CS4H
Introduction to CS with Scratch

Youth Guide

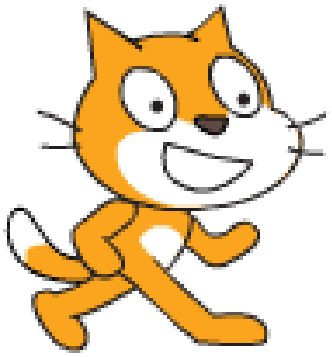
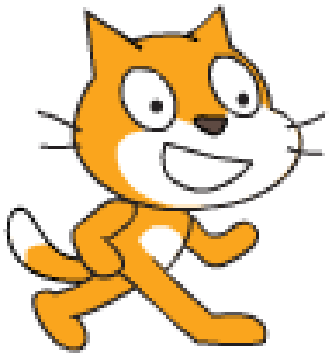


Table of Contents



About this Guide

This guide is meant as a quick introduction to the basic elements of programming within the Scratch environment. It covers the most fundamental principles of programming in any programming language: sequence, iteration, conditionals, variables, and modularization. After going through this guide you should be able to write simple programs for a variety of purposes. Learning to program is like learning to play a musical instrument: only with lots of practice can you improve your skills and create beautiful things. This guide should give you some of the fundamentals on which to build, but you will need to spend lots of time on your own practicing, experimenting, exploring, and creating. Luckily, doing that is easy and fun with Scratch. So, let's get started!

This guide was created as part of the "4-H Computing Connections" grant funded as part of the University of Illinois Extension and Outreach Initiative. (See, http://web.extension.illinois.edu/initiative/)

Principal Investigator: Lenny Pitt

Co-Principal Investigator: Maya Israel

Co-Principal Investigator: George Reese

Key Extension/Outreach Contact: Robert Smith (4-H Statewide Robotics Educator)

Primary authors are Lenny Pitt, Judy Rocke, and Jana Sebestik.

Input on all aspects was provided by the CS4H team:

- Travis Faust
- Maya Israel
- Lenny Pitt
- George Reese
- Judy Rocke
- Saadeddine Shehab
- Robert Smith

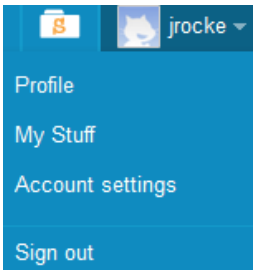
Creating with Scratch

Scratch is a visual programming language that lets you create and share projects with others. It encourages you to problem solve, think logically, test and evaluate outcomes, and collaborate with others. To try out Scratch, see examples of projects, and join Scratch click here: https://scratch.mit.edu/

Scratch works best if your browser is Firefox or Chrome.

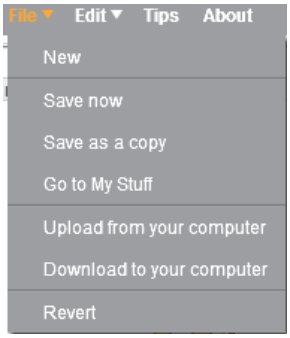
Create your own account on the Scratch website. Click 

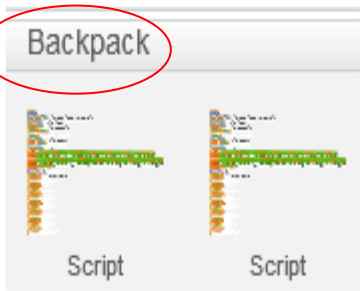
and follow the directions. Then click Try it Out to get started.



There are advantages to having your own account. You can create, save, and share your activities on the Scratch website, and you can view other people's projects. To return to your saved projects, login using your password. Click the drop down menu under your login name and select My Stuff.

If you click File while you are working online there are more options. You can Download or Upload a project to or from your computer. You can also use Revert to delete all changes you made to an activity. This resets the activity to the way it was when you opened it.



