 __ change x by و __ set x to) استخدام تعليمتي __ change y by و __ set y to حيث
 __ change y by تحرك الكائن نحو الأعلى إذا كانت القيمة المحددة من قبل المبرمج موجبة وتحركه نحو
 الأسفل إذا كانت القيمة سالبة.



عندما تعرّفت على تعليمات الحركة هل تساءلت عن سلوك الكائن عند اصطدامه بحافة المنصة؟

if on edge, bounce

لابدّ أنك قد فعلت، لذلك أتاحت لنا لغة البرمجة الرسومية السكراتش تعليمة وظيفتها جعل الكائن يرتدّ عند
 وصوله إلى الحافة وهي if on edge, bounce.

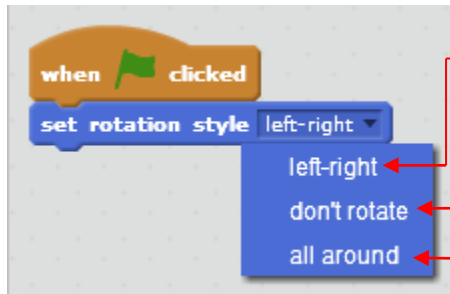


لكن إذا كان اتجاه الكائن 90 وارتدّ، هل يحافظ على هذا الاتجاه بعد ارتداده؟ أم أن اتّجاهه يصبح 90-؟

set rotation style left-right

للإجابة عن السؤال السابق بإمكانك أن تقوم بالتجريب. بعد هذا الارتداد يتغيّر اتجاه الكائن ويصبح 90- فإذا
 لم تكن ترغب في دوران الكائن بشكل حرّ في كل الاتجاهات، بإمكانك أن تختار حدود حرية دوران الكائن
 من التعليمة التالية: __ set rotation style.

لنمط الدوران في هذه التعليمة ثلاث حالات:



- أن يكون له اتجاهين (يمين ويسار).
- ألا يلتف مطلقاً ولا يغير اتجاهه.
- أن يكون قادر على الدوران في جميع الاتجاهات (حرية حركته غير مقيدة).

كما ذكرنا سابقاً أنه بإمكاننا إظهار القيمة العددية لاتجاه الكائن على المنصة ، كذلك بإمكانك إظهار فاصلة الكائن أو ترتيبه. ولا يقتصر الأمر على إظهاره على المنصة بل يمكنك الاستفادة من هذه القيمة أيضاً كأن تخزنها في متحول أو تجري عليها عمليات حسابية بما يخدم مشروعك.



بإمكانك إظهار القيمة على المنصة أو إبطال إظهارها من خلال الضغط على المربع كما هو موضح في الصورة أعلاه.

- x position : يحوي قيمة جبرية، وهي x الكائن (فاصلة الكائن).
- y position : يحوي قيمة جبرية، وهي y الكائن (ترتيب الكائن).
- direction : يحوي قيمة جبرية، وهي اتجاه الكائن.

نلاحظ عدم ظهور تعليمات هذه الفئة عند برمجة المنصة لأن منصة السكراتش ثابتة ولا تتحرك.

لبّات المظاهر Looks blocks

لا شك أن المظهر الجيد للكائنات في الألعاب والمشاريع التفاعلية يضيف رونقاً جميلاً، ولبنات هذه الفئة تقوم بتلك المهمة وهي التحكم بمظهر الكائنات والمنصّة.



سنعرّف هاتين التعليمتين البرمجتين.



من الواضح أن هاتين التعليمتين تجعل الكائن يقول النص المكتوب في الفراغ بالشكل التالي.



لكن ماهو الفرق بين say __ for __ secs و say _ ؟

عند استعمال say _ لا تختفي الفقاعة التي يظهر بداخلها النص، ويتم تنفيذ التعليمات التي تلي هذه الـ say أثناء تنفيذ الـ say.

ملاحظة: يمكن إخفاء النص في تعليمة الـ say بوضع نص فارغ بتعليمة say _ أخرى تليها.

أما عند استعمال say __ for __ secs فستظهر الفقاعة التي يظهر بداخلها النص لمدة زمنية مساوية للمدة المحددة لها في فراغ الثواني، ولا يتم تنفيذ التعليمات التي تليها إلا عند اختفاء هذه الفقاعة وانتهاء المدة الزمنية المحددة.



الكودستان البرمجتان السابقتان متكافئتان؛ تنفذان التنفيذ ذاته.



التعليمتان أعلاه مشابهتان لـ say __ for __ secs و say __ والفرق بينهما نفس الفرق بين say __ for __ secs و say __. ولكن الاختلاف البسيط بين say و think هو كيفية ظهور الفقاعة.



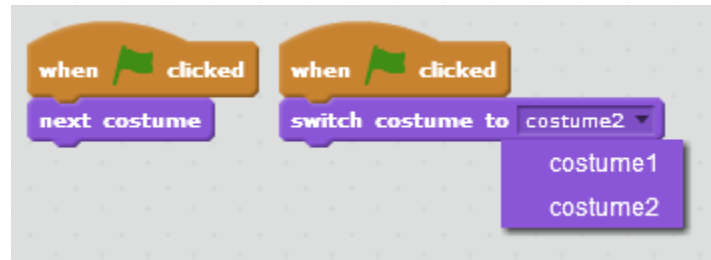
التعليمتان hide و show تقومان بإظهار الكائن و إخفائه. يوضح المثال الآتي تكرار عشر مرات لإظهار الكائن والانتظار ثانية والعودة لإخفائه من جديد.



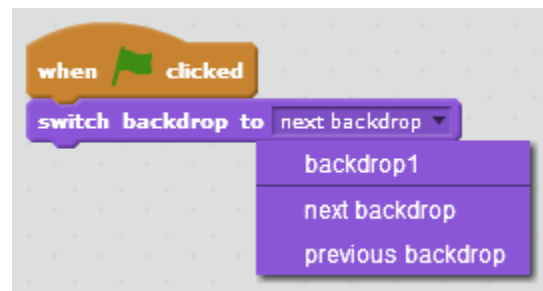
ملاحظة: في نهاية هذا البرنامج سيكون الكائن مخفٍ لأن التكرار توقف عندما كان الكائن بصورته غير المرئية (المخفية).



هناك طريقتين لتغيير مظهر كائن ما، إما أن تختار التعليمة `switch costume to` _ التي تحدد فيها اسم المظهر المراد الحصول عليه أو أن تختار تعليمة `next costum` التي تعطيك المظهر التالي لكائنك دون أن تحدد له المظهر الذي تريده.



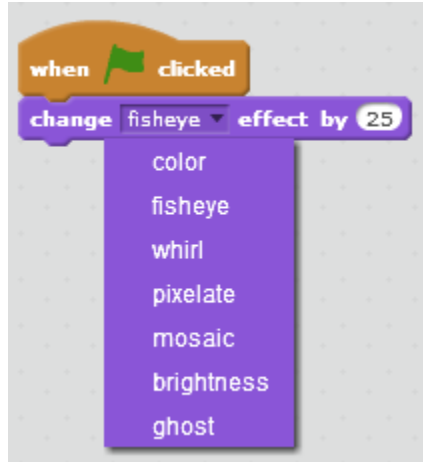
بنفس الطريقة السابقة بإمكانك تغيير الخلفية من تعليمات كائن آخر كما موضّح بالصورة أدناه:



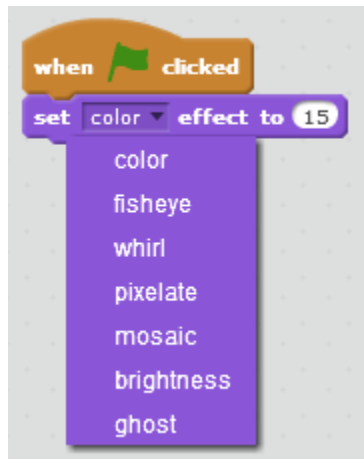
بإمكانك اختيار اسم الخلفية أو الخلفية التالية أو السابقة من خلال الضغط على السهم.



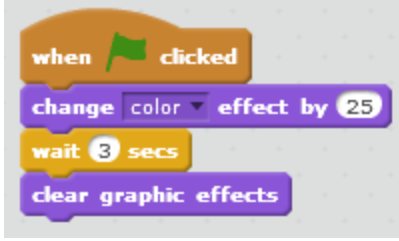
نتيح لك بيئة السكراتش تغيير المؤثرات (اللونية- الضوئية... إلخ) لكائناتك وحتى للمنصّة، ويمكنك التحكم بمقدار التغيير حيث يتم ذلك من خلال التعليمة الآتية:



بإمكانك أيضاً جعل قيمة تلك المؤثرات ثابتة؛ تصبح قيمة المؤثر مساوية للقيمة المطلوبة من قبل المصمم فور وصول المترجم إلى التعليمة الموضّحة أدناه في الصورة:



في حال تغييرك لمؤثرات كائنك، قد تحتاج في مرحلة من مراحل مشروعك إلى عودة كائنك إلى شكله الأصلي (الشكل الافتراضي الذي كان عليه قبل تطبيق المؤثرات)، لذلك أتاح لك السكراتش تعليمة `clear graphic effects`.



تقوم تعليمية clear graphic effects بإزالة جميع المؤثرات وتعيد الكائن لشكله الأولي، ففي المثال السابق يتغير التأثير اللوني بمقدار 25 و يبقى على هذه الحالة ثلاث ثوانٍ ثم يزيل المؤثرات ويتوقف البرنامج.



تغيير حجم كائنك يضيفي على مشروعاتك تفاعل، فبإمكانك تغيير حجم الكائن (تكبير أو تصغير) بشكل تدريجي بواسطة التعليمية change size by __ حيث يصبح حجم الكائن الجديد هو حجم الكائن السابق + القيمة الموضوعه في الفراغ. وبإمكانك أيضاً تحديد حجم الكائن من خلال وضع القيمة في الفراغ المخصص لها في تعليمية set size to __ فيصبح حجم الكائن مساوٍ للقيمة التي وضعتها بغض النظر عن حجمه السابق.



إذا كان مشروعك يحوي العديد من الكائنات، وتريد ترتيب الكائنات وفق طبقات؛ كأن تصمم ساعة حائط تحوي عقارب، فلا بدّ من ظهور العقارب فوق الساعة. لذلك نلجأ إلى تعليمية go to front.

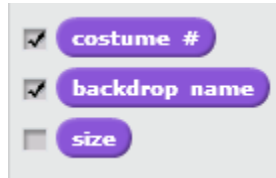


وبعملية معاكسة للبنة السابقة، يمكننا أن نجعل الكائن يختبئ وراء عدد من الكائنات تحدده له.



يوضح هذا المثال اللبنتين السابقتين؛ فعند الضغط على العلم الأخضر يذهب الكائن إلى الأمام ويتحرك عشر خطوات ثم يذهب خلف طبقة واحدة ويتوقف البرنامج.

نلاحظ في نهاية لبنات المظاهر وجود التالي:



بإمكانك إظهار القيمة على المنصة أو إبطال إظهارها من خلال الضغط على المربع كما هو موضح في الصورة أعلاه.

- costume # : الرقم التسلسلي للكائن.
- backdrop name : اسم الخلفية الحالية.
- size : حجم الكائن.

نلاحظ عدم وجود جميع التعليمات البرمجية الخاصة بالكائن في التعليمات البرمجية الخاصة بالمنصة كتعليمة تغيير الحجم لأن حجم المنصة ثابت ولا يمكن تغييره أو تعليمة go to front لأن المنصة دائماً في الطبقة الأخيرة.

لبنات العمليات Operators Blocks

تتيح لك لبنات هذه الفئة بالقيام بالعمليات الحسابية والمنطقية التي سنشرحها كل على حدى مع أمثلة موضحة. إذا تأملنا لبنات هذه الفئة نلاحظ عدم إمكانية وضع أي منها بشكل منفصل في الكود البرمجي، وإنما يجب وضعها ضمن say أو أي لبنة أخرى تؤدي الغرض المطلوب.

نلاحظ العمليات الحسابية الأربعة التالية، والتي تحوي كل منها على فراغين يمكننا ملؤهما بالقيم المناسبة للحصول على النتيجة المطلوبة. خرج (Output) تلك اللبنات هو قيمة جبرية.



يوضح المثال التالي استخدام واحدة من اللبنات الأربعة السابقة حيث يقوم الكائن بقول الناتج 15 لمدة ثانيتين.



في كثير من الأحيان يتطلب البرنامج المراد تصميمه الحصول على قيمة عشوائية، فنلجأ إلى المترجم لتوليدها بحيث نحدد المجال الذي نريد تلك القيمة أن تتراوح ضمنه بإعطائه بداية (قيمة صغرى) ونهاية (قيمة كبرى). يمكننا التعليمة `pick random _ to _` من القيام بذلك.

لنمعن النظر في المثال الآتي، ولنكتشف ماذا ينفذ؟



عند النقر على العلم الأخضر يقول الكائن رقم عشوائي بين الـ10 والـ100 لمدة ثانيتين. يكرر العملية السابقة خمس مرات بسبب وجود حلقة التكرار.

سؤال: في المثال السابق هل القيمتان 10 و 100 تنتميان إلى المجال المعطى؟ بصيغة أخرى: هل من الممكن أن يقول الكائن 10 أو 100 ؟

الجواب هو نعم، القيمتين 10 و 100 تنتميان إلى المجال.

نلاحظ عمليات المقارنة الثلاثة التالية والتي تحوي كل منهنّ على فراغين يمكننا ملؤهما بالقيم المناسبة للحصول على إحدى النتيجتين إما true أو false. حيث خرج تلك اللبئات هو قيمة منطقية (بوليانية).

ملاحظة: القيمة المنطقية تكون إما true أو false.



في المثال الآتي يقول الكائن true لأن الـ $4 < 5$ فالعملية محققة.



لنتأمل المثال التالي، هل يمكنك معرفة ماذا سيقول الكائن؟



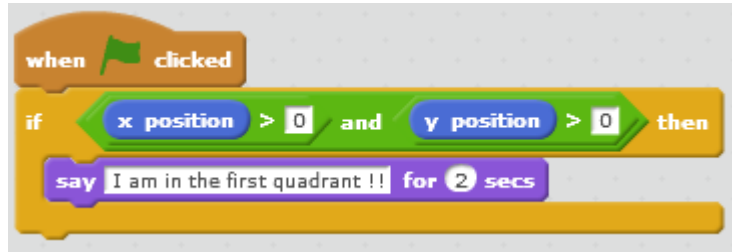
بالتأكيد سيقول الكائن false لأن الرقم 6 غير مساو للرقم 16 وبالتالي المساواة غير محققة.

باستطاعتك مزج عدة مخارج منطقية في مخرج واحد باستعمال مفاهيم البوابات المنطقية , AND , OR , NOT. مثلاً: من الممكن أن تختبر إذا كانت قيمة ما أكبر من 10 وأصغر من 20 في آن واحد، أو أن تختبر إذا كانت قيمة ما مساوية للعدد 5 أو 7. بإمكانك أيضاً اختبار إذا كانت قيمة ما غير مساوية لقيمة أخرى؛ كأن تختبر هل فاصلة الكائن (x الكائن) غير مساوية للصفر.

and

AND: أنت ستذهب إلى السباحة إذا كان الطقس جميلاً وإذا كان لديك المال الكافي؛ أنت لن تذهب إلا بتحقق الشرطين السابقين. كذلك تعمل البوابة المنطقية AND، حيث تعطي الخرج true في حال كانت جميع المداخل true وتعطي الخرج false في حال كان أحد المداخل false على الأقل (واحدة منهما false أو الاثنين معاً في حال وجود مدخلين فقط).

المثال الآتي يختبر إذا كان الكائن في الربع الأول من المنصة عن طريق اختبار شرطين هما (الفواصل <0 و الترتيب <0) ولا يعتبر الشرط محقق إلا بتحقق الشرطين معاً.



or

OR: أنت ذاهب إلى الحديقة المجاورة لمنزلك في ساعة متأخرة، ولهذه الحديقة بابان، يكفي أن يكون أحدهما مفتوح لتتمكن من الدخول، كذلك تمثل البوابة المنطقية OR، حيث تعطي الخرج true في حال تحقق أحد الشروط فقط. كما تعطي قيمة true أيضاً في حال تحقق أكثر من شرط، بينما تعطي قيمة false إذا كانت جميع الشروط غير محققة.

المثال الآتي يختبر الشرطين (الفواصل <0 و الترتيب <0) فيكفي تحقق أحدهما فقط ليدور الكائن 90 درجة.



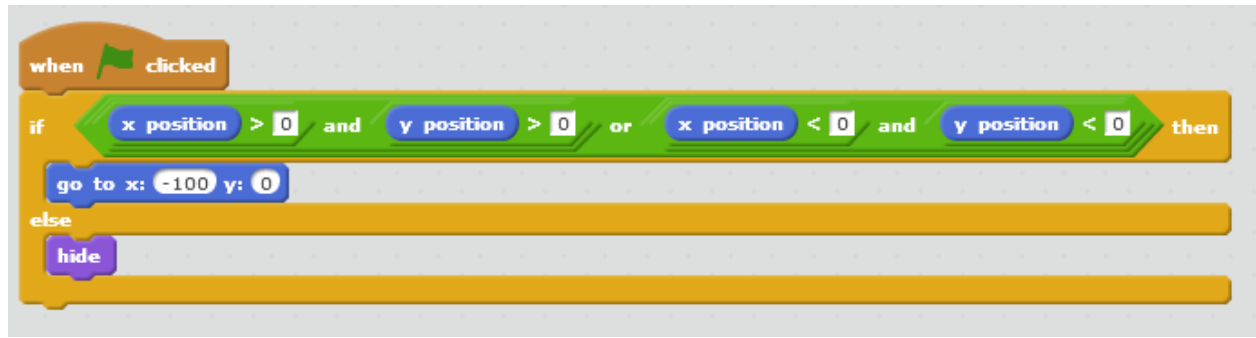
not

NOT: الطقس غائم وأنت ذاهب في نزهة على الدراجة لكنك قلق بشأن الطقس وهطول المطر، فأنت ستذهب على الدراجة في حال عدم هطول المطر. كذلك تعمل البوابة المنطقية NOT حيث تعطي الخرج true في حال عدم تحقق الشرط المعطى لها وتعطي قيمة false في حال تحققه.



يوضح الشكل مثال عن البوابة NOT فهو يقوم باختبار فاصلة الكائن إذا كانت مساوية للقيمة 10؛ وإذا لم تكن مساوية لها يجعل فاصلته 10.

يمكننا دمج أكثر من شرط في أكثر من بوابة كما في المثال الآتي:



فهو اختبار للكائن إذا كان في الربع الأول أو الربع الرابع وإذا تحقق أحد الشرطين السابقين يجعل الكائن ينتقل إلى النقطة التي فاصلتها -100 وترتيبها 0، أما إذا لم يتحقق أي منهما يختفي الكائن.

ملاحظة: يجب لفت الانتباه أن لكل بوابة منطقية شكل مميز وجدول يصف سلوكها ويوضح قيمة الخرج لكل دخل منطقي محتمل لها، يسمى هذا الجدول بـ **جدول الحقيقة**.

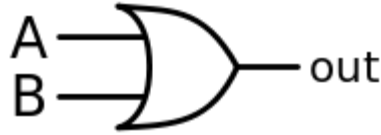


AND: بفرض A الدخل الأول و B الدخل الثاني.

دخل		خرج
A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

جدول الحقيقة للبوابة AND:

نعبّر عن هذه البوابة بالعلاقة: $A \cdot B$

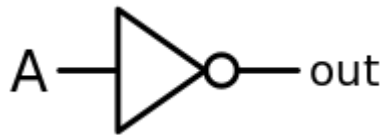


OR: بفرض A الدخل الأول و B الدخل الثاني.

جدول الحقيقة للبوابه OR:

دخل		خرج
A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

نعبّر عن هذه البوابه بالعباره: $A + B$



NOT: بفرض A الدخل.

جدول الحقيقة للبوابه NOT:

دخل	خرج
A	not A
1	0
0	1

نعبّر عن هذه البوابه بالعباره: \bar{A}



في الكثير من الألعاب يعرض لك في نهاية اللعبة بعض المعلومات، مثلاً: your score is 1625. وفي السكراتش إذا أمعنا النظر في أسفل المنصة نجد إحداثيات مؤشر الفأرة على محوري الفواصل والترتيب عند تحرّكه في المنصة.

سنقوم بتصميم مشروع يجعل الكائن يتبع مؤشر الفأرة ويقول إحداثياته بالشكل الآتي:

سلسلة محرفية: وهي مجموعة من الأحرف والرموز

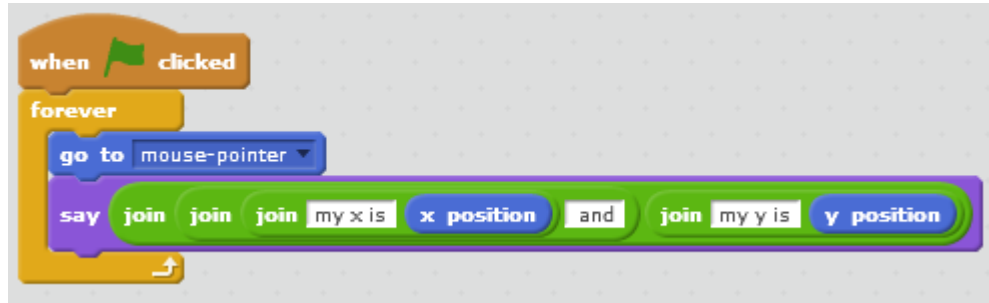
my x is __ and my y is __

لنفكر معاً كيف يمكن دمج السلسلتين المحرفيتين السابقتين مع قيمتي فاصلة الكائن وترتيبه.

تمكنا اللبنة join من القيام بذلك.