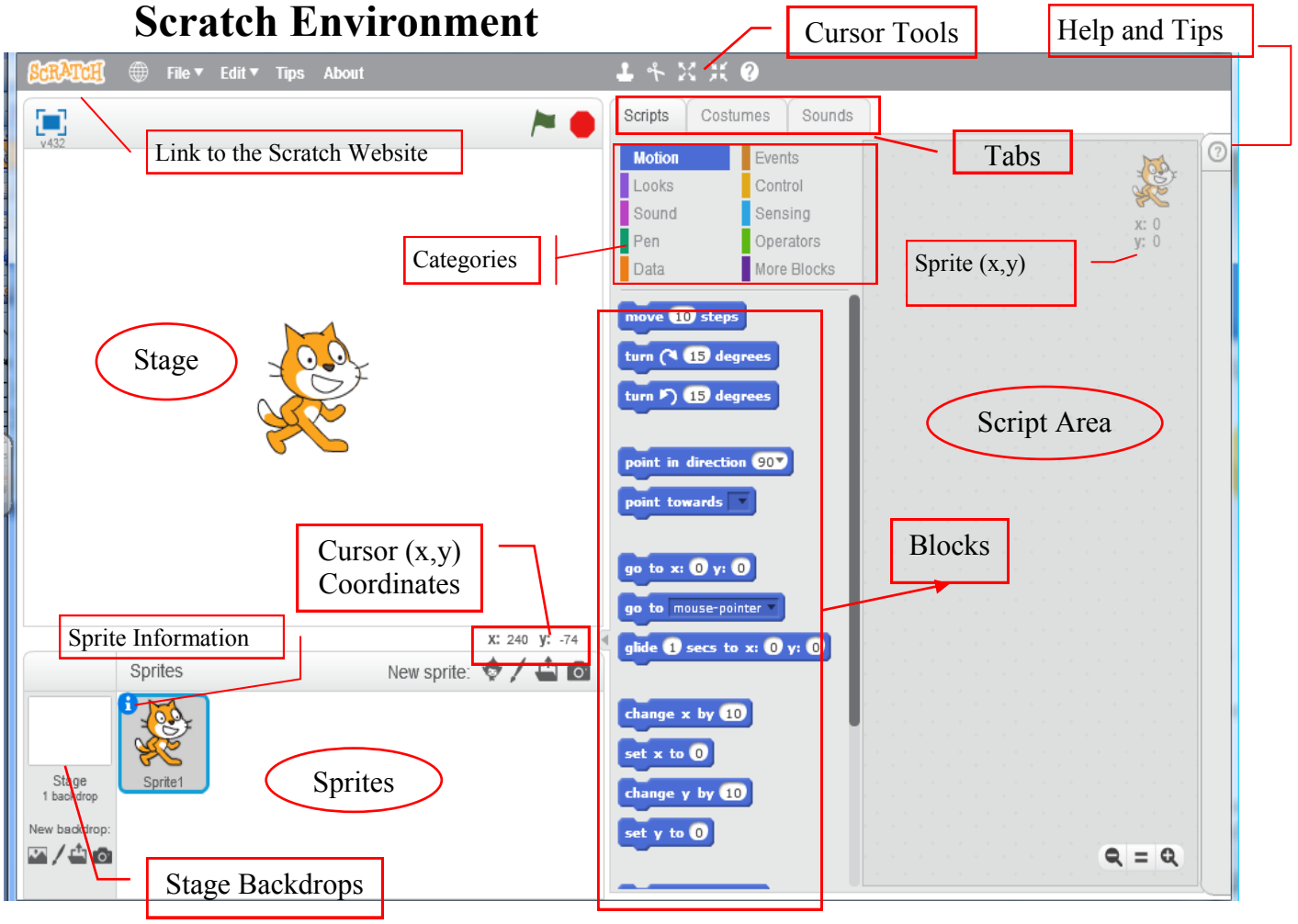


Another advantage to working online is the Backpack at the bottom of the Script Area. You can save scripts in your backpack from one project and then pull them out to use in another project. Go to the ScratchEd YouTube Channel to learn more.

https://www.youtube.com/watch?v=nQ8UrAhht4I

It is possible to download a version of Scratch onto your computer so you can use Scratch without an internet connection. See the instructions at the end of this guide.

Scratch Environment



Stage: The area where the Sprites appear

Sprites: Objects that move about the stage

Script Area: Area used to place blocks for Scripts

Tabs: Switches between Scripts, Costumes, and Sound tabs

Categories: Click each to show blocks in that category

Sprite (x,y) Coordinates: Shows the (x,y) coordinates of the selected Sprite

Cursor (x,y) Coordinates: Shows the (x,y) coordinates for the cursor

Blocks: An item or tile used to command your Sprite to perform an action

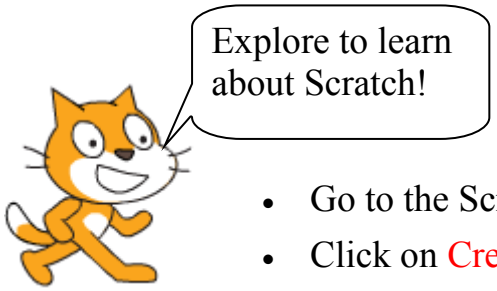
Stage Backdrops: Used to change the background of the stage

Help and Tips: Shows helpful tips, frequently asked questions, and activity suggestions

Cursor Tools: Used to make a stamp, cut, grow or shrink a Sprite

Sprite Information: Shows sprite's (x,y) location, rotation style, visibility, and whether it can be dragged in presentation mode

Link to Scratch Website: Use to learn more about Scratch, create stories, games, and animations - Share with others around the world



Getting Started with Scratch

The blocks in each category are color coded to match the name of the category. This makes it easy to find a block.

- Go to the Scratch website at https://scratch.mit.edu/
- Click on Create in the menu bar to open a new Scratch project.
- Your new project page should look something like the previous page, where we've identified the different elements to help you understand all that is going on. The Scratch environment has lots to explore.
- Spend time clicking and dragging blocks around.
- Snap blocks together to create a script.
- Share what you discover.

The Stage

The Scratch cat is the sprite that appears on the stage when you open a new activity. Sprites are the objects that follow instructions to move around the stage. Find out how the stage is organized.

1. Drag this block from the MOTION category into the Script Area.
2. Put 0 into the x box and 0 in the y box. Click the block. Where is the cat now?
3. Input ordered pairs into the x and y boxes. Try (150, 150) and then click on the block. Try (-150, -150). What does each ordered pair of numbers tell the cat to do?
4. Predict where the cat will go with (-150, 150) and (150, -150). Test it. Try other ordered pairs. Predict where the cat will go before you test each ordered pair.
5. How is the stage organized?
6. Get rid of the **go to x: y:** block by dropping it back to the blocks area from which it came, or by right-clicking on it and then selecting delete from the drop-down menu.

Sprites

Make sprites speak, move, change color, size, and direction by using blocks from the MOTION and LOOKS categories.

1. Pull out these blocks and drop them into the Script Area.
2. Click on the **move 10 steps** block. What do you notice? The cat should have moved a small amount. A step in Scratch is a single pixel on the screen, so 10 steps is not very far. Change the value 10 by clicking on it and replacing it with 100 and then click on the block again. In Scratch, wherever you see a white area inside of a block you can type in your own values to change the behavior.
3. Snap the blocks together. Click on the top block . What does the cat do?

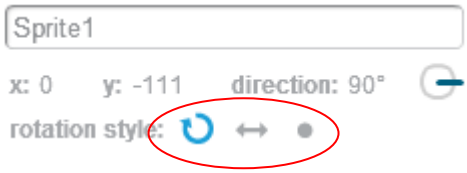
3. Separate the blocks in a script by clicking a block and pulling it away. Put the blocks in different orders or sequences. Run the script again after each change. How does the sequence make a difference?