لكن كيف يمكننا دمج اللبنتين السابقتين؟ لنتبع الخطوات التالية:



بفرض أن الكائن في نقطة إحداثياتها (-20,40) سيقول: my x is -20 and my y is 40 نلاحظ وجود مسافة بعد is وقبل وبعد and في المثال السابق، لذلك لا تظهر القيم -20 و 40 موصولة بما قبلها. لماذا لا تعدّل المثال السابق وتنشئ نصوص خاصة بك أو تتحقق بنفسك كيف يبدو النص في حال عدم وجود المسافات؟

length of

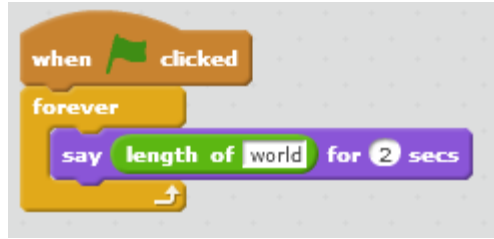
لنتعرّف الآن لبنة length of، باعتبار أي نص هو سلسلة محرفية، تمكننا هذه اللبنة من الوصول إلى أي حرف من تلك السلسلة، فقط بتحديد رقمه (باعتبار الحرف الأول رقمه 1).

W	O	R	L	D
1	2	3	4	5

فإذا أردنا عرض حرف الـ l من كلمة world ننفذ الكود التالي:



أما إذا أردنا معرفة طول تلك السلسلة فنقوم بما يلي:



ملاحظة: يمكن اعتبار الأعداد سلاسل محرفية باعتبار أن كل رقم هو حرف، وبذلك يمكننا الوصول إلى أي جزء (منزلة) من العدد ويمكننا أيضاً معرفة طوله (عدد منازل) بنفس الطريقة السابقة.



لمعرفة باقي قسمة عدد على آخر نلجأ إلى لبنة `_ mod _` ، فمثلاً إذا أردنا عرض قيمة باقي قسمة 16 على 3 فننفذ الكود التالي:



سيقول الكائن في هذا المثال 1 لمدة ثانيتين، لأن $16/3=5$ والباقي

1، حيث: $3*5=15$

ناتج ضرب $3*5$

$$15+1=16$$

الباقي



ولتقريب أي قيمة إلى أقرب جزء من عشرة (لتقريب العدد والتخلص من الفاصلة) نلجأ إلى لبنة `round_`، كما موضح في الشكل:



حيث في لبنة الـ `say` الأولى سيكون الخرج 10، أما في الثانية فسيكون 9.



هناك الكثير من العمليات الحسابية التي لم نذكرها، دُكر بعضها في هذه اللبنة، ونستخدمها كسائر لبنات هذه الفئة.

نزهة في عالم الرياضيات

سنقوم بشرح مبسط عن كل واحدة من تلك العمليات في هذه اللبنة.

- **abs** وهي اختصار لـ **absolute value** وتعني القيمة المطلقة. القيمة المطلقة لعدد ما هي المسافة بين هذا العدد والصفر. بمعنى آخر القيمة المطلقة لعدد موجب هي العدد نفسه والقيمة المطلقة لعدد سالب هي العدد السالب مضروباً بـ -1.

أمثلة:

$$|-5| = 5$$

$$|8| = 8$$

- **floor** تقرّب العدد المدخل لها إلى أقرب أصغر عدد صحيح.

أمثلة:

$$\lfloor 1.83 \rfloor = 1$$

$$\lfloor -5.1 \rfloor = -6$$

- **ceiling** تقرّب العدد المدخل لها إلى أقرب أكبر عدد صحيح لها.

أمثلة:

$$\lceil 3.14 \rceil = 4$$

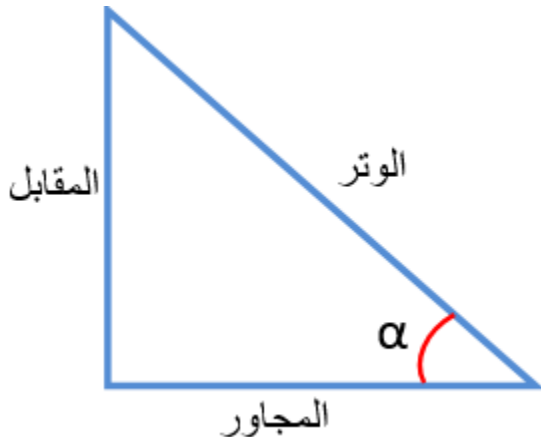
$$\lceil 7.9 \rceil = 8$$

- **sqrt** وهو الجذر التربيعي للعدد المدخل لها.

أمثلة:

$$\sqrt{64} = 8$$

$$\sqrt{36} = 6$$



- **$\sin \alpha$** وهو طول الضلع المقابل للزاوية α على طول الوتر في المثلث القائم، قيمته تتراوح بين 1 و -1.

- **$\cos \alpha$** وهو طول الضلع المجاور للزاوية α على طول الوتر في المثلث القائم، قيمته تتراوح بين 1 و -1.

- **$\tan \alpha$** وهو $\frac{\sin \alpha}{\cos \alpha}$ أو هو طول الضلع المقابل

للزاوية α على طول الضلع المجاور لها في المثلث القائم.

قيم الـ \sin والـ \cos لبعض الزوايا الشهيرة:

A	$\sin \alpha$	$\cos \alpha$
$30^\circ, \frac{\pi}{6} \text{ rad}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$
$60^\circ, \frac{\pi}{3} \text{ rad}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$
$45^\circ, \frac{\pi}{4} \text{ rad}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$

- $\arcsin x$ هو أي قياس الزاوية التي الـ \sin لها يساوي x .
- $\arccos x$ هو أي قياس الزاوية التي الـ \cos لها يساوي x .
- $\arctan x$ هو أي قياس الزاوية التي الـ \tan لها يساوي x .
- \ln هو اللوغاريتم الطبيعي؛ حيث لوغاريتم عدد بالنسبة للأساس هو الأس المرفوع على الأساس والذي سينتج ذلك العدد، فمثلاً يمكن أن نقول لوغاريتم X بالنسبة للأساس b هو y ونعبر عنها بالشكل التالي: $X=b^y$.
- \log هو اللوغاريتم العشري؛ لوغاريتم عدد ما بالنسبة للأساس 10. مثال: $\log_{10} 1000 = 3$.
- e^n يساوي تقريباً 2.718؛ حيث e^n هو e مضروب بنفسه n مرة. مثال: e^3 هو e مضروب بنفسه ثلاث مرات أي $e \cdot e \cdot e$ ويساوي تقريباً 20.086.
- 10^n ؛ حيث 10^n هي 10 مضروبة بنفسها n مرة. مثال: 10^6 هي $10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10$ ويساوي تقريباً 1.000.000.

بذلك نكون قد انتهينا من شرح جميع لبنات هذه الفئة مع الأمثلة مع ملاحظة عدم وجود اختلاف بين لبنات هذه الفئة عند الكائنات ولبناتها عند المنصّة.

لبينات التحسس Sensing Blocks

جميع الكائنات الحية تتفاعل مع الوسط المحيط بها، وكذلك كائنات السكراتش؛ فهي تتفاعل مع الكائنات الأخرى ومؤشر الفأرة وجميع عمليات الإدخال (صوت- فيديو- نص...) عن طريق لبيانات هذه الفئة .sensing blocks



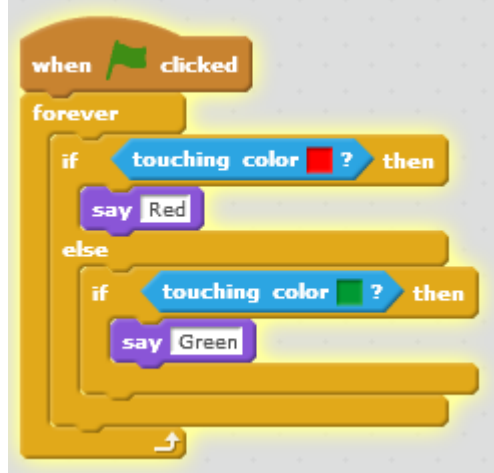
تقوم هذه اللبنة باختبار هل هذا الكائن يلامس كائن آخر أو مؤشر الفأرة أو حتى حافة المنصة، وتعطي قيم منطقية true او false (التي وضعناها في فصل blocks operator)، فإذا كان الكائن يلامس الكائن المحدد تعطي اللبنة قيمة true، أما إذا لم يكن يلامسه فتعطي قيمة false.



يوضح المثال السابق كيفية استخدام هذه اللبنة؛ فعند نقر العلم الأخضر يختبر الكائن دائماً، فإذا كان يلامس مؤشر الفأرة يدور خمس درجات بالاتجاه الموجب (عكس اتجاه دوران عقارب الساعة)، أما إذا لم يكن يلامسه فيدور خمس درجات بالاتجاه السالب.



تقوم هذه اللبنة باختبار هل هذا الكائن يلامس اللون المحدد له في المربع، وكذلك تعطي قيم منطقية true او false، فإذا كان الكائن يلامس اللون المحدد تعطي قيمة true، أما إذا لم يكن يلامسه فتعطي قيمة false.



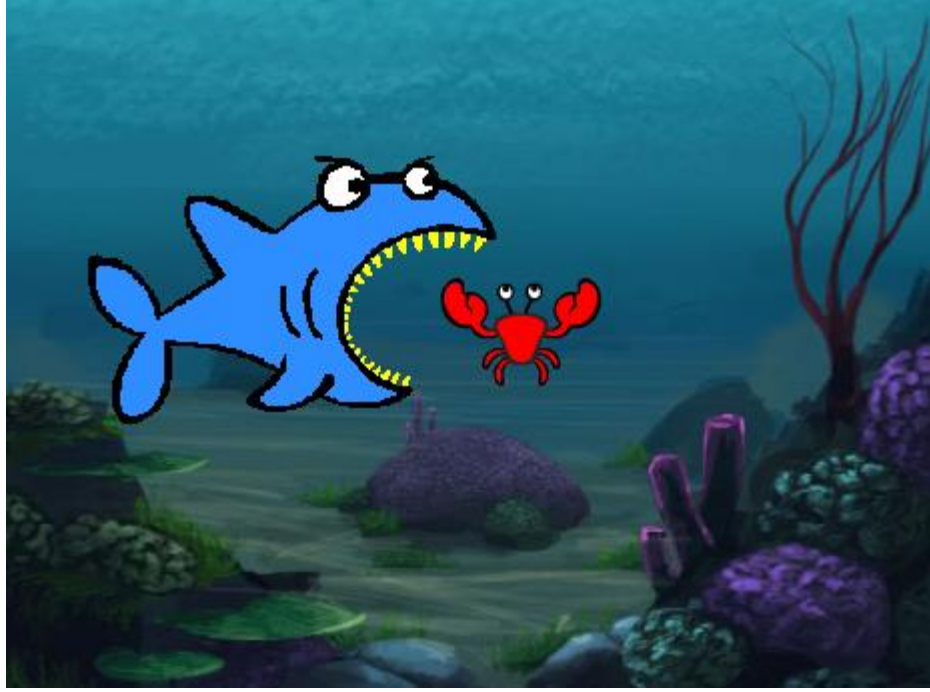
يوضح المثال السابق كيفية استخدام هذه اللبنة؛ فعند النقر على العلم الأخضر يختبر الكائن دائماً، فإذا كان يلامس اللون الأحمر يقول Red، أما إذا كان يلامس اللون الأخضر يقول Green.

ملاحظة: لتغيير اللون الموضوع في الفراغ نضغط على الفراغ مما يؤدي إلى تغيّر في شكل المؤشر ثم نحدد لون من المنصة وننقر بالمؤشر عليه، فيتغير اللون الموضوع في هذه اللبنة.



تقوم هذه اللبنة باختبار هل اللونان المحددان متلامسان، وكذلك تعطي هذه اللبنة قيم منطقية true او false، فإذا كان اللونان المحددان متلامسين تعطي اللبنة قيمة true، أما إذا لم يكونا متلامسين فتعطي قيمة false. تفيد هذه اللبنة في إعلامنا بتلامس جزء محدد من الكائن مع جزء محدد من كائن آخر أو تلامس جزء محدد من كائن مع كائن آخر والمثال التالي سيوضح ذلك.

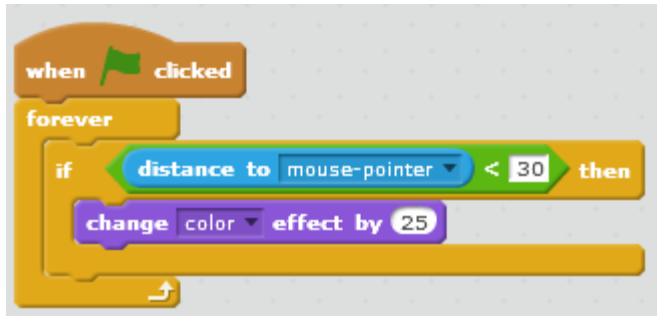




يبين المثال السابق كيفية استخدام هذه اللبنة؛ فعند النقر على العلم الأخضر يختبر الكائن دائما اذا كان السرطان يلامس اللون الأصفر (لون فم سمكة القرش)، فإذا كان يلامسه ستغير السمكة مظهرها مستعدةً لالتهام السرطان. نلاحظ أن السمكة لن تغير مظهرها عند تلامس السرطان مع أي نقطة منها مغايرة للون الأصفر (لون الفم)، وهذا ما يجعل من مشروعك مشروع منطقي، فليس من المعقول أن تلتهم السمكة السرطان إذا اصطدم بعينيها مثلاً.

distance to ▾

تقوم هذه اللبنة بقياس المسافة بين الكائن وكائن آخر أو مؤشر الفأرة أو حافة المنصة. تعطي هذه اللبنة قيم عددية (أقصى حد تعطيها 300).

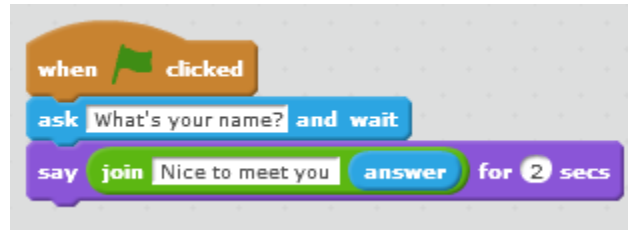


يوضح المثال استخدام هذه اللبنة؛ فعند النقر على العلم الأخضر يختبر الكائن، فإذا كانت المسافة بينه وبين مؤشر الفأرة أصغر من 30 يغير لونه بمقدار 25.

answer

ask and wait

تؤمن هذه اللبنة التفاعل الذي يتم بين النصوص التي يدخلها المستخدم وكائنات السكريبتش. كما تقوم هذه اللبنة بعرض النص المدخل إليها من قبل المبرمج على شكل فقاعة تخرج من الكائن، ولا يتحدد ظهورها بوقت؛ بل تبقى ظاهرة إلى أن يقوم المستخدم بإدخال نص أو إيقاف البرنامج. يرتبط وجودها بلبنة الجواب حيث يختزن النص المدخل من قبل المستخدم في هذا الجواب.

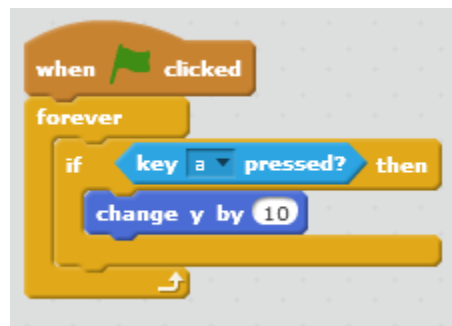


يوضح المثال السابق استخدام هاتين اللبنتين، حيث يقوم الكائن بسؤالك What is your name ثم يقول لك لمدة ثانيتين X Nice to meet you باعتبار الاسم المدخل هو X. نلاحظ استخدام لبنة join (التي تعرفنا عليها في فصل operators blocks) في دمج الاسم المدخل والنص المراد عرضه.



key space pressed?

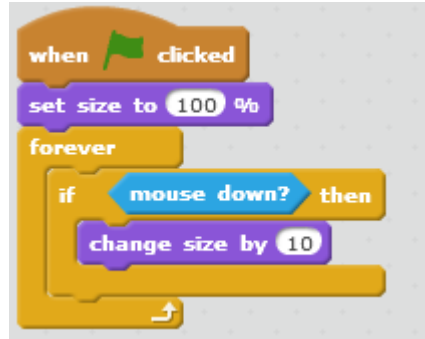
تقوم هذه اللبنة باختبار هل الزر المحدد مضغوط، تعطي هذه اللبنة قيمة منطقية true أو false، فإذا كان هذا الزر مضغوط تعطي اللبنة قيمة true، أما إذا لم يكن مضغوط فتعطي قيمة false.



يوضح المثال السابق كيفية استخدام هذه اللبنة؛ فعند النقر على العلم الأخضر يختبر هل الزر المحدد (a) مضغوط، إذا كان مضغوط يزيد إحداثياته على محور الترتيب بمقدار 10 (يصعد الكائن نحو الأعلى بمقدار 10).

mouse down?

تقوم هذه اللبنة باختبار هل زر الفأرة مضغوط، تعطي هذه اللبنة قيم منطقية true أو false، فإذا كان زر الفأرة مضغوط تعطي اللبنة قيمة true، أما إذا لم يكن مضغوط فتعطي قيمة false.



يبين المثال السابق كيفية استخدام هذه اللبنة؛ فعند النقر على العلم الأخضر يعطي قيمة ابتدائية لحجم الكائن، ثم يختبر دائماً إذا كان زر الفأرة مضغوط، فإذا كان مضغوط يزداد حجم الكائن بمقدار 10؛ أي يزداد الحجم بمقدار 10 طالما زر الفأرة مضغوط.

mouse x

تخزن هذه اللبنة قيمة إحداثيات مؤشر الفأرة على محور الفواصل. في الكثير من الألعاب نحتاج لتحريك الكائن على محور الفواصل فقط تبعاً لحركة مؤشر الفأرة، فإذا أردنا تطبيق ذلك في مشرونا نعطي قيمة ثابتة لإحداثيات الكائن على محور الترتيب ونجعل قيمة الإحداثيات على محور الفواصل متغيرة تبعاً لقيمة إحداثيات مؤشر الفأرة على هذا المحور باستخدام هذه اللبنة كما موضح في المثال التالي:



mouse y

كذلك تخزن هذه اللبنة قيمة إحداثيات مؤشر الفأرة على محور الترتيب. في الكثير من الألعاب نحتاج لتحريك الكائن على محور الترتيب فقط تبعاً لحركة مؤشر الفأرة، فإذا أردنا تطبيق ذلك في مشرونا نعطي

قيمة لإحداثيات محور الفواصل ونجعل قيمة محور الترتيب متغيرة تبعا لقيمة إحداثيات مؤشر الفأرة على هذا المحور باستخدام هذه اللبنة كما موضح في المثال التالي.



loudness

تقوم هذه اللبنة بقياس شدة الصوت الذي يتحسسه الميكرفون الموصول بالحاسوب، تعطي هذه اللبنة قيم عددية تتراوح بين (1-100). لإظهار شدة الصوت على المنصة نقوم بالضغط على المربع لتظهر إشارة



الـ ✓ كما في الشكل التالي :



يوضح المثال استخدام هذه اللبنة؛ فعند النقر على العلم الأخضر يختبر، فإذا كانت شدة الصوت أعلى من 50 يتحرك الكائن درجة واحد بالاتجاه السالب (عكس جهة دوران عقارب الساعة).

video motion on this sprite

تقوم هذه اللبنة بقياس مقدار الحركة واتجاه الحركة في الفيديو المصوّر من قبل كاميرا الويب. نميز في هذه اللبنة أربعة أنواع :

video motion on this sprite

تستخدم هذه اللبنة لقياس شدة الحركة في الفيديو تحت الكائن المحدد.

video motion on Stage

تستخدم هذه اللبنة لقياس شدة الحركة في الفيديو على كل المنصة.

video direction on this sprite

تستخدم هذه اللبنة لقياس اتجاه الحركة في الفيديو تحت الكائن المحدد.

video direction on Stage

تستخدم هذه اللبنة لقياس اتجاه الحركة في الفيديو على كل المنصة.

turn video on

تشغل هذه اللبنة كاميرا الويب. يتم التشغيل باختيار الخيار on ويتم الإطفاء باختيار الخيار off ونلاحظ وجود خيار ثالث on-flipped حيث يقوم بعكس صورة الفيديو الملتقطة من كاميرا الويب.

set video transparency to 50 %

تستخدم هذه اللبنة لتحديد شفافية الفيديو (من 1 إلى 100%).

timer

تعد هذه اللبنة بمثابة مؤقت؛ حيث يبدأ بعد الثواني واختزانها من لحظة تشغيل البرنامج (الضغط على العلم الأخضر) وحتى إيقافه أو إعادة ضبط المؤقت (جعله يساوي الصفر).

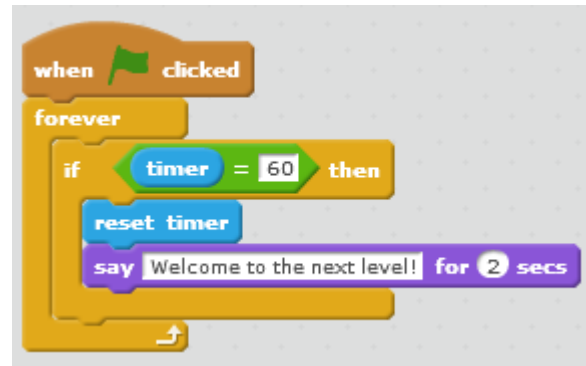


يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر (نعلم أن المؤقت سيبدأ عد الثواني)، يختبر دائماً إذا كان عدد الثواني مساوٍ لـ 30، فإذا كان مساوٍ لها يقول الكائن Game Over! كما هو موضح بالشكل.