You can duplicate an entire script. Right click on the top block of the script. Then select duplicate

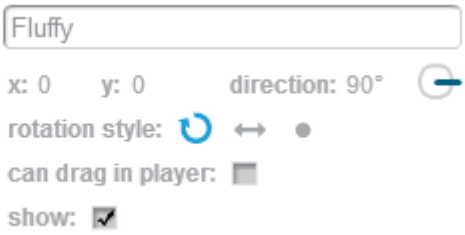
Delete a script you no longer need by dragging the script into the block area and dropping it there. Undelete a script by clicking Edit in the menu above the stage. Then select undelete.

4. Change the input numbers of the blocks, and use the drop down menu in the **point in direction** block. Select (- 90). Run the script again. Why is the cat upside down?

5. A sprite's rotational style is controlled in the sprite's information box. Click the ***i*** on the sprite icon. The sprite's information box will open. This box gives the sprite's name, its (x,y) location, the direction it is pointing, and its three rotation styles. The first style is full rotation, the second is back and forth, and the last is no rotation. Try each and run the script again. Think about when it makes sense to use each.



6. This is also a good time to change the sprite's name to something other than "Sprite1". Your projects may have multiple sprites, and it will be less confusing if you name them something that explains what they are. So, click on the ***i*** and enter "cat" or "fluffy" or some other name for the cat besides "Sprite1".

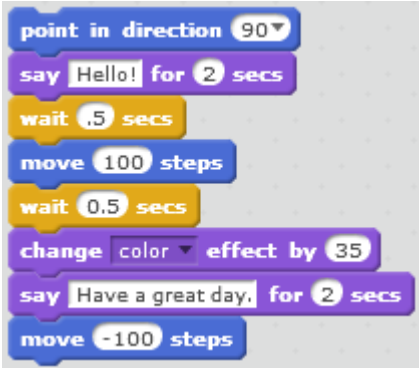


7. Sometimes the cat moves too quickly to see. Pull out these blocks and add them to your script to make the cat follow the list of instructions below. In what sequence do you place the blocks? What numbers do you input into the **glide** block to make the cat glide, and then return to its starting location?

- glide across the stage
- wait one second
- change color
- wait a second
- change size
- create a text bubble
- glide back across the stage

Starting and Stopping Scripts

A script is a series of connected blocks that give instructions to a Sprite.



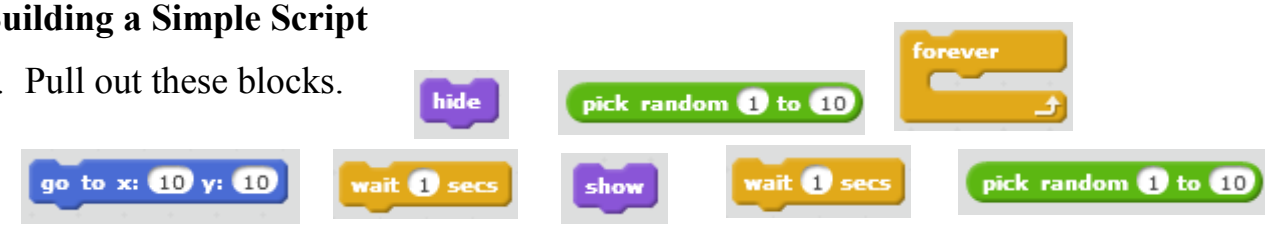
1. Create this script.
2. Click any block to start the script.
3. There are other ways to start a script. Pull out this block. Add it to the script. When you click the green flag above the stage, all scripts with the **green flag block** will start at the same time.

4. Pull out this block. It has a drop down menu. Explore other ways to start a script using the drop down menu. If you want more than one script to run at the same time, use the same **event block** to start the scripts.

5. To stop all scripts that are running, click the stop sign on the top of the stage.

Building a Simple Script

1. Pull out these blocks.



2. The **random 1 to 10** blocks will drop into the **go to x:, y:** block. Try this.

3. The **forever** block will expand to let you put other blocks inside. The blocks inside the **forever** block repeat forever in a loop.