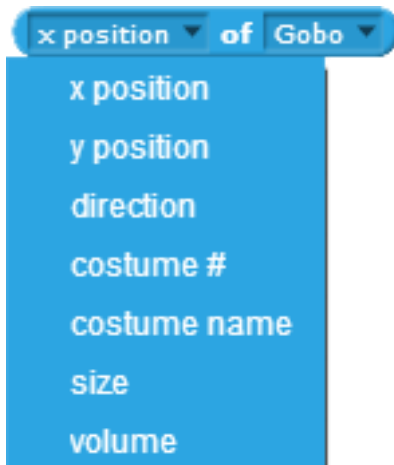


reset timer

وضحنا أثناء شرح اللبنة السابقة timer بأنه يبدأ بعد الثواني واختزانها من لحظة تشغيل البرنامج وحتى إعادة ضبط المؤقت (جعله يساوي الصفر)، لكن كيف تتم إعادة ضبطه؟ بكل بساطة لنتم إعادة ضبط المؤقت باستخدام هذه اللبنة، فالمترجم عندما يجدها في البرنامج يجعل المؤقت مساوٍ لـ 0.



يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يختبر، هل قيمة المؤقت مساوية لـ 60، إذا كانت مساوية لها يعيد ضبطه (يجعله يساوي الصفر) ويقول Welcome to the next level! لمدة ثانيتين.



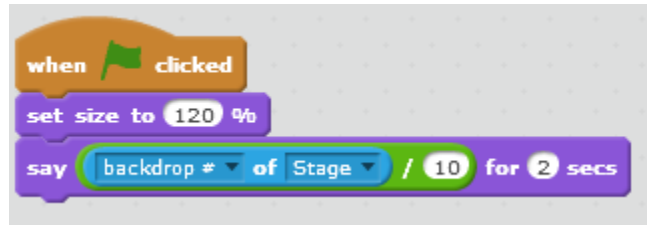
تخبرنا هذه اللبنة بمعلومات عن الكائن أو المنصة، حيث تعيد قيم عددية وسلاسل محرفية.

x position: يقوم بإعطائنا قيمة عددية x الكائن (إحداثيات الكائن على محور الفواصل).

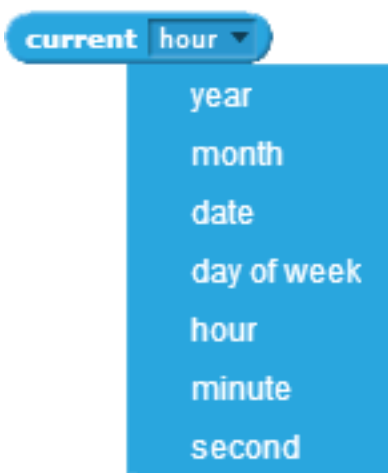
y position: يقوم بإعطائنا قيمة عددية y الكائن (إحداثيات الكائن على محور الترتيب).
 direction: يقوم بإعطائنا قيمة عددية (اتجاه الكائن).
 costume #: يقوم بإعطائنا قيمة عددية (رقم المظهر الحالي للكائن).
 costume name: يقوم بإعطائنا سلسلة محرفية (اسم المظهر الحالي للكائن).
 size: يقوم بإعطائنا قيمة عددية (حجم الكائن).
 volume: يقوم بإعطائنا قيمة عددية (شدة صوت الكائن).

عندما نريد معرفة معلومات عن المنصة نقوم باختيار stage بدلاً من اسم الكائن، نميز في الخلفية المعلومات التالية:

backdrop #: يقوم بإعطائنا قيمة عددية (رقم الخلفية الحالي).
 backdrop name: يقوم بإعطائنا سلسلة محرفية (اسم الخلفية الحالي).
 volume: يقوم بإعطائنا قيمة عددية (شدة صوت الخلفية).



يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يجعل حجم الكائن 120% ثم يقول الحجم مقسوماً على 10.



تخبرنا هذه اللبنة بـ (الدقائق- الثواني- الساعة- التاريخ- اليوم- الشهر- السنة) الحالية بحسب توقيت حاسوبك، حيث تعطي هذه اللبنة قيمة عددية.

يوضح المثال التالي استخدام بسيط لتلك التعليمية، حيث يقول الكائن السنة الحالية.



days since 2000

تخبرنا هذه التعليمة بعدد الأيام التي انقضت منذ عام 2000 الميلادي.



يوضح المثال السابق استخدام بسيط لتلك التعليمة، حيث يقول عدد الأيام التي مرّت منذ عام 2000، لا بد أنك تذكرت تعليمة الـ round وهي تقرب القيمة لأقرب عدد صحيح.

ما حاجتنا لتعليمة round في المثال السابق؟

تعليمة days since 2000 تعطي قيم عددية تحوي فواصل (عدد الساعات)، لذلك قرّبنا لأقرب عدد صحيح. بإمكانك إزالة تعليمة round من البرنامج وتجريبه بنفسك.

username

تحدّثنا عن موقع السكراتش على شبكة الويب، وهذه التعليمة تخزن اسم المستخدم للمستخدم الذي يزور مشروعك. فمثلاً إذا أردت الترحيب بـ زائر مشروعك ما عليك إلا إضافة الكدسة البرمجية التالية إليه:



لبّات القلم

Pen Blocks

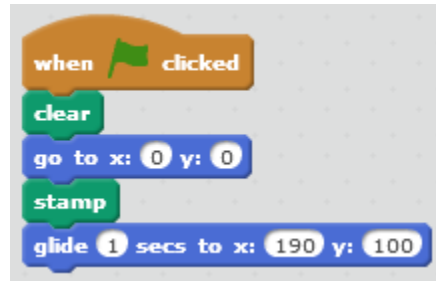
تقوم لبّات هذه الفئة بجعل الكائن يرسم مساره أثناء تحرّكه.



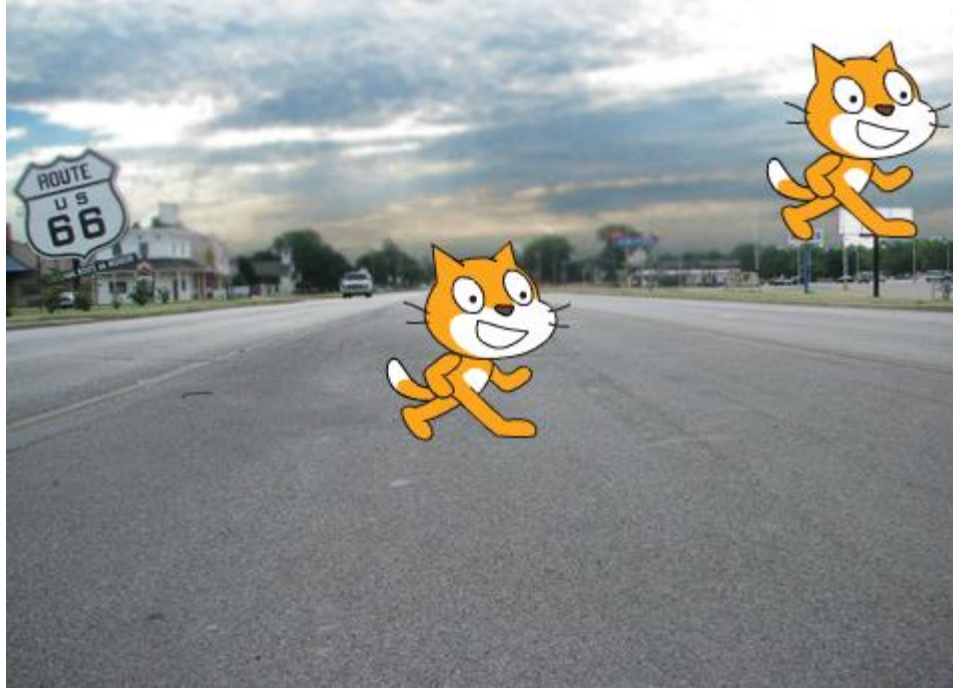
تقوم هذه اللبنة بمسح المنصة من جميع الرسوم المرسومة من قبل الكائنات. نضعها غالباً في بداية البرنامج لتهيئة منصة نظيفة.



تقوم هذه اللبنة بنسخ الكائن دون إدراج كائن جديد بين الكائنات.



يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند الضغط على العلم الأخضر يمسح الكائن المنصة ثم يذهب إلى الإحداثيات (x=0, y=0) وينسخ نفسه ثم يتزحلق لمدة ثانية واحدة إلى الإحداثيات (x=190, y=100). تنفيذ البرنامج السابق هو التالي:

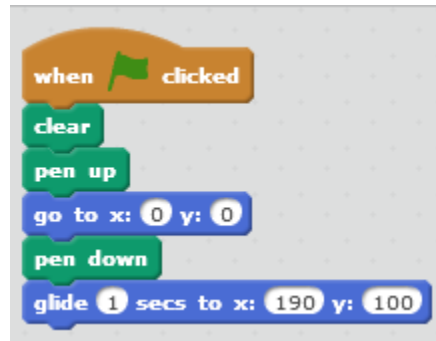


نلاحظ أن الكائن نسخ نفسه أثناء وجوده في الإحداثيات $(x=0, y=0)$ ، ثم ذهب الكائن الأصلي إلى الإحداثيات $(x=190, y=100)$ ، بينما بقي الكائن المنسوخ في الإحداثيات $(x=0, y=0)$ لأن عملية النسخ تمت قبل ترحلق الكائن ووصله إلى إحداثياته الجديدة.

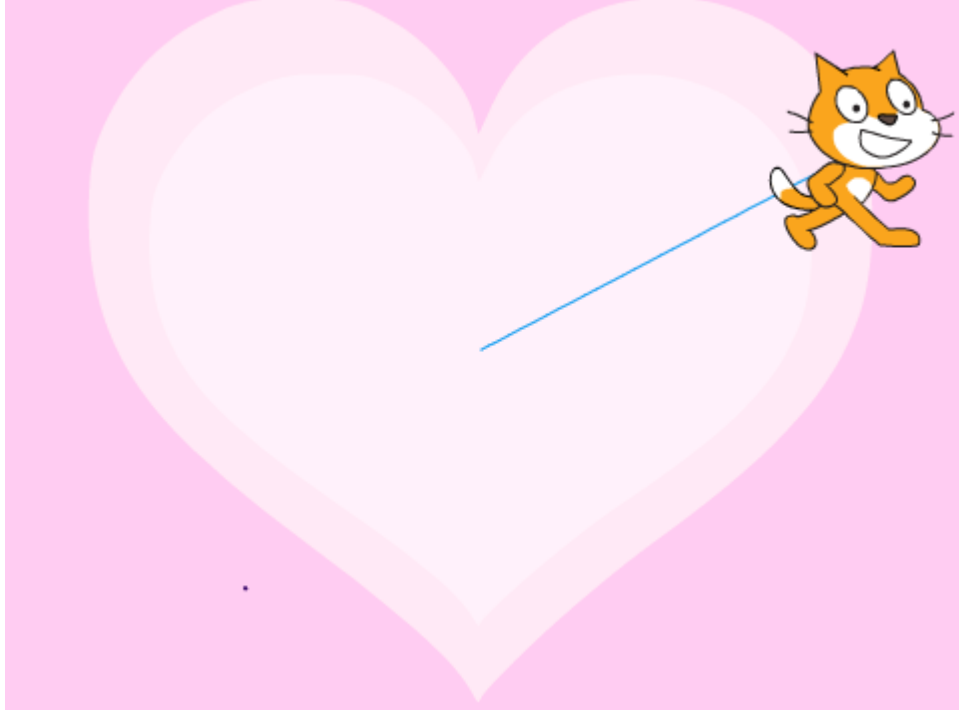


لننتقل الآن إلى جعل الكائن يرسم مساره أثناء تحركه. لنتخيل أن كل كائن في السكراتش يملك قلم للرسم، بإمكانه وضعه على المنصة بحيث يترك أثر لمساره أثناء تحرك الكائن وبإمكانه تخبئته بحيث لا يترك أي أثر لمساره أثناء تحركه على المنصة. تمكننا هاتان اللبنتان من إنزال قلم الكائن ورفع.

ملاحظة: الحالة الافتراضية لجميع الكائنات أن يكون القلم مرفوع.

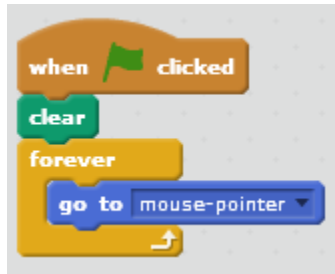


يوضح المثال السابق كيفية استخدام هاتين اللبنتين، فعند النقر على العلم الأخضر يتم مسح المنصة، ثم رفع القلم ويذهب الكائن إلى الإحداثيات $(x=0, y=0)$ ، ثم ينزل القلم ويتزحلق لمدة ثانية واحدة إلى الإحداثيات $(x=190, y=100)$ معطياً التنفيذ التالي:



بيناً قبل قليل في الملاحظة أن الحالة الافتراضية لجميع الكائنات أن يكون قلمها مرفوع، لماذا إذاً وضعنا تعليمة pen up في بداية البرنامج السابق؟

سأساعدك صديقي المتعلم في الإجابة عن هذا السؤال، قم بحذف الكدسة البرمجية السابقة وأنشئ بدلاً منها الكدسة التالية كما موضح في الشكل الآتي:



قم بتنفيذ البرنامج الآن، ماذا تلاحظ؟

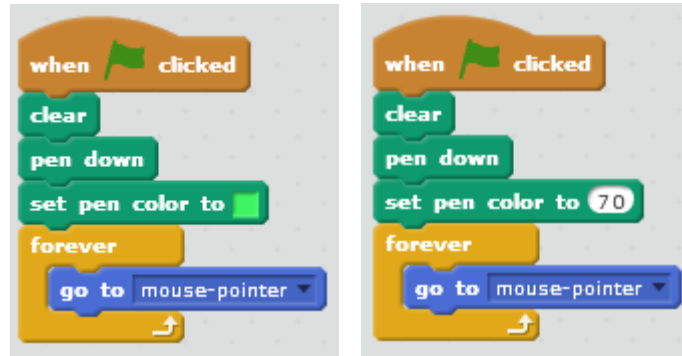
أنت لم تقم بوضع تعليمة pen down في برنامجك لكن الكائن قام بالرسم، إذاً السكراتش يخزن الوضع النهائي لقلم الكائن، وفي البرنامج السابق كان الوضع النهائي له pen down لذلك نحن بحاجة إلى

تعليلة pen up في بداية البرنامج في حال لا نريد الكائن أن يرسم عند بدء البرنامج الذي يحوي فيما بعد تعليلة الـ pen down.



إذا أردنا تحديد اللون الذي يرسم الكائن فيه مساره نستعمل واحدة من هاتين اللبتين، إما نحدد لون أو نضع القيمة العددية لهذا اللون بحيث تتراوح القيم العددية بين (0-200).

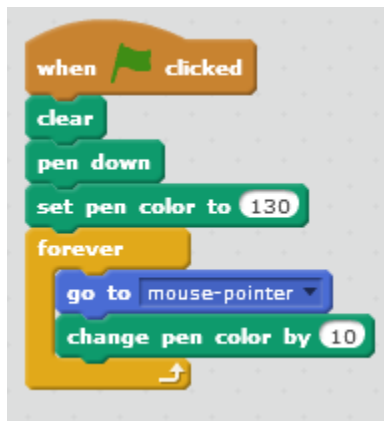
0=أحمر 70=أخضر 130=أزرق 170=وردي



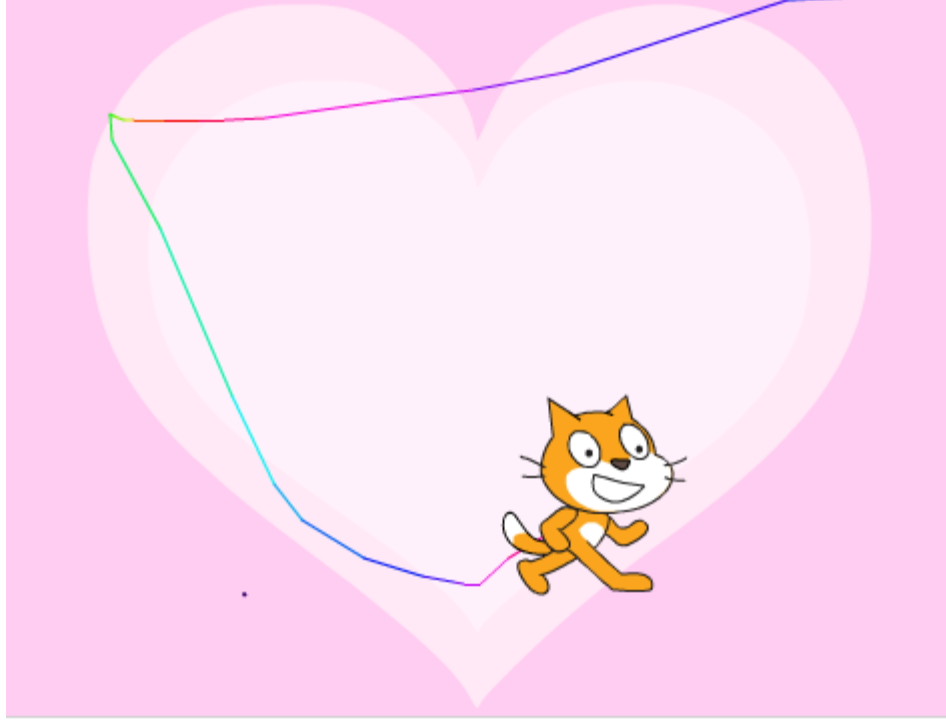
يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند الضغط على العلم الأخضر يسمح الكائن المنصة ثم يُنزل القلم ويجعل لونه أخضر، ويتبع حركة مؤشر الفأرة؛ أي يرسم الكائن تبعاً لمسار حركة مؤشر الفأرة. نلاحظ عدم وضع تعليلة pen up في البرنامج السابق لأننا نريد الكائن أن يبدأ برسم المسار مع بداية البرنامج.



تمكننا هذه اللبنة من تغيير لون القلم بمقدار معين.



يوضح المثال كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يتم مسح المنصة وإنزال القلم وجعل لونه مساوٍ للـ 130 (اللون الأزرق) بحيث يتبع الكائن مؤشر الفأرة ويغير لونه بمقدار 10. نلاحظ بدء لون المسار باللون الأزرق ثم تدرّجه كما موضح في الشكل التالي:



change pen shade by 10

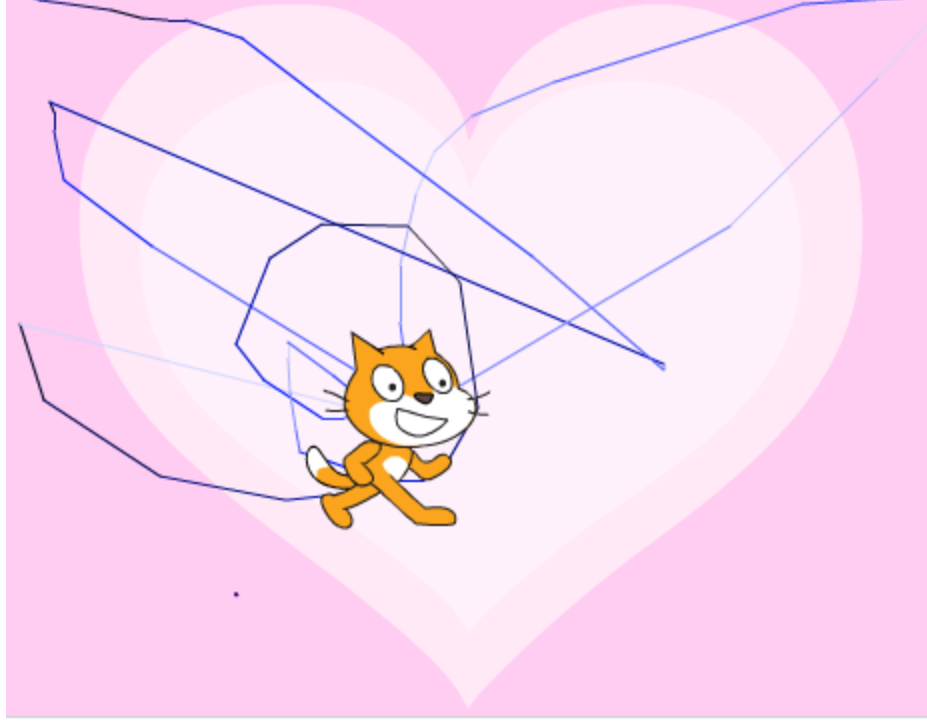
تمكننا هذه اللبنة من تغيير ظل القلم بقيمة محددة. يتراوح ظل القلم بين الـ (0-100) والقيمة الافتراضية هي 50.



ملاحظة: إذا كان ظل القلم مساوٍ للـ 0 فلون القلم سيكون مائل للأسود، أمّا إذا كان الظل مساوٍ للـ 100 فلون القلم سيكون مائل للأبيض.