It is a good idea to test a script as you create it. As you add blocks to a script, run the partial script to see if it is doing what you expect it to do. It is easier to find problems with the script if you test it as you add blocks.

4. Change the sequence of the blocks inside the **forever** block. The new sequence should make the cat do the following repeatedly:
    * appear for a second
    * disappear for a second,
    * move to a random (x, y) location on the stage
    * then the loop should start over

5. Change the input numbers on the **random** block so that the cat will be able to appear anywhere on the stage. What numbers should you use? Test the script.

6. Change the number in the **wait** block. Try using a decimal.

7. Does the script make the cat appear, disappear, and reappear at a random location?

New Sprites

Get new sprites by clicking New sprite below the Stage. There are four different ways to get new sprites.

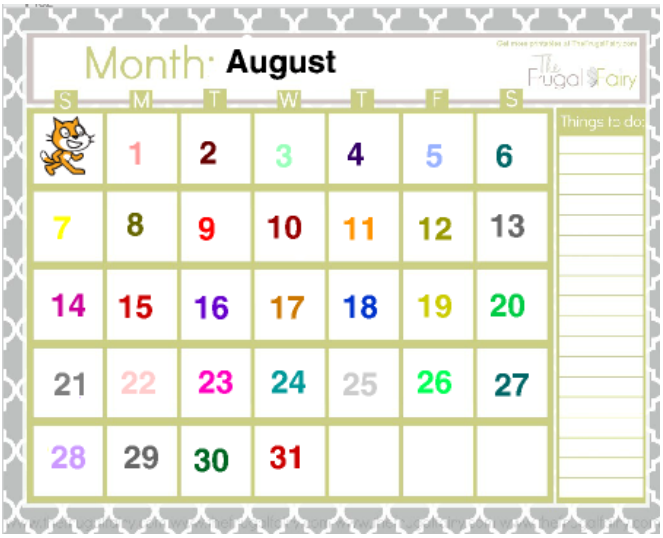
1. Explore each way to get new sprites.

2. To delete a sprite, right click the sprite and select delete.

3. Select a new sprite. Write a script to make it move, talk, or change color.

Sequence

Computers will do exactly what you tell them to do. They will follow instructions one step at a time, in exactly the order you give them. It is important to understand this, because oftentimes when the program is not behaving the way you think it should, it is because it is doing exactly what you said, not what you meant. To "debug" a program it is useful to think like the computer, and carefully examine the instructions one step at a time. In this section we'll learn some basic commands, and see how simply assembling them in the right order can result in interesting and useful behavior.

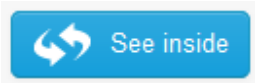


Calendar Activity

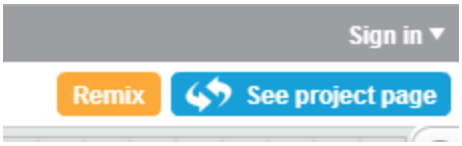
Open a new tab and then open the calendar activity file at:

https://scratch.mit.edu/projects/52455886/

Select See Inside to see the Scratch editor for this activity.



The Calendar Activity is a Scratch file created by CS4H. When you add your own work to a file, you are creating a Remix. Click the Remix button to begin.



The cat's starting position for this activity is in the first block before August 1. You can create a **reset script** which makes the cat return to this position whenever needed.

1. To do this, pull out this block.

2. Find the x, y coordinates for the cat while it is at this starting location by looking at the Sprite's coordinates in the upper left corner of the Script Area. Type the cat's coordinates into the block.

Hint: Check the cat's rotational style to be sure it is set to full rotation.

3. To start this reset script, use this block.

4. Drag the cat to a different location. Press the space bar. Does the cat return to its starting location?

5. The cat moves so quickly that you don't see him move. Use this block to make it wait at any location. Run your script.

6. Drag your reset script to a corner of the Script Area.

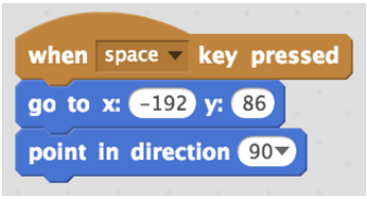
Begin a new script.

7. Use the **move** block to make a new, separate script that makes the cat move to August 1. Then use the reset script to return the cat to the starting position.

8. How many steps would you tell the cat to move for it to land on the middle of the date August 5? Test it.

9. Right now the cat is pointing right. Pull out this block. Explore its drop down menu. What does this block do? Make the cat point down, left, right, and up.

10. Add a **point in direction (90)** block to the reset script so that the cat will always start in the first box, pointing right. Now your reset script looks like this.



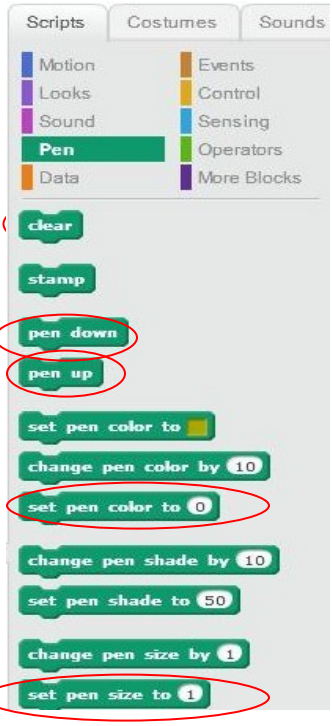
11. Now create a new script. Use only the move and point blocks to make the cat move to August 9. Start the script with this block.

12. Change the order of the blocks in your script. Does the script still work? Why is order important when you are writing scripts?

13. Use only the **move, point,** and **wait** blocks to make the cat land on your birthday date when the green flag is clicked. (For example, if your birthday is May 17, make the cat go to August 17.) Is the order important? How will you start this script?

14. Use only the **move, point,** and **wait** blocks to make the cat first go to your birth date, wait there for two seconds, and then go to your partner's birth date.

You can start or stop drawing the path the cat is taking by using blocks in the PEN category. Change the size and color of the pen by pulling out the correct blocks. Clear away old paths that the cat has made by using the **clear** block.



15. Make the cat draw the path it takes moving from August 6 to August 16.

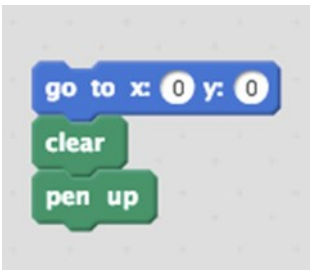
16. Reset and try this block. Try changing the number in the block. What happens to the cat?

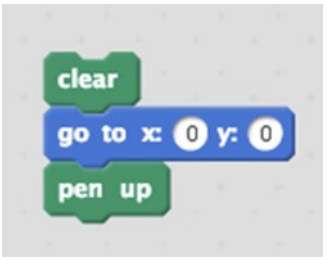
17. Add the **pen up** and **clear** blocks to the reset script to clear old paths when you return the cat to its starting location.

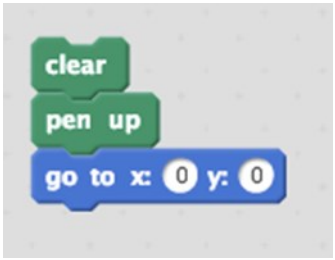
18. Reset by hitting the space bar to run the reset script. Now create this script. Before you run it, guess on which date the cat will stop. Test this. Were you correct?

19. Reset and then change the sequence of the blocks. Does the cat land in the same spot? Why or why not?

20. Create each of the reset scripts below. Try each by first moving the cat forward some number of steps so that it first draws a line, and then click each of these reset scripts. Which of these reset scripts make the cat do what you want it to do? Why? Think about what the correct order is for these commands. There is more than one way to do this.