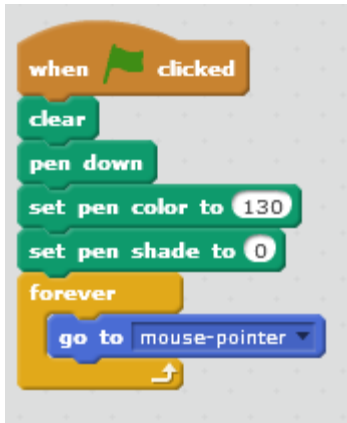


يوضح المثال كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يتم مسح المنصة وإنزال القلم الخاص بالكائن وجعل لون القلم أزرق بحيث يتبع الكائن مؤشر الفأرة ويغير ظله بمقدار 10، سيكون لون القلم من مشتقات الأزرق كما موضح بالشكل التالي:

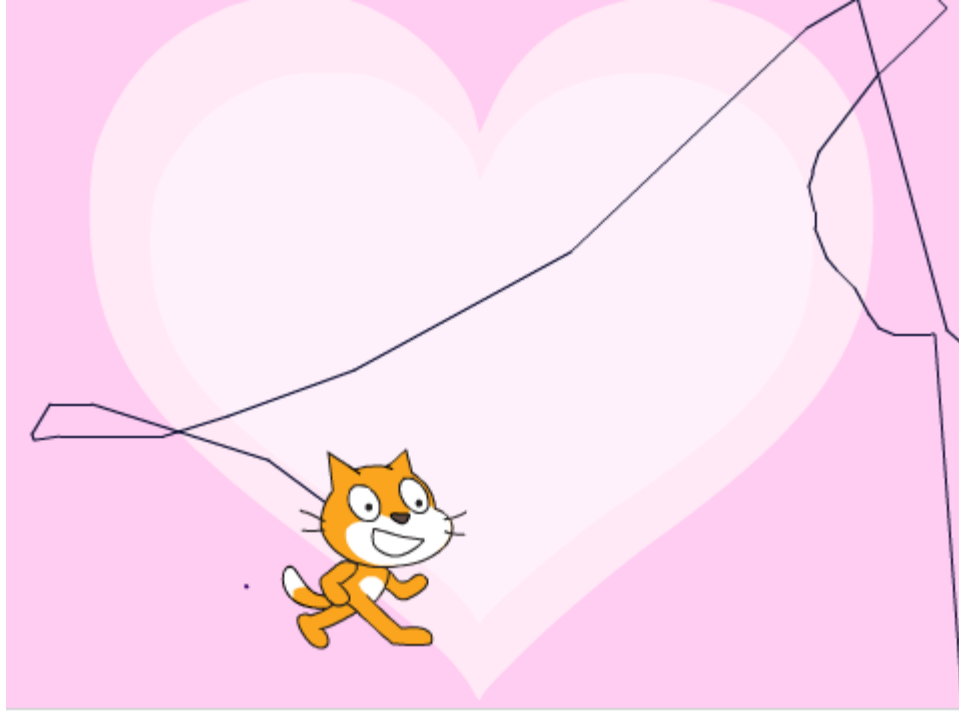


set pen shade to 0

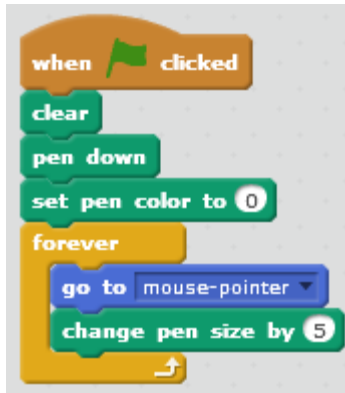
تمكننا هذه اللبنة من إعطاء ظل القلم قيمة محددة.



يوضح المثال كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يتم مسح المنصة وإنزال القلم الخاص بالكائن وجعل لون القلم أزرق وجعل ظله مساوٍ للـ 0 بحيث يتبع الكائن مؤشر الفأرة (أشرنا سابقاً أن الظل المساوي للـ 0 سيكون لونه مائل للأسود)، سيعطي البرنامج السابق التنفيذ التالي:



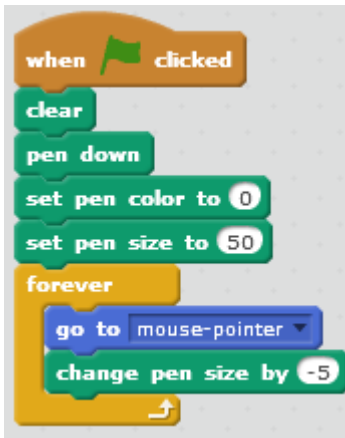
change pen size by



تمكننا هذه اللبنة من تغيير حجم خط القلم. في حال وضع قيم التغير موجبة سيزداد حجم الخط، أما في حال وضعها سالبة سيتناقص حجم الخط.

يوضح المثال كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يتم مسح المنصة وإنزال القلم الخاص بالكائن وجعل لون القلم أحمر بحيث يتبع الكائن مؤشر الفأرة ويزيد حجمه بمقدار 5.

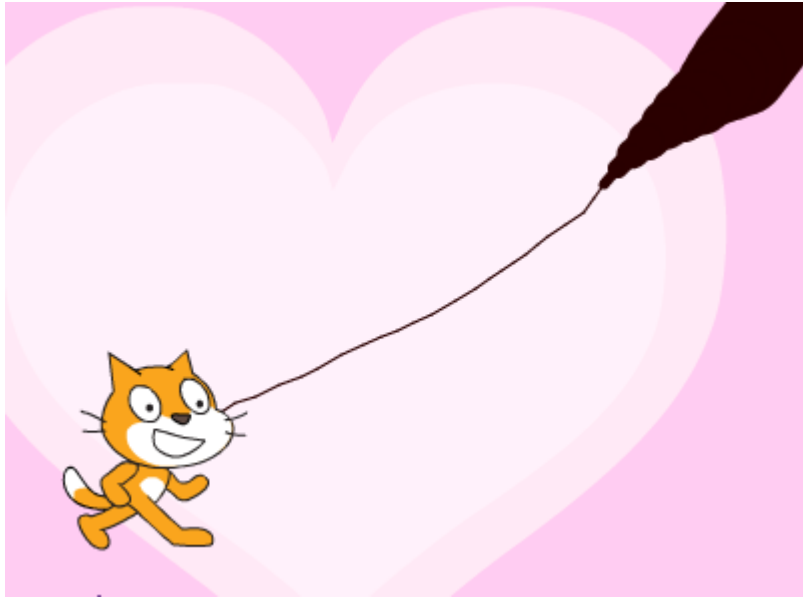
set pen size to



تمكننا هذه اللبنة من إعطاء قيمة محددة لحجم القلم.

يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يتم مسح المنصة وإنزال القلم الخاص بالكائن وجعل لون القلم أحمر وجعل

حجمه 50 بحيث يتبع الكائن مؤشر الفأرة ويزيد حجمه بمقدار 5- أي ينقص بمقدار 5 ليعطي التنفيذ الآتي (يبدأ الخط تخين ثم يستدق):



نلاحظ اختفاء لبنات هذه الفئة من المنصة ما عدا لبنة clear، فبإمكان المنصة أن تسمح ما عليها من آثار أقلام الكائنات.

لبّات الصوت Sound Blocks

تضيفي التأثيرات الصوتية على الألعاب والمشاريع حيوية، لذلك يتيح لنا السكراتش إضافة أصوات إلى مشاريعنا عن طريق لبّات هذه الفئة.

play sound

تقوم هذه اللبنة بتشغيل صوت نحدده لها، ولا يتوقف البرنامج عندها؛ أي يتم تنفيذ التعليمات التي تليها أثناء إصدارها للصوت.



يوضح المثال السابق استخدام هذه اللبنة، فعند النقر على العلم الأخضر يصدر الكائن الصوت المحدد له meow ويقول أثناء إصداره للصوت I'm a cat!! لمدة ثانية واحدة.

play sound until done

تقوم هذه اللبنة بتشغيل صوت نحدده لها، بحيث يتوقف البرنامج عندها؛ أي لا يتم تنفيذ التعليمات التي تليها إلا بعد انتهائها من إصدار الصوت.



يوضح المثال السابق استخدام هذه اللبنة، فعند النقر على العلم الأخضر يصدر الكائن الصوت المحدد له meow وبعد انتهائه من إصدار الصوت يقول I'm a cat!! لمدة ثانية واحدة.

stop all sounds

عندما يصل المترجم إلى هذه اللبنة يقوم بإيقاف جميع الأصوات المُشغلة في تلك اللحظة.

play drum ▾ for beats

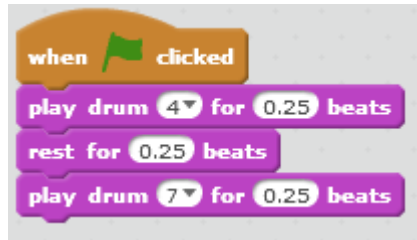
تقوم هذه اللبنة بتشغيل نغمة لألة موسيقية محددة لمدة زمنية محددة أيضاً. بالرغم من وجود كلمة drum والتي تعني طبل، لكن تتيح هذه اللبنة تشغيل خيارات عدة من نغمات آلات موسيقية مختلفة.



يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يشغل الكائن الآلة رقم 4 لمدة 0.4 من وحدة الإيقاع ويكررها خمس مرّات.

rest for beats

توقف هذه اللبنة الصوت لمدة محددة من وحدة الإيقاع.



يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يشغل الكائن الآلة الموسيقية رقم 4 لمدة 0.25 من وحدة الإيقاع ويتوقف عن إصدار الصوت لمدة 0.25 من وحدة الإيقاع أيضاً ثم يعود بعدها لتشغيل الآلة رقم 7 لمدة 0.25.

play note for beats

تقوم هذه اللبنة بتشغيل العلامة الموسيقية المحددة لمدة زمنية محددة أيضاً. (بإمكانك إعطاء قيم للعلامات الموسيقية تتراوح بين 0-127 مع العلم أن القيمة 60 هي القيمة الوسطى).

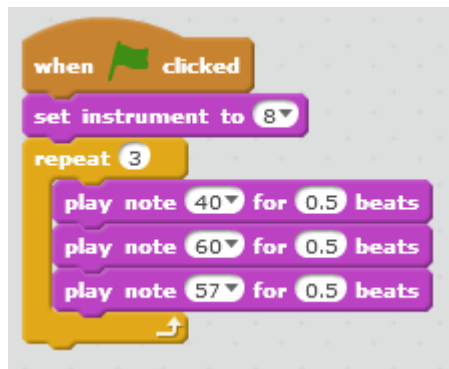


يوضح المثال السابق كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يتم تشغيل العلامة 40 ثم 60 ثم 57 مدة 0.5 من وحدة الإيقاع لكل منها، ثم يكرر تلك العملية ثلاث مرّات.

set instrument to

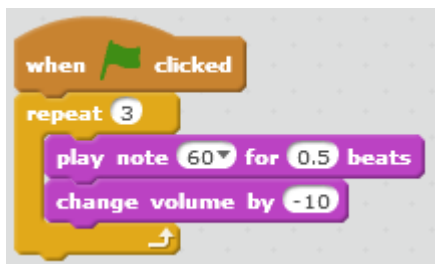
في المثال السابق، ألم تتساءل صديقي المتعلّم ما هي الآلة التي تشغّل العلامة الموسيقية المحددة؟ الآلة هي البيانو في الوضع الافتراضي لكن بإمكانك تغيير تلك الآلة عن طريق استعمال هذه اللبنة.

المثال التالي يوضح استخدام تلك اللبنة، فعند النقر على العلم الأخضر يتم جعل الآلة رقم 8 Cello ويكرر ثلاث مرات عملية تشغيل العلامات 40، 60، 57 على الترتيب مدة 0.5 من وحدة الإيقاع لكل منها.



change volume by

تمكّنا هذه اللبنة من التحكم بشدة الصوت، حيث تتراوح شدة الصوت بين 0-100 (الوضع الافتراضي لها 100).



يبين المثال السابق كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يقوم الكائن بتشغيل العلامة الموسيقية 60 لمدة 0.5 من وحدة الإيقاع ثم ينقص شدة الصوت بمقدار 10 ويكرر العمليتين

السابقتين ثلاث مرّات.

set volume to 0%

في المثال السابق أَلَمْ تتساءل صديقي المتعلّم ماهي شدة الصوت لحظة النقر على العلم الأخضر؟ للإجابة عن سؤالك ببساطة: شدة الصوت عند كل مرة ننقر فيها العلم الأخضر تكون الشدة نفسها التي توقف البرنامج عندها؛ أي بفرض انتهى التنفيذ عندما كانت شدة الصوت 60، فعند النقر على العلم الأخضر في المرة التي تليها ستكون شدة الصوت 60.

تتيح لك هذه اللبنة بتحديد شدة الصوت في أي لحظة تريدها من البرنامج.

في المثال التالي سنحدد شدة الصوت 100 في بداية البرنامج، وسيشغل الكائن العلامة الموسيقية رقم 60 لمدة 0.5 من وحدة الإيقاع وينقص شدة الصوت بمقدار 10، سيكرر العمليتين السابقتين عشرة مرات. ما هي شدة الصوت في نهاية البرنامج؟ للإجابة عن هذا السؤال علينا أن نطرح من شدة الصوت الابتدائية وهي 100 القيمة المطلقة لتغير شدة الصوت 10 مضروبة بعدد مرات التكرار 10 أيضاً. لتصبح العملية الحسابية $100 - (10 \times 10) = 0$ إذن شدة الصوت عند انتهاء البرنامج من التنفيذ في كل مرة هي صفر.

volume

لإظهار شدة الصوت على المنصة بإمكانك الضغط على المربع كما في الشكل التالي:

☒ volume

change tempo by

تمكننا هذه اللبنة من تغيير سرعة أداء كائن ما، مع العلم أن سرعة الأداء هي عدد الضربات في الدقيقة الواحدة. وكلما زادت سرعة الأداء زادت سرعة الصوت الذي يشغله الكائن.

في المثال التالي، عند النقر على العلم الأخضر يكرر الكائن ثلاث مرات (تشغيل العلامات 40، 60 على الترتيب مدة 0.5 من وحدة الإيقاع لكل منها وزيادة سرعة الأداء بمقدار 10).



set tempo to bpm

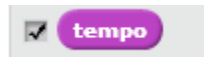
تقوم هذه اللبنة بتحديد سرعة الأداء.

في المثال التالي، عند النقر على العلم الأخضر يجعل الكائن سرعة أدائه 100 ضربة في الدقيقة الواحدة، ويكرر ثلاث مرات (تشغيل العلامات 40، 60 على الترتيب لمدة 0.5 وحدة إيقاع وزيادة سرعة الأداء بمقدار 10).



tempo

لإظهار سرعة الأداء على المنصة بإمكانك الضغط على المربع كما في الشكل التالي:

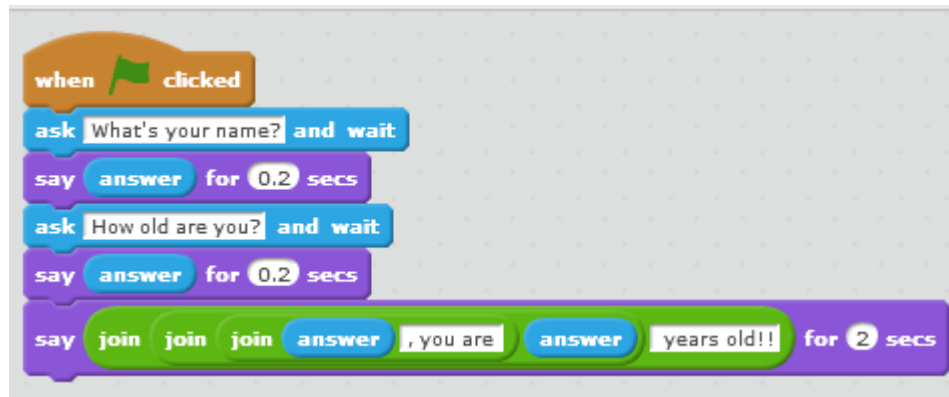


لبينات البيانات Data Blocks

المتحولات Variables

نعلم جميعاً وجود مراكز للذاكرة في أدمغتنا، تقوم بتخزين المعلومات واسترجاعها عند الحاجة، وكذلك كائنات السكراتش قادرة على تخزين المعلومات واسترجاعها عندما يُطلب ذلك. لكن ما هي المعلومات التي تخزنها وأين تخزنها؟ كائنات السكراتش قادرة على تخزين أي معلومة يريد المبرمج في أماكن معينة في الذاكرة تدعى المتحولات سنتعرف عليها في هذا الفصل.

نهدف في المثال التالي بجعل الكائن يسأل المستخدم عن اسمه وعمره، ثم يقول له Noura, you are 17 years old!! بفرض أن الاسم المدخل هو Noura والعمر 17.



يبدأ البرنامج بالسؤال عن الاسم ويقول الاسم لمدة زمنية قصيرة ثم يسأل عن العمر ويقول له لمدة قصيرة أيضاً. التساؤل الآن، هل يحقق البرنامج السابق الهدف الذي تحدثنا عنه؟

الجواب ببساطة هو لا، لأن الكائن سينفذ التنفيذ الآتي بفرض أن الاسم المدخل هو Noura والعمر 17.



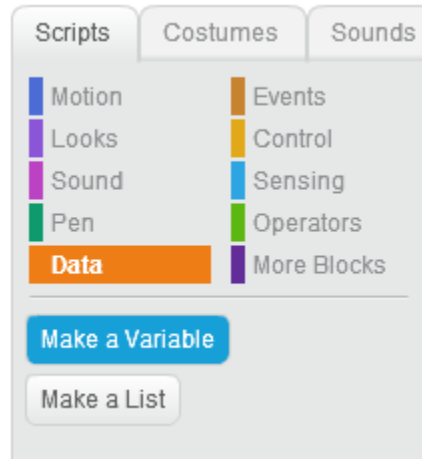
الكائن يحتفظ بآخر جواب مدخل له في لبنة الـ answer. والمثال السابق يوضح ذلك؛ بما أن 17 هي آخر جواب مدخل له فهذا يعني أن القيمة الموجودة في لبنة الـ answer ستبقى إلى أن يُسأل المستخدم سؤال جديد.

ويجب عنه، فتصبح لبنة الـ answer تحوي الإجابة الجديدة.

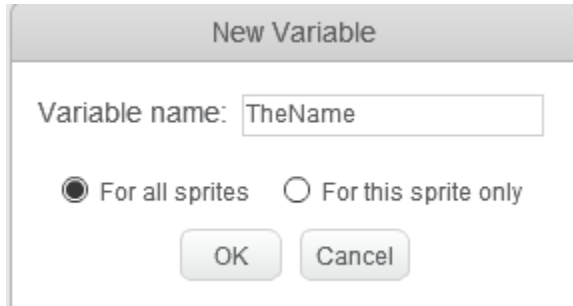
لكن ماذا لو أردنا الاحتفاظ بكلتي الإجابتين؟ ببساطة ما علينا سوى استخدام المتحولات.

نلعم أن خزان المياه نحفظ فيه الماء (التي حصلنا عليها من البئر، الصنبور، النبع...) ريثما نحتاجها، كذلك المتحولات، تشبه خزان المياه الذي يحوي ماء لكن المتحولات تحوي قيم (أرقام، أحرف أو قيم منطقية). نحصل على هذه القيم من لبنات عديدة كلبنة الـ answer و الـ size و الـ position و....

لنعود إلى مثالنا السابق ونجعله يقوم فعلياً بالهدف الذي صنع من أجله. نريد أن نستخدم متحولين أحدهما لحفظ الاسم والثاني لحفظ العمر. من قائمة Data نختار Make a Variable كما موضح في الشكل:



بعد الضغط على Make a Variable يظهر لدينا مستطيل لتحديد اسم المتحول الذي سنسميه TheName في برنامجنا كما هو موضح في الشكل:



ثم نضغط على زر ok وبذلك نكون انتهينا من إنشاء المتحول. نلاحظ بعد ذلك ظهور لبنات جديدة في لبنات هذه الفئة والتي ستسمح لنا من إجراء العمليات على المتحولات واستخدام قيمها.

ننشئ متحول ثانٍ لنخزن فيه عمر المستخدم ولنسميه TheAge.

الآن لجعل الكائن يقول اسم المستخدم وعمره علينا باستبدال الأجوبة بالمتحولات التي أنشأناها.

كيف أختار اسماً للمتحول؟

بإمكانك اختيار اسم المتحول الذي ترغب به، لكن من الأفضل أن تختار اسماً مناسباً للغرض الذي صنع من أجله المتحول، لسهولة فهم البرنامج أثناء مراجعته من قبلك أو حتى قراءته من قبل مبرمج آخر.

انتبه: لا يمكنك إنشاء أكثر من متحول يحمل نفس الاسم. ويجب ملاحظة أن السكراتش يستطيع التمييز بين الأحرف الصغيرة والأحرف الكبيرة، بإمكانك تسمية متحول Name وتسمية متحول آخر name في نفس البرنامج بينما لا تستطيع تسمية أكثر من متحول Name في نفس البرنامج.



تمكننا هذه التعليمة من إسناد قيم للمتحول.

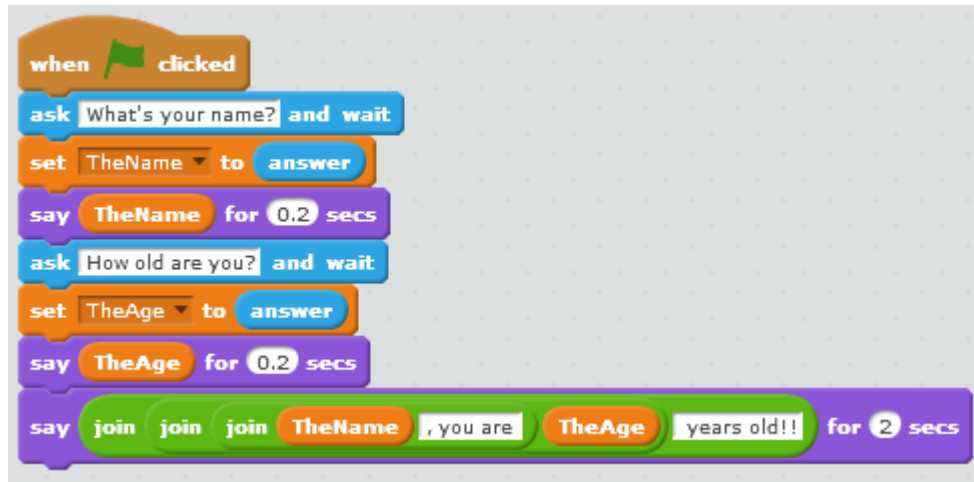
يوضح المثال التالي استخدام هذه اللبنة، فعند النقر على العلم الأخضر يصبح المتحول المحدد x يحوي قيمة 120.5.



كما يمكن أن نسند للمتحول قيم نأخذها من البرنامج مثل اسم الخلفية أو الإحداثيات على محور الفواصل أو جواب لسؤال (الذي سيساعدنا في حل مثالنا السابق).



بالعودة لمثالنا السابق، سنسند الجواب الأول إلى TheName وسنسند الجواب الثاني إلى TheAge وسنقوم بجعل الكائن يقول في البداية قيمة TheName بدلاً من قيمة answer ثم يقول قيمة TheAge كما موضح في الشكل:



سينفذ البرنامج السابق التنفيذ التالي:



قد يتوارد إلى ذهنك فكرة جديدة ألا وهي أن يقول لك الكائن بعمر ك بعد عشرة سنوات، لماذا لا تجرب أن تقوم بذلك؟



تمكننا هذه التعليمة من زيادة وإنقاص قيمة المتحول.

يوضح المثال التالي استخدام هذه اللبنة، فعند النقر على العلم الأخضر يُنقص 5 من قيمة المتحول x. أي في حال كانت القيمة قبل الإنقاص 32 ستصبح بعد الإنقاص 27.



hide variable

show variable

تمكننا هاتان اللبنتان من إخفاء وإظهار قيمة المتحول (أو اسمه وقيمته) عن المنصة.

في المثال التالي يظهر المتحول على المنصة لمدة عشرة ثواني ثم يختفي.



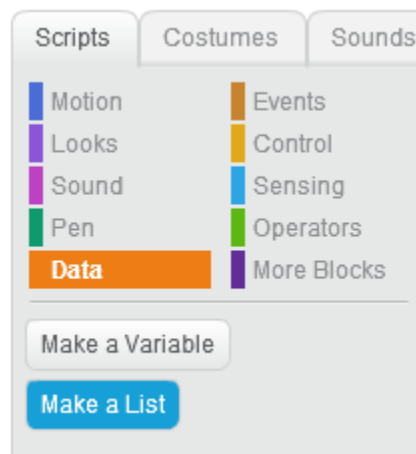
ملاحظة: بإمكانك تغيير طريقة عرض المتحول على المنصة من خلال النقر مرتين متتاليتين على صورته على المنصة.

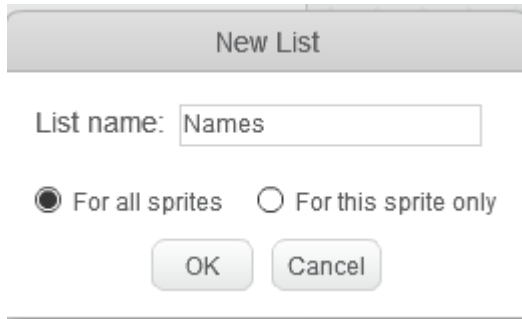
القوائم Lists

القط سكراتش معلم في مدرسة ويريد أن يحتفظ بأسماء طلابه وطالباته في سجل باستخدام برنامج السكراتش، سنساعد صديقنا القط في إنجاز هذه المهمة.

لدينا في السكراتش ما يعرف بالقوائم **Lists** وهي عبارة عن قائمة (مجموعة) تحوي عدداً من المتحولات، بحيث يمكننا حذف متحولات موجودة مسبقاً فيها أو إضافة متحولات جديدة إليها في أي وقت نريده من البرنامج ويمكننا تحديد القيم التي ستحويها هذه المتحولات.

لإنشاء قائمة جديدة، من Data نختار Make a List كما موضح في الشكل:





A dialog box titled "New List" with a text input field labeled "List name:" containing the text "Names". Below the input field are two radio buttons: "For all sprites" (selected) and "For this sprite only". At the bottom are "OK" and "Cancel" buttons.

بعد الضغط على Make a List يظهر لدينا مستطيل لتحديد اسم القائمة التي سنسميها Names في برنامجنا لمساعدة القط كما هو موضح في الشكل:

الآن سنضيف أسماء طلاب القط سكراتش (Tom- Claire Jack- Charlie- Mary- Venessa إلى القائمة (كل اسم من القائمة يسمى عنصر).



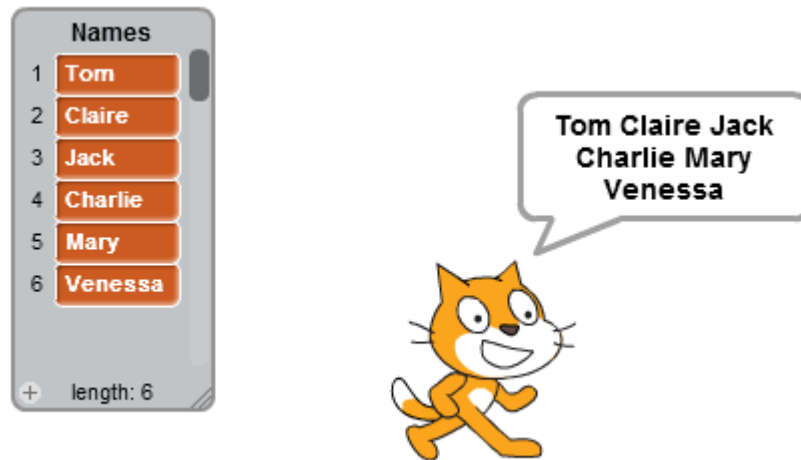
تمكننا هذه اللبنة من إضافة الأسماء إلى القائمة المحددة (قائمة القط المعلم في مثالنا)، مع العلم أن القائمة تكون خالية من أي عنصر عند إنشائها.



يوضح المثال التالي كيفية إضافة أسماء طلاب القط إلى القائمة ثم يقول هذه الأسماء لمدة ثلاث ثوانٍ عند انتهائه من الإضافة.



نلاحظ أن القائمة أصبحت ممثلة بالعناصر السابقة (الأسماء التي كتبناها) وأن طولها length أصبح مساوٍ لعدد العناصر ويساوي 6. كما نلاحظ أن الكائن يقول العناصر بوضع فراغ بين العناصر والذي يليه.



بعد أن صنعنا القائمة تذكر القط أن الطالب Charlie غادر صفه، لذا يجب علينا أن نساعد القط في حذف اسم Charlie من القائمة.

لنمعن النظر في القائمة الموجودة قرب القط في الصورة السابقة نجد أن العنصر الأول الذي تمت إضافته له رقم 1 والعنصر الثاني 2 والثالث 3 وهكذا؛ فكل عنصر في هذه القائمة له رقم.



تمكننا هذه اللبنة من حذف العنصر المحدد من القائمة المحددة وذلك بتحديد رقم العنصر واسم القائمة، كما تمكننا من حذف جميع عناصر القائمة باختيار all أو حذف العنصر الأخير منها فقط.

بالعودة لمثالنا السابق نجد أن اسم Charlie هو العنصر الرابع. لذا علينا إضافة لبنة الحذف لإزالة العنصر الرابع من القائمة. سنضيف أيضاً إلى بداية البرنامج لبنة تقوم بحذف جميع العناصر وذلك لجعل المصفوفة فارغة كلما نقرنا على العلم الأخضر، ليصبح البرنامج بالشكل التالي:



نلاحظ أن العنصر الرابع الذي يحتوي Charlie قد حل محله العنصر الذي يليه Mary، وأصبح طول المصفوفة 5 بدلاً من 6 (بسبب حذف عنصر منها) كما موضح في الشكل:



طلب مدير المدرسة من المعلمين أن يضعوا أسماءهم في بداية قائمة أسماء الطلاب.



تمكننا هذه اللبنة من إضافة عناصر إلى القائمة في أي مكان نحدده منها عن طريق رقم العنصر أو إلى نهايتها أو إلى مكان عشوائي منها.

يستطيع القط الآن أن يضيف اسمه Cat بسهولة إلى القائمة وذلك من خلال وضع هذه اللبنة في أي مكان من البرنامج بشرط وضعها قبل لبنة الـ say.



بعد أن انتهى القط من إعداد قائمته أرسلها إلى المدير، فكان هناك خطأ في اسم الطالبة Mary فهي تدعى Maria. ارتبك القط بسبب هذه الخطأ، هل هناك طريقة لانتشال صديقنا من هذه المشكلة؟

replace item of list with

تمكننا هذه اللبنة من استبدال قيمة عنصر معين نحدد رقمه أو عنصر عشوائي أو العنصر الأخير من القائمة بقيمة جديدة نحددها له في الفراغ.

سنستبدل الآن اسم Mary بـ Maria. لنتأمل معاً ماهو رقم العنصر الذي يحوي اسم Mary؟

في البداية (قبل أن نحذف العنصر الذي يحوي Charlie) كان العنصر الخامس، فبعد الحذف يكون أصبح العنصر الرابع، لكننا أضفنا اسم القط إلى بداية القائمة فأصبح العنصر الذي يحوي اسم Mary العنصر الخامس من جديد. لذا علينا أن نستبدل العنصر الخامس بـ Maria، ليصبح البرنامج بالشكل الآتي:



item of list

تخبرنا هذه اللبنة بالقيمة المختزنة في عنصر معين نحدد رقمه أو عنصر عشوائي أو العنصر الأخير من القائمة.

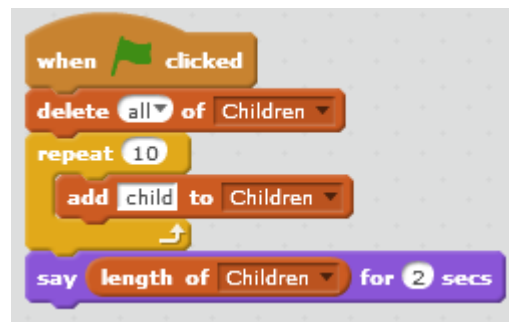
في المثال التالي بفرض أن لدينا قائمة اسمها Fruit تحتوي مجموعة من أسماء الفواكه، سيقوم الكائن بقول عنصر عشوائي منها لمدة ثانيتين عند النقر على العلم الأخضر.



length of list

تعطينا هذه اللبنة قيمة عددية وهي طول القائمة بمعنى آخر عدد عناصر القائمة.

في المثال التالي يقوم الكائن بحذف جميع العناصر من المصفوفة Children ، وبعدها تكرار إضافة عنصر يحوي كلمة child عشرة مرات، ثم يقول طول القائمة (عدد عناصرها).

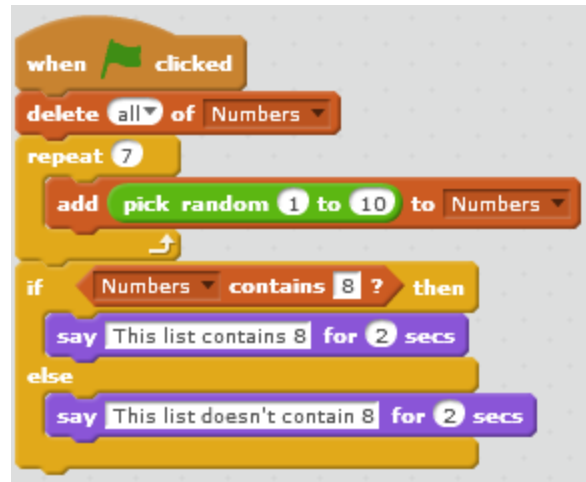


لا بدّ أنك استنتجت أن عدد عناصرها هو 10 لأننا كررنا عملية إضافة عنصر واحد إلى هذه القائمة عشرة مرات، كما أننا حذفنا جميع العناصر السابقة التي كانت متواجدة في القائمة قبل النقر على العلم الأخضر.

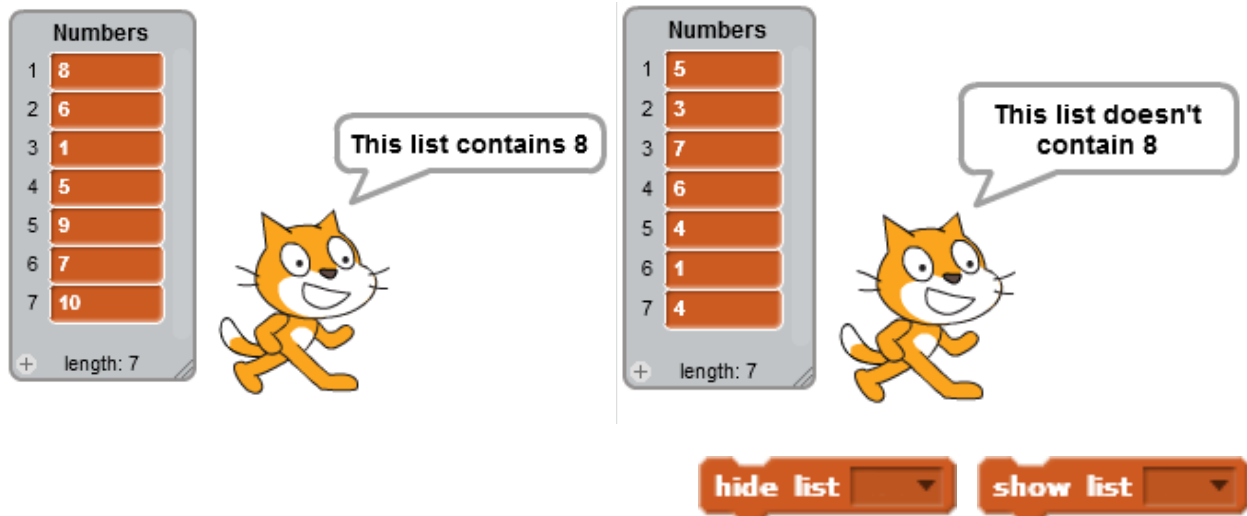
list contains

تعطينا هذه اللبنة قيمة منطقية true أو false؛ إما القائمة تحوي قيمة ما true أو لا تحويها false .

يوضح المثال التالي كيفية استخدام هذه اللبنة، فعند النقر على العلم الأخضر يقوم الكائن بحذف جميع العناصر من القائمة Numbers ثم يكرر سبع مرات إضافة رقم عشوائي إليها يتراوح بين 1-10، وعندما ينتهي من إضافة الأرقام السبعة (من إنشاء سبعة عناصر) يختبر هل الرقم 8 موجود ضمن القائمة؛ فإذا كان موجود يقول الكائن This list contains 8 لمدة ثانيتين، أما إذا لم يكن موجود فيقول This list doesn't contain 8 لمدة ثانيتين أيضاً.



يكون تنفيذ البرنامج كالتالي:



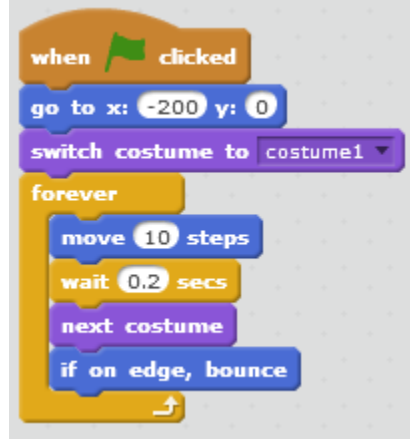
تمكننا هاتان اللبنتان من إخفاء وإظهار القائمة المحددة عن المنصة.
في المثال التالي تظهر القائمة Numbers على المنصة لمدة عشرة ثواني ثم تختفي.



أمثلة مشروحة

المثال الأول: حركة كائن

سنقوم بالمثال التالي بتحريك الكائن على محور الفواصل مع تغيير مظهره كل 0.2 ثانية. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسة البرمجية التالية.



عند النقر على العلم الأخضر، ينتهي الكائن لبدء البرنامج بالذهاب إلى الإحداثيات $x: -200$, $y: 0$ وجعل مظهره costume1. ويبقى يكرر باستمرار: التحرك عشرة خطوات والانتظار 0.2 ثانية وتغيير مظهره (جعل مظهره المظهر التالي) والاختبار فيما إذا كان يلامس الحافة أو لا، فإذا كان يلامسها يستدير.

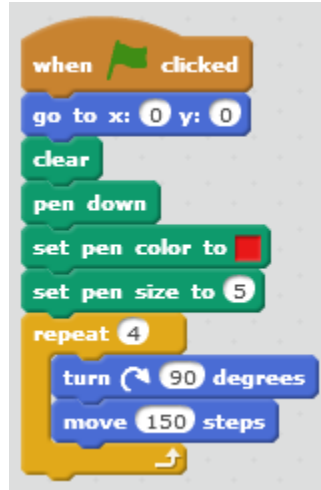
لماذا وضعنا انتظار لمدة 0.2 ثانية؟

وضعنا لبنة الانتظار لكي نجعل الحركة تبدو واقعية ونرى كل مظهر لمدة 0.2 ثانية. بإمكانك صديقي المتعلم إزالة تعليمة الانتظار وتنفيذ البرنامج، ستلاحظ عدم وضوح مظاهر الكائن لأن الحركة أصبحت سريعة جداً (لا يوجد فاصل زمني بين المظهرين).

يمكنك تغيير سرعة الكائن بتغيير الفاصل الزمني بين كل مظهرين. تزداد السرعة كلما كان الفاصل الزمني أقل والعكس صحيح.

المثال الثاني: رسم مربع

سنقوم بالمثال التالي بجعل الكائن يرسم مربع. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسة البرمجية التالية.



كيف نبدأ التفكير لتنفيذ مشروع كهذا؟

أولاً كيف يمكن للكائن أن يرسم مربع؟ بإمكان الكائن أن يرسم مربع بإنزال القلم الخاص به على المنصة والتحرك بمسار مربعي.

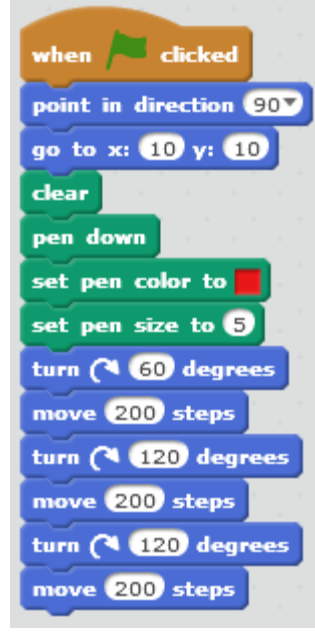
عند الضغط على العلم الأخضر يذهب الكائن إلى مبدأ الإحداثيات $x:0, y:0$ ثم يمسح المنصة (للتأكد من خلوها من أي رسم سابق) ويُنزل القلم الخاص به جاعلاً لون القلم أحمر وحجمه 5.

ومن أجل تحرك الكائن في المسار المربعي: يكرر أربع مرات (عدد أضلاع المربع) الدوران 90 درجة (الزاوية بين كل ضلعين في المربع) والتحرك 150 خطوة، بإمكانك زيادة وإنقاص عدد الخطوات وبذلك تتغير مساحة المربع المرسوم.

لرسم المربع بشكل تدريجي يمكنك إضافة تعليمة انتظار إلى حلقة التكرار.

المثال الثالث: رسم مثلث متساوي الأضلاع

سنقوم بالمثال التالي بجعل الكائن يرسم مثلث متساوي الأضلاع. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسة البرمجية التالية.



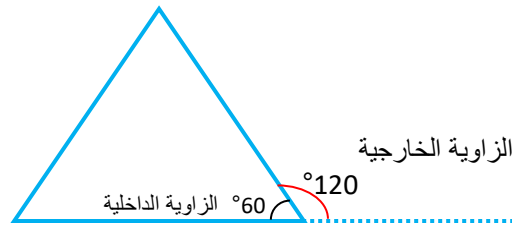
كيف نبدأ التفكير لتنفيذ مشروع كهذا؟

أولاً كيف يمكن للكائن أن يرسم مثلث؟ بإمكان الكائن أن يرسم مثلث بإنزال القلم الخاص به على المنصة والتحرك بمسار مثلثي.

نعلم أن كل زاوية في المثلث المتساوي الأضلاع تساوي 60 درجة، لنتخيل الكائن يتحرك لرسم مثلث طول كل من أضلاعه 200 خطوة، بفرض أنه بدأ مساره وكان اتجاهه 90 درجة، سيستدير 60 درجة ويتحرك 200 خطوة، هل يمكننا تكرير ذلك ثلاث مرّات؟

الجواب هو لا، لا يمكنك ذلك، بإمكانك أن تجرب ذلك بنفسك ستجد أن الشكل المرسوم ليس مثلث. عند تحرك الكائن على مسار مثلثي يجب عليه أن يدور بمقدار الزاوية الخارجية للمثلث والتي تساوي 120 درجة وليس بمقدار الزاوية الداخلية التي تساوي 60 درجة.

ملاحظة: الزاوية الخارجية في مثلث تساوي مجموع الزاويتين الداخليتين غير المجاورتين لها.



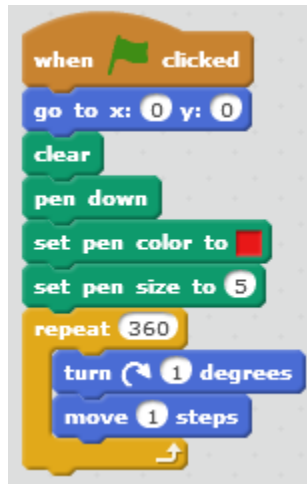
لننفذ ذلك في السكراتش. عند الضغط على العلم الأخضر يجعل الكائن اتجاهه 90 درجة ويذهب إلى مبدأ الإحداثيات x:10, y:10 ثم يمسح المنصة (للتأكد من خلّوها من أي رسم سابق) ويُنزل القلم الخاص به جاعلاً لون القلم أحمر وحجمه 5. بعد ذلك يدور 60 درجة ويتحرك 200 خطوة ثم يدور 120 درجة ويتحرك 200 خطوة، يكرر العملية الأخيرة (الدوران 120 درجة والتحرك 200 خطوة) مرتين.

لماذا دار الكائن في بداية البرنامج 60 درجة ولم يدور 120 درجة؟

دار 60 درجة فقط لجعل رأس المثلث يكون في الأعلى وبذلك يبدو الشكل بمظهر أكثر أناقة، لكن لا مشكلة في حال دورانه في البداية 120 درجة أيضاً، سيكون الفرق فقط أن المثلث مقلوب. صديقي المتعلم بإمكانك تجريب ذلك، عندما يطرأ أي تساؤل آخر في ذهنك حاول أن تغير في اللبنة وفكر ماذا يمكن أن يحدث في حال كان البرنامج كذا بدل من هكذا؟

المثال الرابع: رسم دائرة

سنقوم بالمثال التالي بجعل الكائن يرسم دائرة. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسة البرمجية التالية.



كيف نبدأ التفكير لتنفيذ مشروع كهذا؟

أولاً كيف يمكن للكائن أن يرسم دائرة؟ بإمكان الكائن أن يرسم دائرة بإنزال القلم الخاص به على المنصة والتحرك بمسار دائري.

عند الضغط على العلم الأخضر يذهب الكائن إلى مبدأ الإحداثيات $x:0, y:0$ ثم يمسح المنصة (للتأكد من خلّوها من أي رسم سابق) ويُنزل القلم الخاص به جاعلاً لون القلم أحمر وحجمه 5. ويكرر 360 مرة: الدوران درجة واحدة والتحرّك خطوة واحدة.

لماذا كررنا الدوران درجة واحدة 360 مرة؟
لقد كررنا الدوران درجة واحدة 360 مرة لأن مجموع درجات الدائرة هو 360 درجة.
ماذا يحدث لو أن الكائن لا يتحرك في كل مرة يدور فيها؟

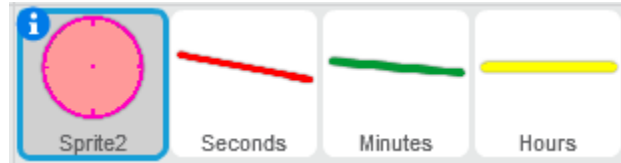
سيكون الشكل المرسوم نقطة وليس دائرة.

المثال الخامس: الساعة

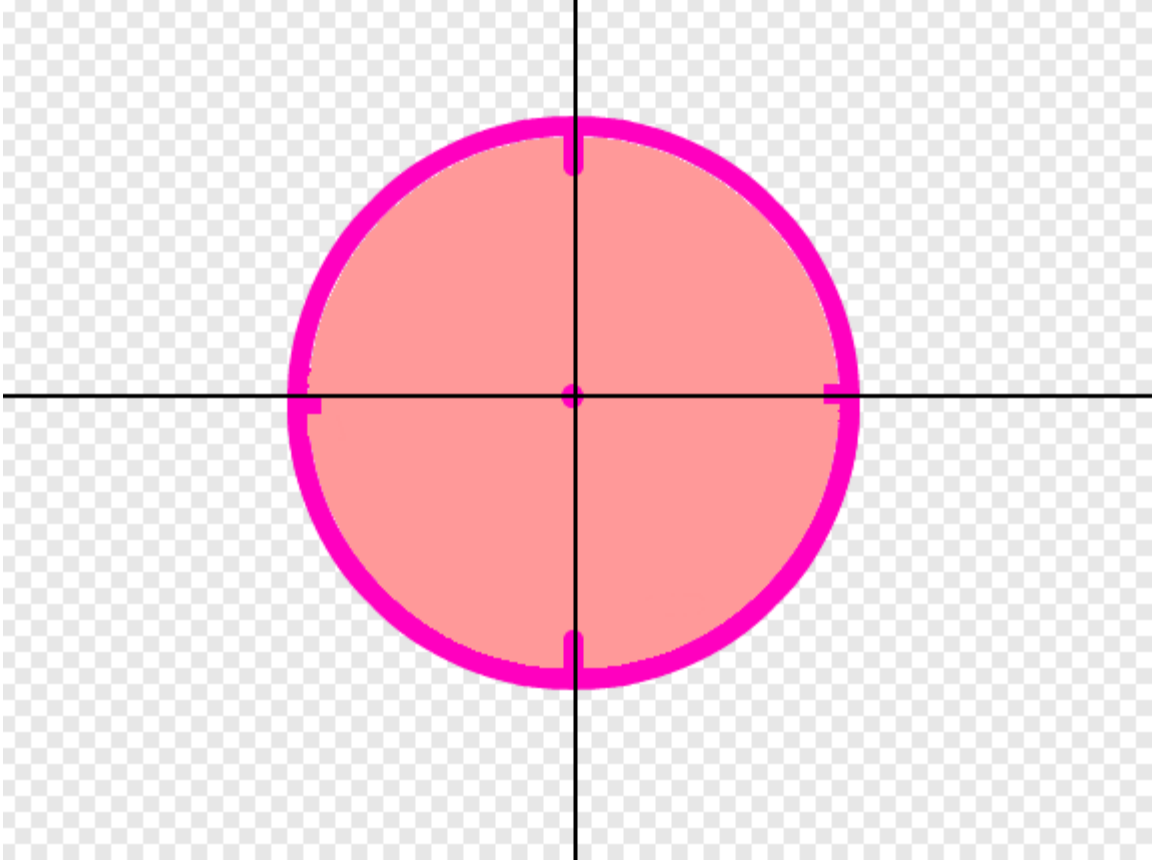
سنقوم بالمثال التالي بإنشاء ساعة حائط، لها ثلاثة عقارب. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسات البرمجية المخصصة لكل كائن.

لدينا الكائنات التالية :

- الساعة Sprite2.
- عقرب الثواني Seconds.
- عقرب الدقائق Minutes.
- عقرب الساعات Hours.



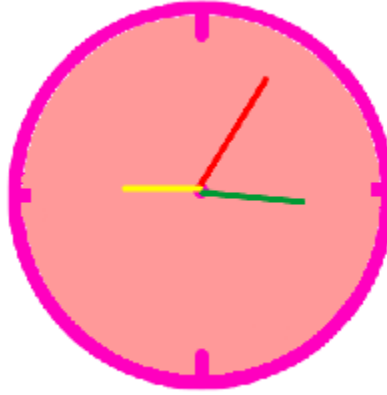
قمنا برسم الكائنات السابقة باستخدام المحرر. علينا الانتباه إلى عدد من القضايا أثناء إنشاء الكائنات في محرر الرسم سواء في هذا البرنامج أم في أي برنامج آخر، كأن يكون مركز الكائن Sprite2 هو مركز الدائرة، كما هو موضح بالشكل:



عند استخدام لبنة الدوران يدور الكائن حول مركزه، لذلك لجعل حركة الكائنات Seconds و Minutes و Hours تحاكي حركة عقارب الساعة يجب أن يكون مركزها في طرفها وحسراً في الطرف الذي يلامس مركز الدائرة. نضع المركز كما موضح في الصورتين التاليتين للعقارب الثلاثة:



لننتقل الآن إلى مرحلة تجميع الكائنات على المنصة والبدء ببرمجتها.



الكدسة التالية مسؤولة عن برمجة الكائن Sprite2. عند النقر على العلم الأخضر، يذهب الكائن دائماً إلى مبدأ الأحداث ثم ينتقل ثلاث طبقات إلى الخلف لضمان ظهور الكائنات الأخرى وهي العقارب أمامه.



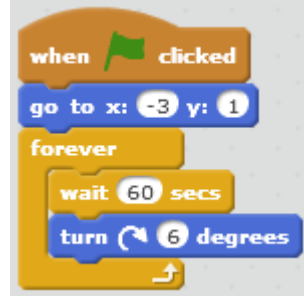
الكدسة البرمجية التالية مخصصة لبرمجة الكائن Seconds (عقرب الثواني)، فعند النقر على العلم الأخضر يذهب إلى الإحداثيات x:-3, y:1 ويكرر باستمرار الانتظار لمدة ثانية واحدة والدوران 6 درجات بالاتجاه غير المباشر (باتجاه دوران عقارب الساعة).



لماذا تم وضع قيمة 6 دون غيرها في لبنة الدوران؟

تم وضع القيمة 6 في لبنة الدوران لأن عقرب الثواني سيحتاج إلى 60 حركة لإتمام دورة كاملة؛ حيث كل حركة تعبر عن ثانية، ونعلم أن الدورة الكاملة تساوي 360 درجة وبالتالي $\frac{360}{60}=6$

الكدسة البرمجية التالية مخصصة لبرمجة الكائن Minutes (عقرب الدقائق)، فعند النقر على العلم الأخضر يذهب إلى الإحداثيات x:-3, y:1 ويكرر باستمرار الانتظار لمدة 60 ثانية ثم الدوران 6 درجات بالاتجاه غير المباشر (باتجاه دوران عقارب الساعة).



لماذا تم وضع قيمة 6 دون غيرها في لبنة الدوران؟

تم وضع القيمة 6 في لبنة الدوران لأن عقرب الدقائق سيحتاج إلى 60 حركة لإتمام دورة كاملة؛ حيث كل حركة تعبر عن دقيقة، ونعلم أن الدورة الكاملة تساوي 360 درجة وبالتالي $\frac{360}{60}=6$

الكدسة البرمجية التالية مخصصة لبرمجة الكائن Hours (عقرب الساعات)، فعند النقر على العلم الأخضر يذهب إلى الإحداثيات x:-3, y:1 ويكرر باستمرار الانتظار لمدة 3600 ثانية ثم الدوران 30 درجة بالاتجاه غير المباشر (باتجاه دوران عقارب الساعة).



لماذا تم وضع قيمة 30 دون غيرها في لبنة الدوران؟

تم وضع القيمة 30 في لبنة الدوران لأن عقرب الساعات سيحتاج إلى 30 حركة لإتمام دورة كاملة؛ حيث كل حركة تعبر عن ساعة، ولدينا 12 ساعة (تأشير) سيقف 3600 ثانية عند كل ساعة منها، ونعلم أن $\frac{360}{12}=30$ الدورة الكاملة تساوي 360 درجة وبالتالي

المثال السادس: الآلة الحاسبة

سنقوم بالمثال التالي بجعل الكائن يعمل كآلة حاسبة. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسة البرمجية التالية.



كيف نبدأ التفكير لتنفيذ مشروع كهذا؟
أولاً كيف يمكن للكائن أن يعمل كآلة حاسبة تقوم بعمليات الجمع والضرب والطرح والقسمة؟
نعلم أن الآلة الحاسبة التي تقوم بهذه العمليات تحتاج إلى ثلاث عمليات إدخال: عملية الإدخال الأولى للعدد الأول والثانية للإشارة (+, -, *, /) والثالثة للعدد الثالث.
ثانياً نفكر، ماهي لبنة الإدخال في السكراتش؟

لبنة ask and wait هي اللبنة التي تتيح للمستخدم الإدخال، وبما أنه يلزمنا ثلاث عمليات إدخال سنحتاج إلى ثلاث لبنات ask and wait وفي كل مرة يجيب المستخدم سنخزن جوابه في متحول.

ثالثاً كيف على الكائن أن يعرف نوع الإشارة المدخلة؟

يعرف الكائن نوع الإشارة المدخلة عن طريق أربع لبنات اختبار if، نختبر في كل منها إذا كانت الإشارة مساوية لأحد الإشارات الأربعة (+, -, *, /)، فمثلاً إذا كانت الإشارة (/) sign=/، يختبر الكائن وعندما يتحقق الشرط أي عندما يصل إلى لبنة if sign=/ يقول ناتج قسمة العدد الأول على العدد الثاني، وهكذا لجميع الإشارات.

لننتقل إلى تنفيذ ذلك في السكريبت. عند النقر على العلم الأخضر يسأل الكائن Enter a number وينتظر، وعندما يدخل له المستخدم قيمة يجعل قيمة المتحول Num1 مساوية للقيمة المدخلة. ثم يسأل مرة ثانية Enter the sign وينتظر حتى يدخل له المستخدم قيمة ويخزنها في المتحول Sign ويسأل مرة أخيرة Enter another number وينتظر حتى يدخل له المستخدم قيمة ليخزنها في المتحول Num2.

يبدأ بعدها الاختبار فإذا كانت الإشارة مساوية لـ + أو * أو - أو / أو % يقول لمدة ثانيتين العدد الأول + العدد الثاني أو العدد الأول * العدد الثاني أو العدد الأول - العدد الثاني أو العدد الأول / العدد الثاني أو العدد الأول % العدد الثاني على الترتيب. (%) وتعني باقي القسمة وفي حال إدخالها كإشارة في هذا البرنامج يقول الكائن لك ناتج قسمة العدد الأول على الثاني).

صديقي المتعلم لا بد أنك لاحظت أننا أضفنا إمكانية إدخال إشارة خامسة وهي إشارة باقي القسمة، وذلك فقط لتشجيعك على تطوير هذه الآلة الحاسبة بعمليات أخرى، لماذا لا تضيف إمكانية حساب sin أو cos زاوية، أو لماذا لا تفكر بتعيين سلوك الكائن عند إدخال عملية ليست في الحسبان أي لم تذكر ضمن الشروط؛ كأن يدخل المستخدم (٨) وهي إشارة الأس؟

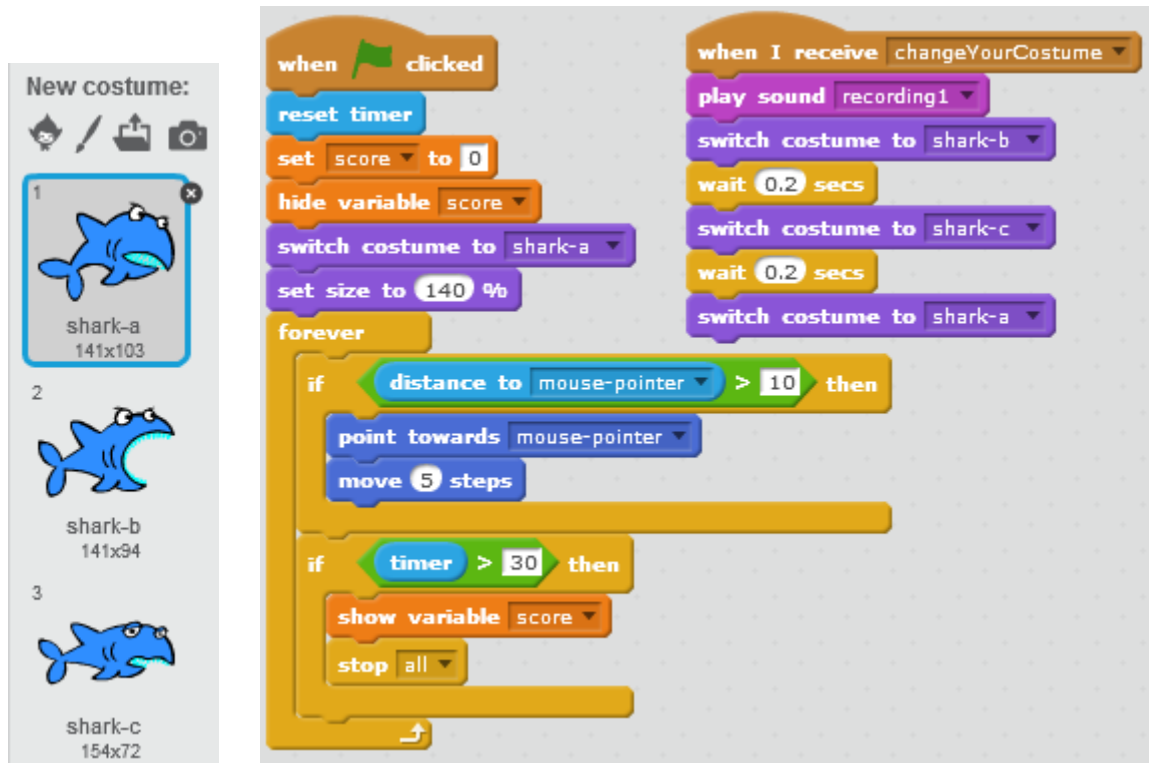
المثال السابع: أعماق البحار

سنقوم بالمثال التالي بتصميم لعبة في حوض سمك، يمكن للمستخدم في هذه اللعبة أن يتحكم بحركة سمكة القرش خلال وقت اللعبة وهو نصف دقيقة، وعليه أن يأكل الكائنات البحرية (ثلاث سمكات وسرطان) المتواجدة في أعماق البحار، في كل مرة يلامس فم سمكة القرش أي من الكائنات البحرية يفتح فمه (يغير مظهره) ويصدر صوت قرمشة (بإمكانك تسجيله) وتختفي السمكة لتعاود الظهور بعد 3 ثوانٍ وتزداد النتيجة بمقدار 10. وعند انتهاء وقت اللعبة تتوقف جميع الكائنات عن الحركة وتظهر النتيجة في منتصف المنصة علماً أنها تكون مخفية خلال سير اللعبة، بينما يظهر وقت اللعبة في الزاوية العليا. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسات البرمجية.

لدينا هذه الكائنات التالية والخلفية صورة التقطها القط سكراتش في أعماق البحار.



الكدستان البرمجتان الآتيتان مسؤولتان عن التحكم بسمكة القرش Shark ذي المظاهر الثلاثة المبينة في الصورة، لاحظ لون أسنانها.



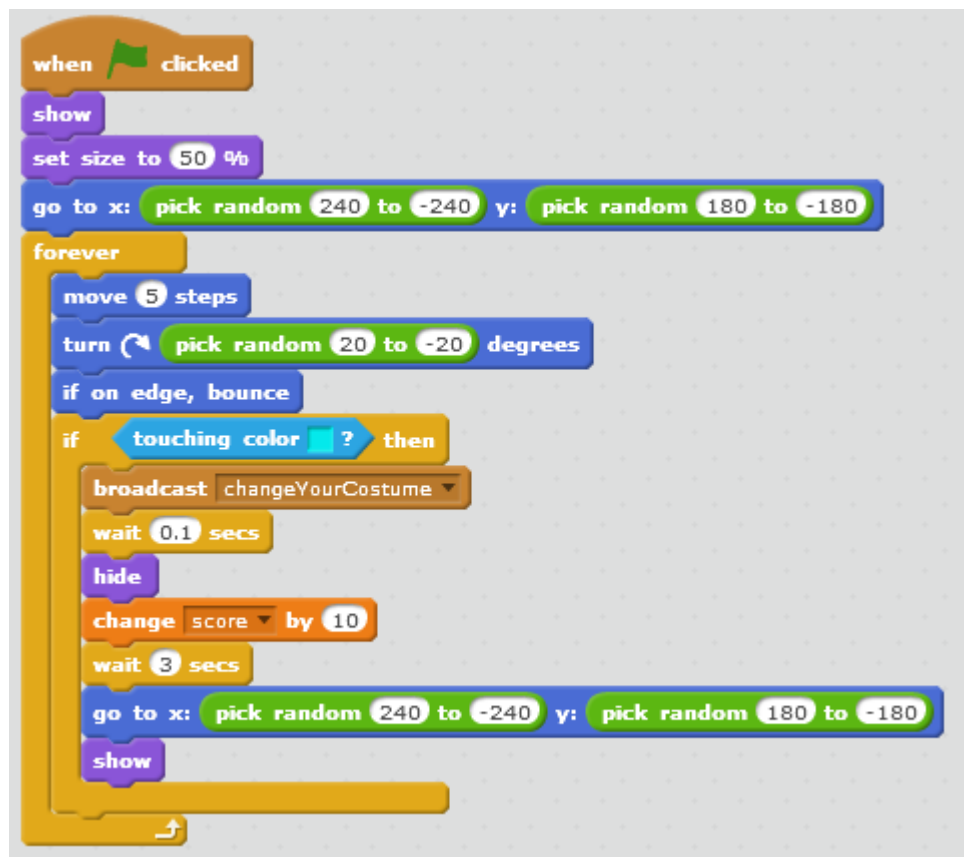
عند الضغط على العلم الأخضر يتم جعل المؤقت مساوياً للصفر وكذلك جعل متحول الـ score مساوياً للصفر مع إخفاء هذا المتحول عن المنصة ثم يأخذ الكائن المظهر shark-a ويجعل حجم القرش 140%. يكرر باستمرار الاختبارين التاليين: الاختبار الأول إذا كانت المسافة بين الكائن وبين مؤشر الفأرة <10 فيجعل اتجاهه باتجاه مؤشر الفأرة ويتحرك 5 خطوات، أما الاختبار الثاني فيختبر الكائن إذا كان المؤقت <30 عندها يعرض متحول الـ score ويوقف البرنامج.

لدينا الكدسة البرمجية الثانية؛ عندما يتلقى الكائن رسالة `changeYourCostume` يبدأ بالتنفيذ ويصدر صوت القرمشة الذي قمنا بتسجيله، ثم يغير مظهره إلى المظهر `shark-b` ثم المظهر `shark-c` ثم يعود إلى المظهر `shark-a` علماً أنه ينتظر لمدة 0.2 بين كل مظهرين.

لماذا وضعنا شرط توقف اللعبة المؤقت $30 <$ ولم نضع المؤقت $= 30$ ؟

لقد قمنا بوضع المؤقت $30 <$ لضمان توقف البرنامج، فمن الممكن أن يصبح المؤقت مساوٍ 30 عندما يكون الكائن في لحظة الانتظار 0.2، عندها لن ينفذ التعليمات المحتواة في شرط المؤقت ولن يتوقف البرنامج.

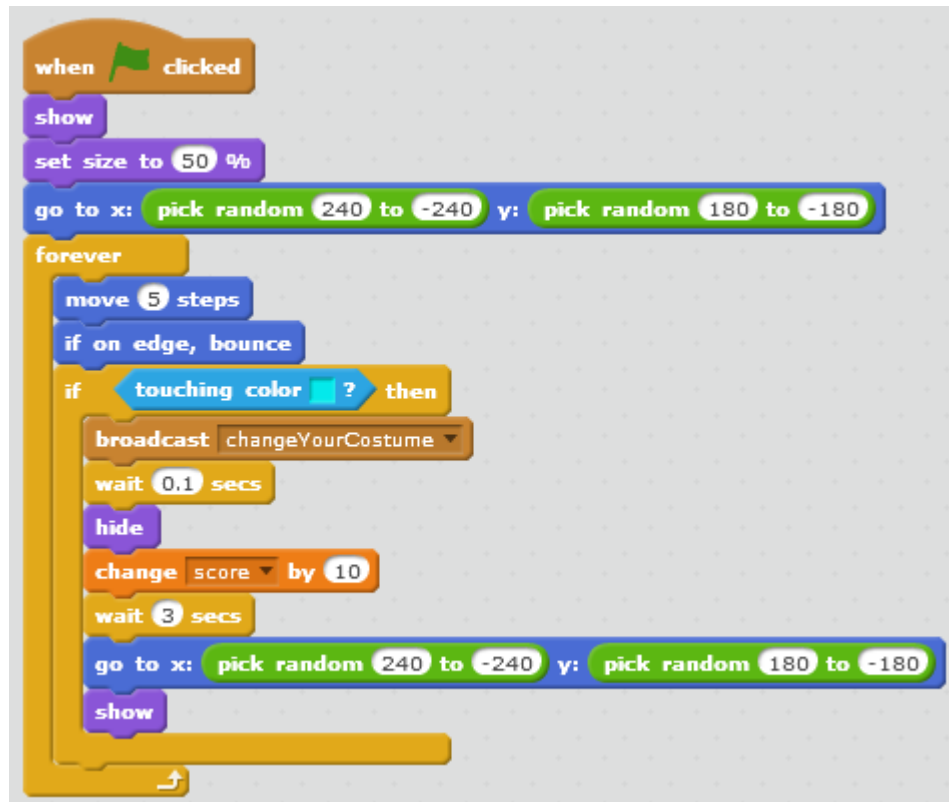
الكدسة البرمجية الآتية مسؤولة عن التحكم بالكائنات `Fish1`, `Fish2`, `Fish3`.



عند النقر على العلم الأخضر، يظهر الكائن ويصبح حجمه 50% ثم يذهب إلى نقطة عشوائية من المنصة، ويكرر باستمرار التحرك خمس خطوات ويدور زاوية عشوائية يتراوح مقدارها بين (20 و -20) وفي حال اصطدم بالحافة يرتد، ويختبر إذا كان يلامس اللون السماوي (لون فم السمكة) عندها يقوم بإرسال `changeYourCostume` (الرسالة التي تتلقاها سمكة القرش لتصدر صوت القرمشة وتغير مظهرها)

وينتظر مدة قصيرة جداً مقدارها 0.1 ثانية لضمان عدم حدوث أي خطأ ثم يختفي ويزيد متحول الـ score بمقدار 10 وينتظر ثلاث ثوانٍ ويذهب إلى نقطة عشوائية من المنصة ثم يعاود الظهور من جديد.

الكدسة البرمجية الآتية مسؤولة عن التحكم بالسرطان Crap.



عند النقر على العلم الأخضر، يظهر الكائن ويصبح حجمه 50% ثم يذهب إلى نقطة عشوائية من المنصة، ويكرر باستمرار التحرك خمس خطوات وفي حال اصطدم بالحافة يرتد، ويختبر إذا كان يلامس اللون السماوي (لون فم السمكة) عندها يقوم بإرسال changeYourCostume (الرسالة التي تتلقاها سمكة القرش لتصدر صوت القرمشة وتغير مظهرها) وينتظر مدة قصيرة جداً مقدارها 0.1 ثانية لضمان عدم حدوث أي خطأ ثم يختفي ويزيد متحول الـ score بمقدار 10 وينتظر ثلاث ثوانٍ ويذهب إلى نقطة عشوائية من المنصة ثم يعاود الظهور من جديد.

ملاحظات:

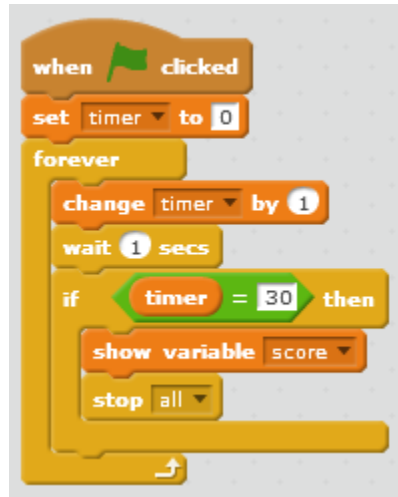
- الفرق بين لبنات السرطان ولبنات السمك أن السرطان لا يدور.

- لقد قمنا بجعل حجم الكائنات (السماك والسرطان) 50% في بداية البرنامج لأن الحجم الافتراضي لهذه الكائنات المدرجة من مكتبة السكراتش قريب من حجم سمكة القرش، وهذا لا يتوافق مع طبيعة مشروعنا.

لإظهار المؤقت عليك التوجه إلى Sensing وتفعيل ظهور الـ timer على المنصة، كما هو موضح بالشكل التالي:

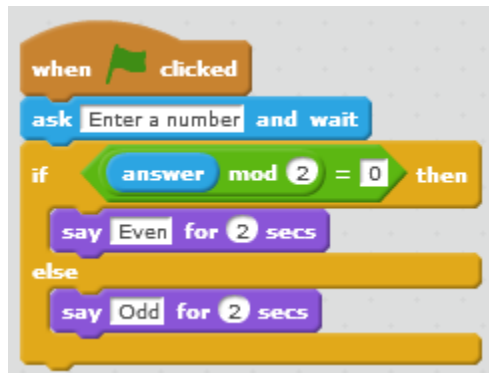


يمكنك تحسين هذه اللعبة وإضافة لمساتك الخاصة إليها، فمثلاً يمكنك تعريف متحول وتسميته timer (بدلاً من استخدام المؤقت الموجود في البرنامج) وزيادته بمقدار واحد كل ثانية، كما هو موضح بالشكل التالي:



المثال الثامن: زوجي ام فردي؟

سنقوم بالمثال التالي بجعل الكائن يطلب من المستخدم إدخال عدد، ثم يخبرنا الكائن إذا كان هذا العدد زوجي Even أم فردي Odd. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسة البرمجية التالية.



سأل القط سكراتش المعلم طلابه متى يكون العدد زوجي؟

أجابه أحد الطلاب: عندما يكون رقم أحاده زوجي.

ابتسم القط المعلم وقال له: صحيح، لكن كيف يمكننا أن نعبر عن ذلك في السكراتش؟

كتب القط المعلم على اللوح ما يأتي:

$$24/2=12 \text{ (والباقي 0)}$$

$$13/2=6 \text{ (والباقي 1)}$$

قال له أحد الطلاب: العدد 24 زوجي وباقي قسمته على 2 تساوي الصفر، وإذا قسمنا أي عدد زوجي على 2 نجد أن باقي القسمة سيكون صفر. بينما العدد 13 فردي وباقي قسمته على 2 تساوي الواحد، وكذلك إذا قسمنا أي عدد فردي على 2 نجد أن باقي القسمة سيكون واحد.

إذاً: كما قال أحد طلاب القط سكراتش العدد الزوجي باقي قسمته على 2 تساوي الصفر أما العدد الفردي فباقي قسمته على 2 تساوي الواحد.

لننتقل إلى تنفيذ ذلك في السكراتش. عند النقر على العلم الأخضر، يطلب الكائن من المستخدم إدخال عدد Enter a number ثم يختبر، فإذا كان باقي قسمة العدد المدخل answer على 2 تساوي 0 يقول الكائن Even وتعني زوجي، أما إذا كان باقي القسمة غير ذلك (وهو حتماً سيكون 1) فيقول الكائن Odd وتعني فردي.

المثال التاسع: الأعداد الزوجية في مجال

سنقوم بالمثال التالي بجعل الكائن يطلب من المستخدم إدخال بداية ونهاية مجال، ثم نخبرنا الكائن بمجموعة الأعداد الزوجية الموجودة ضمن هذا المجال. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسة البرمجية التالية.



أولاً نحن بحاجة إلى متحولين، الأول يخزن قيمة بداية المجال والثاني يخزن قيمة نهاية المجال.

عند النقر على العلم الأخضر يطلب الكائن من المستخدم إدخال قيمة لتكون بداية المجال Enter the beginning of the range ويخزن القيمة المدخلة في المتحول firstNum ثم يطلب من المستخدم إدخال قيمة لتكون نهاية المجال Enter the end of the range ويخزن القيمة المدخلة في المتحول SecondNum. لدينا حلقة تستمر بالتكرار حتى يتحقق الشرط: $SecondNum < firstNum$ ، يختبر الكائن هل المتحول firstNum زوجي، أي هل باقي قسمته على 2 يساوي الصفر؟ إذا كان الشرط محقق (إذا كان زوجي) يقول هذا العدد، ويزيد متحول الـ firstNum بمقدار 1 سواء تحقق الشرط أم لم يتحقق. وعندما يصبح $SecondNum < firstNum$ محقق ينتهي البرنامج.

لماذا وضعنا شرط التكرار حتى $SecondNum < firstNum$ وليس $SecondNum = firstNum$ ؟
قمنا بوضع شرط التكرار حتى $SecondNum < firstNum$ من أجل أن يقول الكائن قيمة نهاية المجال في حال كانت هذه القيمة زوجية.

الآن صديقي المتعلم لماذا لا تضيفي لمساتك الخاصة علل هذا المشروع أو تنشئ مشروع لعرض الأعداد الفردية ضمن مجال يدخله المستخدم؟

المثال العاشر: خمن الرقم الذي اختاره القط

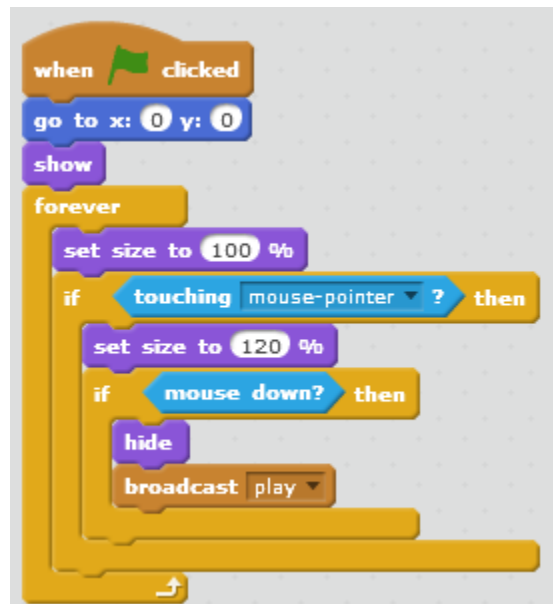
سنقوم بالمثال التالي بجعل الكائن يطلب من المستخدم إدخال بداية ونهاية مجال، ثم يختار الكائن عدد موجود ضمن هذا المجال ويجعل المستخدم يخمنه، يمكن للمستخدم القيام بثلاث محاولات فقط.

لدينا الكائنات التاليان:



في بداية البرنامج يظهر الكائن PLAY وعندما يلامس مؤشر الفأرة يصبح حجمه 120% (يزيد 20% عن الحجم الأصلي) وعند الضغط عليه يختفي ليظهر الكائن Cat. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسات البرمجية التالية.

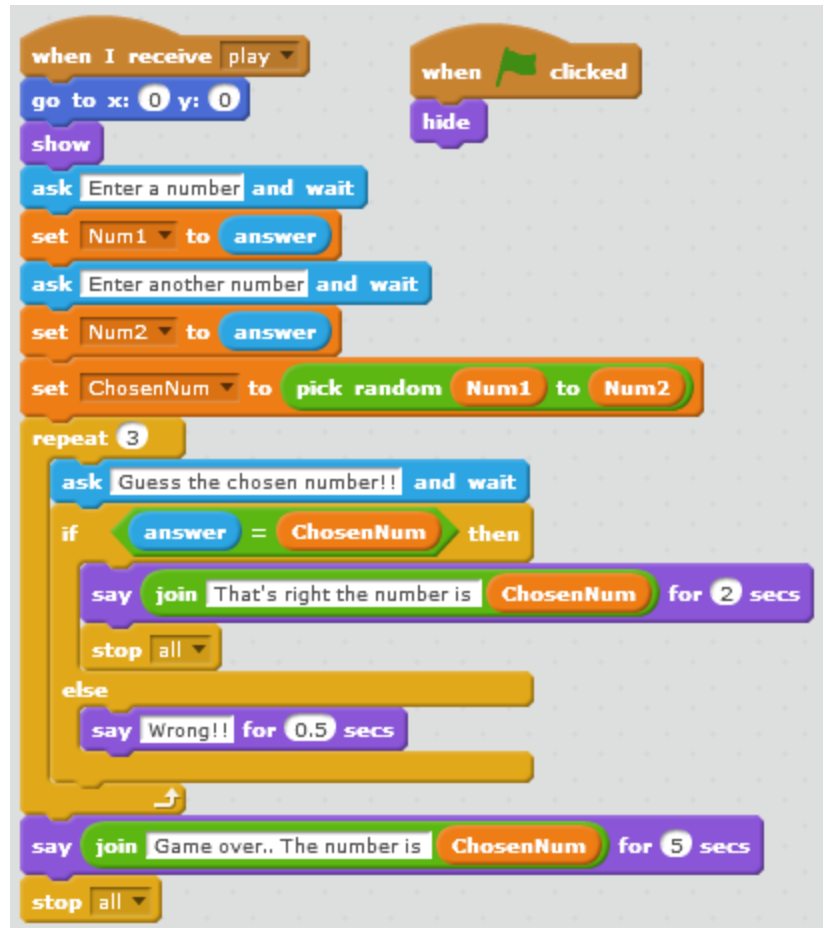
الكدسة البرمجية لبرمجة الكائن PLAY



عند النقر على العلم الأخضر، يذهب هذا الكائن إلى مبدأ الإحداثيات $x:0, y:0$ ويظهر، يكرر باستمرار جعل حجمه مساوٍ للـ 100% (الحجم الطبيعي) ثم يختبر هل هو يلامس مؤشر الفأرة؟ في حال كان يلامسه يزداد حجمه (20%) ليصبح 120% وأثناء ملامسته لمؤشر الفأرة يختبر هل زر الفأرة مضغوط (هل المستخدم يريد الضغط على هذا الكائن وبدء اللعبة)؟ في حال كان مضغوط يختفي هذا الكائن ويرسل رسالة play إلى الكائن cat مشيراً إليه أن يبدأ اللعبة.

لماذا وضعنا لبنة _ set size to ضمن حلقة forever ولم نضعها في بداية البرنامج خارج الحلقة؟ لقد قمنا بوضعها بهذا الشكل من أجل عودة الكائن إلى حجمه الأصلي (100%) عند عدم ملامسته لمؤشر الفأرة.

الكدسة البرمجية لبرمجة الكائن Cat

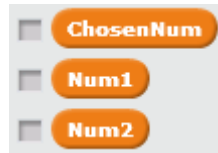


نلاحظ وجود كدستين برمجيتين لهذا الكائن، إحداهما تنفذ عند النقر على العلم الأخضر وتخفي الكائن، أما الثانية فتنفذ عندما تستقبل رسالة play من الكائن PLAY. يطلب الكائن من المستخدم إدخال رقم Enter a number ويجعل المتحول Num1 يخزن قيمة الجواب المدخل، ثم يطلب منه مرة ثانية أن يدخل رقم آخر Enter another number ، ويخزن قيمة الجواب المدخل في المتحول Num2. الآن انتهينا من عملية إدخال حدود المجال وسنبداً بالمرحلة التالية. يجعل الكائن المتحول ChosenNum يخزن قيمة عشوائية تتراوح ضمن المجال المدخل أي ضمن Num1 و Num2. تبدأ الآن عملية تخمين المستخدم للرقم الذي اختاره الكائن، يكرر ثلاث مرات (لأن للمستخدم ثلاث محاولات إدخال فقط) ويطلب منه تخمين الرقم المختار Guess the chosen number!! . بعد أن يدخل المستخدم قيمة ما يختبر هل هذه القيمة تساوي الرقم المختار ChosenNum، في حال كانت تساويها يقول الكائن That's right the number is __ ويقول الرقم عن طريق لبنة join لمدة ثانيتين ويوقف البرنامج.

أما else ونقصد في حال القيمة المدخلة لا تساوي الرقم المختار ChosenNum فيقول الكائن Wrong لمدة 0.5 ثانية ويعود ويطلب من المستخدم إدخال قيمة جديدة في حال لم تتجاوز محاولات الإدخال ثلاث محاولات.

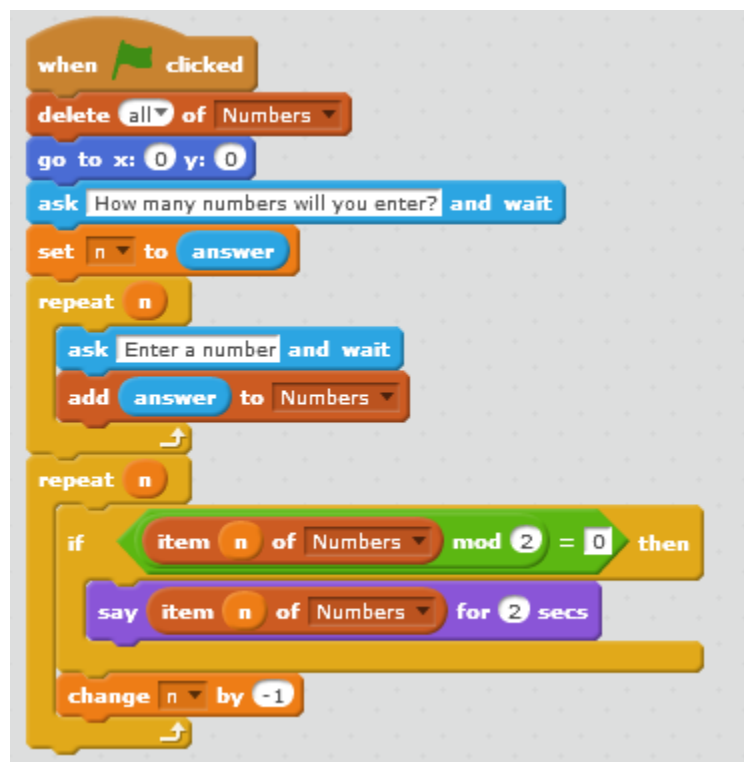
في حال انتهت المحاولات الثلاثة ولم يخمن المستخدم الرقم المختار من قبل الكائن، يقول الكائن
Game over.. The number is __ ويقول الرقم المختار باستخدام لبنة join لمدة خمس ثوانٍ ثم
يتوقف البرنامج.

تنبيه: لا تنسَ أن تقوم بإخفاء متحول ChosenNum عن المنصة. لقد قمنا بإخفاء المتحولات الثلاثة عن
المنصة بإزالة تفعيل ظهورها كما موضح في الشكل التالي:



المثال الحادي عشر: الأعداد الزوجية ضمن مجموعة

قمنا في مثال سابق بعرض الأعداد الزوجية ضمن مجال يحدده المستخدم. أما في هذا المثال نهدف إلى
جعل المستخدم يحدد عدد العناصر التي سيدخلها، ثم يدخل هذه العناصر، وعند الانتهاء من إدخاله
للعناصر، على الكائن أن يقوم بعرض الزوجية منها فقط. حاول أن تنفذ ذلك بنفسك دون الاطلاع على
الكدسة البرمجية التالية.



عند النقر على العلم الأخضر، يقوم الكائن بحذف جميع عناصر المجموعة Numbers ويذهب الكائن إلى مبدأ الإحداثيات $x:0, y:0$ ويسأل المستخدم عن عدد الأعداد التي سيدخلها How man numbers will you enter? ويجعل المتحول n يخزن قيمة الجواب.

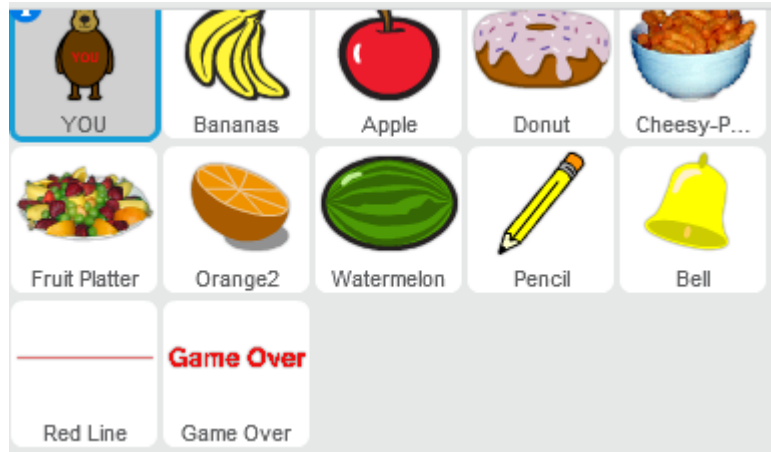
لجعل المستخدم يدخل عدد العناصر الذي حدده نقوم بتكرار n مرة طلب الكائن من المستخدم إدخال رقم وإضافته إلى المجموعة Numbers.

عند الانتهاء من إدخال عناصر المجموعة Numbers، تبدأ مرحلة اختبار كل عنصر هل هو زوجي أم لا. سنستخدم المتحول n كعداد، سنكرر n مرة اختبار هل العنصر n من المجموعة Numbers باقي قسمته على 2 مساو للصفر (هل العنصر n من المجموعة Numbers زوجي كما أوضحنا في مثال سابق)، في حال كان زوجي يقول العنصر n من المجموعة Numbers لمدة ثانيتين، وفي كل الأحوال سواء كان زوجي أم فردي ينقص الـ n بمقدار واحد وبذلك ينتقل إلى العنصر الذي يسبقه، ويجب أن نوضح هنا أن الكائن سيقوم بعرض الأعداد الزوجية من نهاية المصفوفة إلى بدايتها أي بشكل تنازلي لأننا بدأنا الاختبار من آخر عنصر في المصفوفة (العنصر n قبل نقصانه أي القيمة التي أدخلها المستخدم كعدد عناصر للمجموعة). لماذا لا نضيف لمساتك إلى هذا المشروع كأن تجعل الكائن يقول الأعداد الزوجية بشكل تصاعدي أي من بداية المصفوفة إلى نهايتها؟

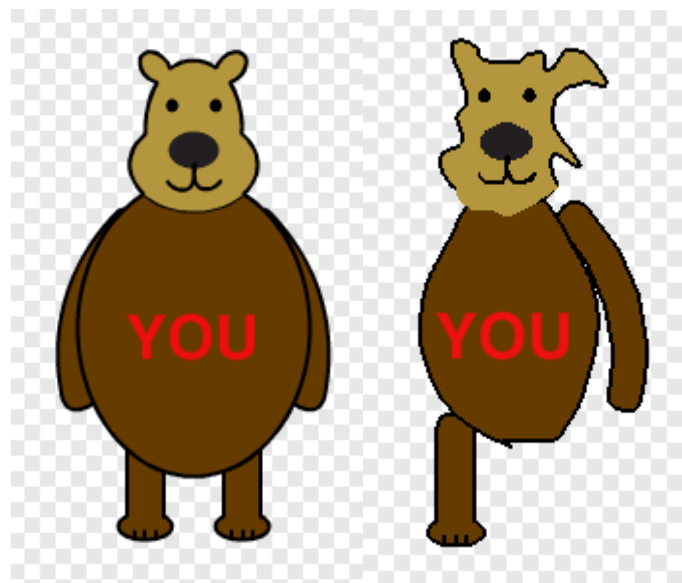
المثال الثالث عشر: إطعام الدب الجائع

سنقوم بالمثال التالي بتصميم لعبة إطعام الدب الجائع، يمكن للمستخدم أن يحرك الدب عن طريق تحريك الفأرة مع العلم أن الدب لا يغير موقعه على محور الترتيب. تبدأ المأكولات Bananas, Apple, Donut, Cheesy-Puffs, Fruit Platter, Orange2, Watermelon بالتساقط من أعلى المنصة بعد الضغط على العلم الأخضر بمدة زمنية تتراوح بين 1-8 ثانية، تستمر هذه المأكولات بالتساقط إلى أن تصطدم بالحاجز ذي اللون الأحمر Red Line أو أن تصطدم بالدب، فإذا اصطدمت بفم الدب يزداد الـ Score بمقدار 10 ويختفي الكائن ويعاود الظهور بعد مدة زمنية تتراوح بين 4-8 ثانية. عندما يساوي الـ Score قيمة الـ 100 يُنقل المستخدم إلى المرحلة التالية مع العلم أن رقم المرحلة يكون 1 في بداية البرنامج. يوجد كائنين Bell و Pencil تتساقط مع المأكولات، في حال اصطدمت إحداها بالدب يتناقص كل من الـ Score والـ Health بمقدار 10. مع العلم أن قيمة الـ Health في بداية البرنامج تكون 100، ويجب جعل الـ Score مساوياً للصفر في بداية كل مرحلة، لذلك على المستخدم أن يحاول ألا يجعل الدب يأكل Bell أو Pencil مطلقاً في بداية أي مرحلة. تظهر عبارة Game Over على المنصة إذا كانت قيمة الـ Health مساوية للـ 0 أو في حال أصبحت قيمة الـ Score أصغر من الـ 0 مع توقف حركة الدب وتغير مظهره. حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسات البرمجية.

لدينا الكائنات التالية:



مظهري الكائن YOU:



الكدسة البرمجية الخاصة ببرمجة الكائن YOU:



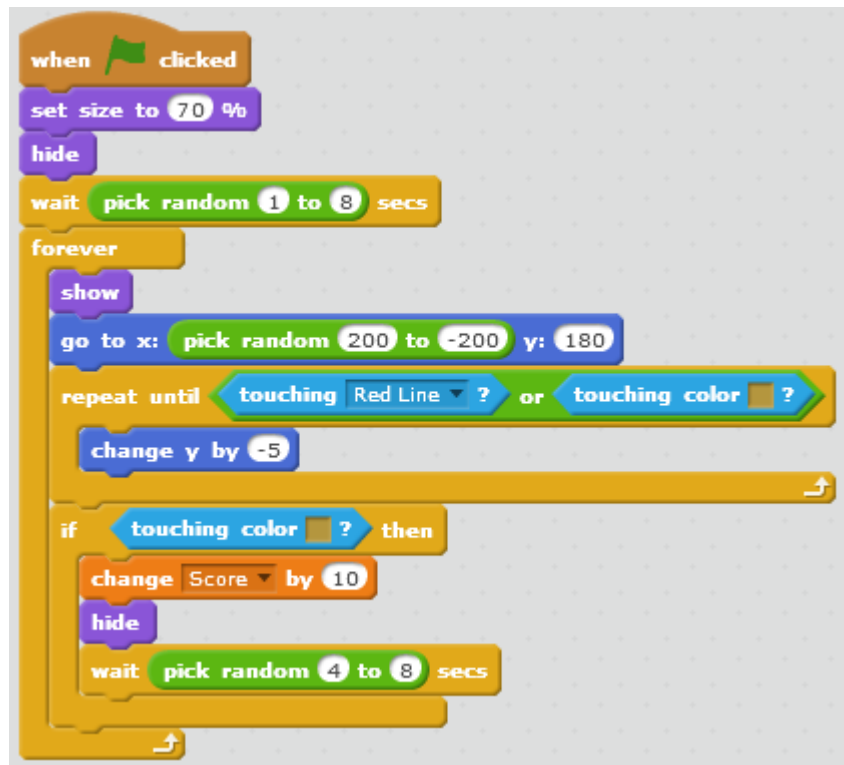
عند النقر على العلم الأخضر يجعل الكائن مظهره bear2-a ويبدأ بتهيئة المتحولات، يجعل قيمة متحول Score مساوية للصفر وقيمة المتحول Health مساوية للمئة وقيمة المتحول Level مساوية للواحد، يمكن تهيئة المتحولات السابقة في أي كائن تريده من البرنامج. يكرر باستمرار التعليمات التالية: يجعل ترتيب الكائن 112- ويجعل فاصلته مساوية لفاصلة مؤثر الفأرة، يختبر هل قيمة الـ Score مساوية للـ 100؟ في حال كان هذا الشرط محقق يجعل قيمة الـ Score صفراً ويزيد الـ Level بمقدار واحد أي ينقل المستخدم إلى المرحلة التالية ويزيد الـ Health بمقدار 10. يختبر أيضاً هل الـ Health مساوياً للصفر أو هل الـ Score أصغر من الصفر؟ في حال تحقق هذا الشرط يخسر الكائن أي يغير مظهره إلى المظهر bear2-a2 ويرسل رسالة game over ويوقف عمل هذا الكائن فقط.

إلى أي كائن يقوم الدب بإرسال رسالة game over؟
يقوم الدب بإرسال رسالة game over إلى الكائن Game Over ذي الكدستين البرمجتين التاليتين:



عند النقر على العلم الأخضر يذهب الكائن إلى مبدأ الإحداثيات ولكنه يكون مختفٍ، يبقى مختفياً إلى أن يستقبل رسالة game over فعندما يستقبلها يظهر ويجعل حجمه 100% ويكرر باستمرار زيادة ونقصان الحجم بمقدار 20 والانتظار لمدة ثانية واحدة بين كل من العمليتين السابقتين.

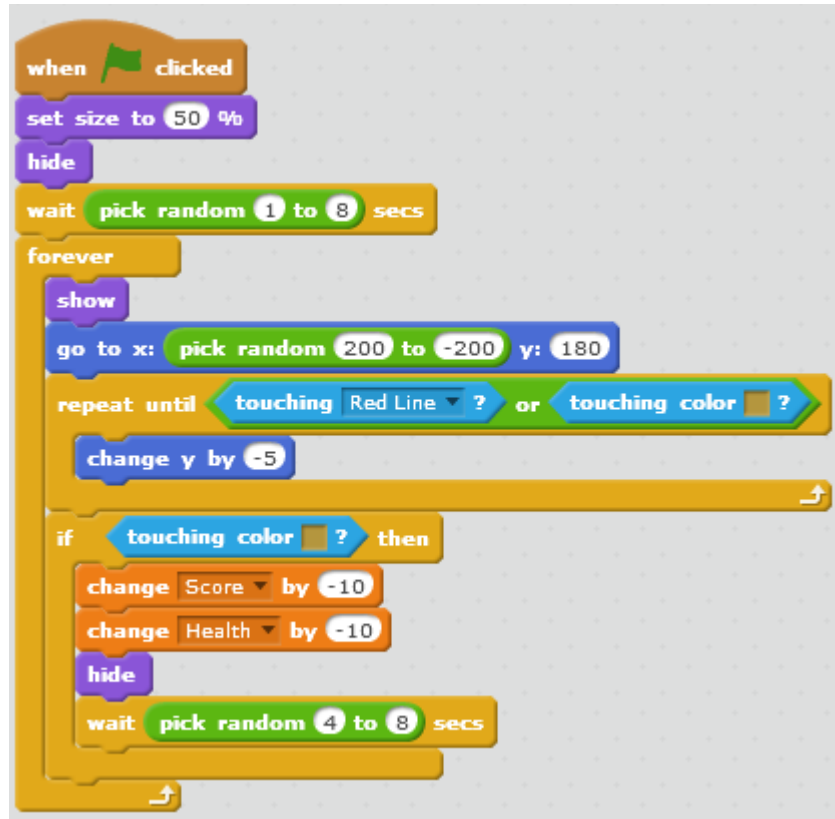
الكدسة البرمجية الخاصة بالكائنات Bananas, Apple, Donut, Cheesy-Puffs, Fruit Platter, Orange2, Watermelon :



عند النقر على العلم الأخضر يجعل الكائن حجمه 70% ويختفي وينتظر مدة عشوائية تتراوح بين 1-8 ثانية ليكرر باستمرار: الظهور والذهاب إلى فاصلة عشوائية تتراوح بين 200, -200 والترتيب الثابت:

180. يسقط نحو الأسفل بمقدار 5 خطوات حتى يلامس الحاجز ذي اللون الأحمر Red Line أو حتى أن يلامس اللون البني وهو لون فم الدب، فإذا اصطدم بهذا اللون (البني) يزداد الـ Score بمقدار 10 ويختفي الكائن ويعاود الظهور بعد مدة زمنية عشوائية تتراوح بين 4-8 ثانية.

الكدسة البرمجية الخاصة بالكائنين Pencil و Bell:



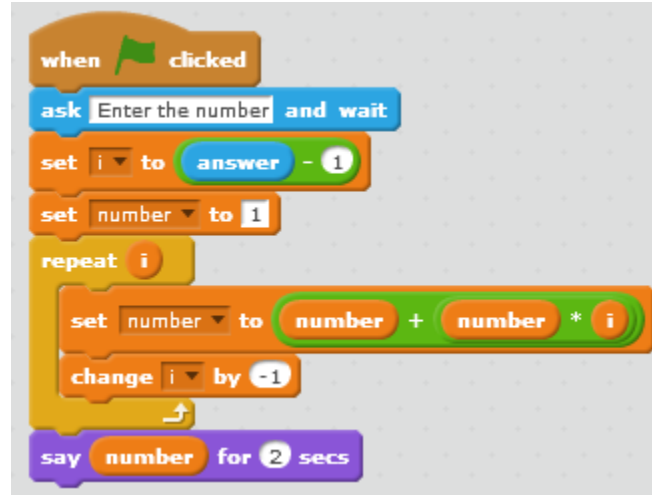
عند النقر على العلم الأخضر يجعل الكائن حجمه 50% ويختفي وينتظر مدة عشوائية تتراوح بين 1-8 ثانية ليكرر باستمرار: الظهور والذهاب إلى فاصلة عشوائية تتراوح بين 200، -200 والترتيب: 180. يسقط نحو الأسفل بمقدار 5 خطوات حتى يلامس الحاجز ذي اللون الأحمر Red Line أو حتى أن يلامس اللون البني وهو لون فم الدب، فإذا اصطدم بهذا اللون (البني) يتناقص كل من الـ Score والـ Health بمقدار 10 ويختفي الكائن ويعاود الظهور بعد مدة زمنية عشوائية تتراوح بين 4-8 ثانية.

لدينا الحاجز الأحمر، عند النقر على العلم الأخضر يذهب دوماً إلى الإحداثيات x:-20, y:-180 كما موضح بالشكل التالي:



المثال الرابع عشر: العامل $n!$

نهدف في هذا المشروع إلى جعل المستخدم يدخل رقم، ثم يقول له الكائن العامل لهذا العدد. كيف نبدأ التفكير لتنفيذ هذا المشروع؟
 أولاً: علينا معرفة العامل $n!$ أو $n!$ عامل، هو جداء جميع الأعداد الصحيحة الموجبة والأصغر أو تساوي n . مثلاً: $5! = 5 * 4 * 3 * 2 * 1 = 120$
 الآن بعد وضوح فكرة البرنامج لديك، حاول تنفيذه دون الاطلاع على الكدسة البرمجية التالية.



عند النقر على العلم الأخضر يطلب الكائن من المستخدم إدخال رقم Enter the number، لدينا المتحول i الذي هو بمثابة عدّاد في هذا البرنامج، نجعل قيمته مساوية لقيمة العدد المدخل (الجواب) مطروحاً منه واحد. ثم لدينا المتحول $number$ نجعل قيمته مساوية لـ 1. ولدينا حلقة تكرار i مرة ما يلي: جعل قيمة المتحول $number$ مساوية لقيمته الحالية مضافاً إليها جداء قيمته بـ i ، وإنقاص قيمة i بمقدار 1. عندما تنتهي حلقة التكرار، أي عندما يكرر i مرة يقول الكائن قيمة المتحول $number$ أي قيمة العامل للعدد المدخل لمدة ثانيتين.

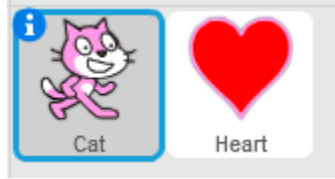
المثال الخامس عشر: مقياس الحب

نهدف في هذا المشروع إلى جعل المستخدم يُدخل اسمين، ثم يقول الكائن له نسبة التوافق بين هذين الاسمين المدخلين. يجب أن نراعي في هذا البرنامج ما يلي:

- عند إدخال اسمين أكثر من مرة يجب في كل مرة أن يعطي الكائن القيمة نفسها.
- يجب أن يتغير حجم الكائن Heart أثناء تشغيل البرنامج.

حاول أن تنفذ ذلك بنفسك قبل الاطلاع على الكدسات البرمجية المخصصة لكل كائن.

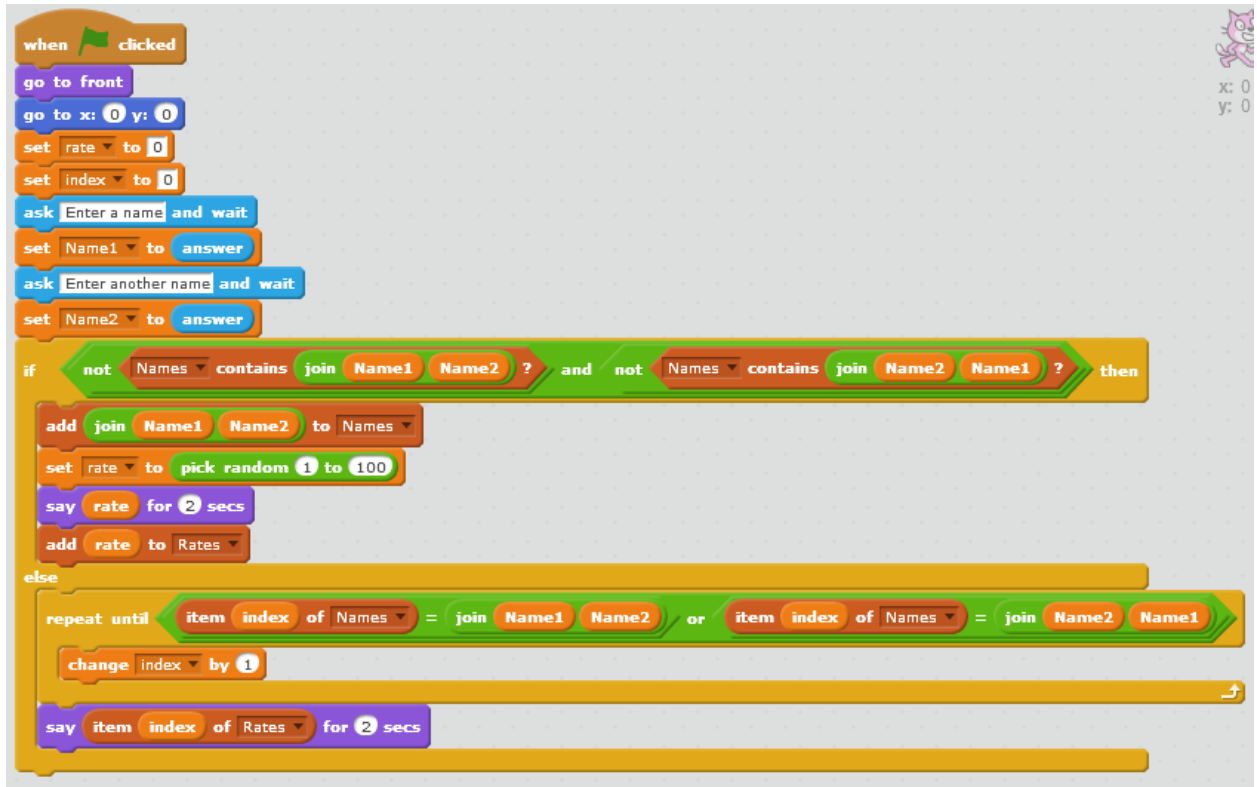
لدينا الكائنات الآتية:



عند النقر على العلم الأخضر، يذهب الكائن Cat إلى الأمام لضمان ظهوره وعدم اختفائه خلف الكائن Heart، ثم يذهب إلى مبدأ الإحداثيات ويجعل قيمة كل من المتحولين index و rate صفراً. ويطلب من المستخدم إدخال اسم Enter a number ويخزن هذا الاسم المدخل في المتحول Num1 ثم يطلب منه إدخال اسم آخر Enter another number ليخزنه في المتحول Num2 وبذلك نكون انتهينا من عملية الإدخال وتخزين الأسماء.

ملاحظة: باستخدام لبنة join سنضم الاسمين المدخلين ليشكلا سلسلة حرفية واحدة وسنخزن هذه السلسلة في قائمة، وبذلك نستطيع الاختبار بسهولة في حال كانت هذه السلسلة موجودة أم لا، مع مراعاة أن المستخدم قد يدخل في تجربة الاسم الأول a والاسم الثاني b، وفي تجربة أخرى الاسم الأول b والاسم الثاني a عندها يجب أن يعرض الكائن نفس النتيجة في كلي التجريبتين.

لننتقل الآن إلى اختبار إذا كان الاسمان مُدخلين مسبقاً، في حال كانا غير مدخلين مسبقاً، أي المجموعة Names تحتوي السلسلة الحرفية Name1+Name2 أو Name2+Name1، يضم المتحولين Name1 و Name2 باستخدام لبنة join ليشكلا سلسلة حرفية واحدة، ثم يضيف هذه السلسلة إلى القائمة Names، يولد بعد ذلك قيمة عشوائية تتراوح بين 1 و100 ويخزنها في المتحول rate، يقول قيمة هذا المتحول لمدة ثانيتين ثم يضيف هذه القيمة إلى القائمة Rates. أما في حال لم يتحقق الشرط السابق، أي إذا كان هذان الاسمان مدخلين مسبقاً، يكرر زيادة المتحول index بمقدار واحد حتى يتحقق الشرط التالي: أن يكون العنصر الذي رقمه index من القائمة Names مساوٍ للسلسلة الحرفية Name1+Name2 أو Name2+Name1 وعندما يتحقق الشرط السابق يقول الكائن العنصر الذي رقمه index من القائمة Rates.



نلاحظ أننا جَمَعْنَا كل اسمين مدخلين في سلسلة واحدة لها فهرس ثابت ضمن القائمة Names، والقيمة التي عرضها الكائن لكل اسمين لها نفس فهرس سلسلة الاسمين لكن في القائمة Rates، والشكل التالي يوضح ذلك:

تم إدخال الاسمين اللذين تم إدخالهما مسبقاً في التجربة الثانية وكانت النتيجة ذاتها في التجريبتين

The image shows a Scratch workspace with a large red heart in the center. A pink cat is running towards the heart. Above the heart, there are two data monitors. The 'Names' monitor on the left has three rows: '1 ClaraTom', '2 AmandaMatt', and '3 RalfKaty'. The 'Rates' monitor on the right has three rows: '1 58', '2 46', and '3 100'. A yellow circle highlights the value '46' in the 'Rates' monitor, and a yellow arrow points from this circle to a speech bubble containing the number '46'. A red box highlights the 'Name1' and 'Name2' labels in the top right corner of the workspace.

Names	
1	ClaraTom
2	AmandaMatt
3	RalfKaty

Rates	
1	58
2	46
3	100