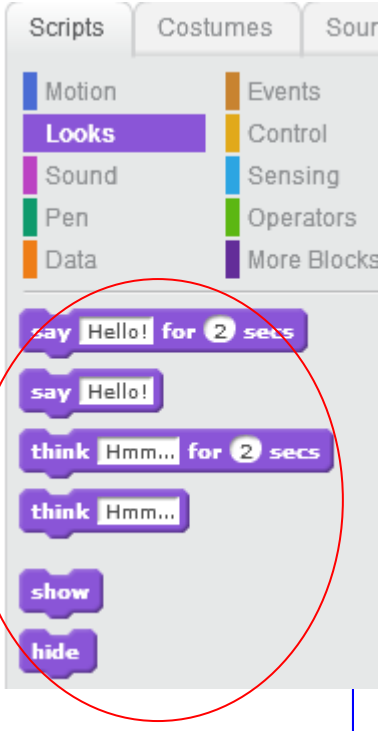






Challenge:

- Use only the circled blocks and the **wait** block to write a script that makes the cat:
  - Say, "This will be fun!" for two seconds.
  - Disappear for one second
  - Reappear.
  - Think, "Did you miss me?"
  - Is the sequence important in this script?
- Create a challenge of your own for the cat. Create the script that would make the cat meet that challenge. Test the script to see if it meets that challenge.



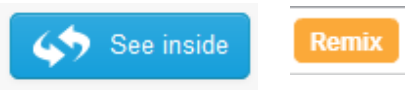
Talk About

- What were some of the things you learned in these activities?
- Why is a reset script useful when using Scratch?
- What ideas would you share with someone who has never used Scratch before?
- How would you explain sequence?

Iteration

Computers will not only do exactly what you tell them, they will also do the same task over and over and over. They never get tired, and they never make a "mistake." Asking a computer to do something over and over and over, is called iteration.

Box It In Open a new tab and new Calendar Activity.
https://scratch.mit.edu/projects/52455886/
Click See Inside and Remix.



1. Create a reset script that will make the cat start on August 17, pointing right. Add the **clear** block to the reset script. Use the space bar to start it. Put the reset script in the corner of the Script Area.

2. Then create this new script on the right. Copy it three times and connect all four copies.

3. Your new script should look like the script on the left.

4. Add a **pen down** block to the top of the script.

5. Predict what will happen before you run the script. Run the script. What happens when you Reset the script?

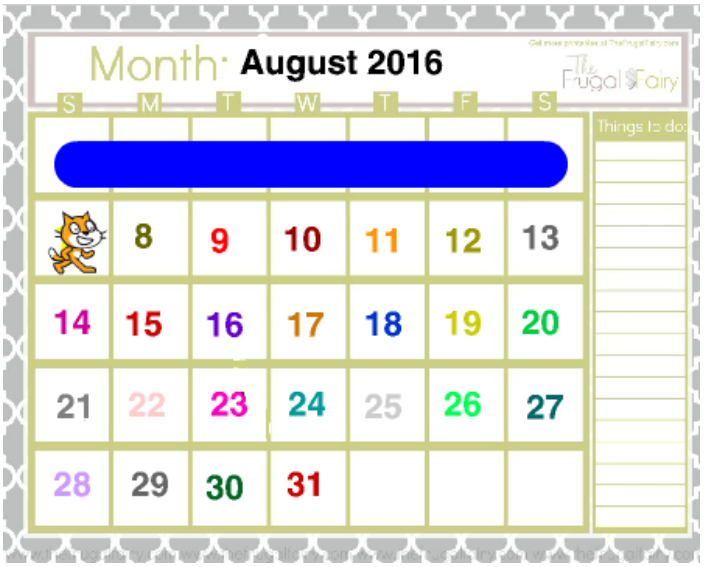
6. Notice that you used the same three blocks over and over. Instead, you can use a **repeat** block from the CONTROL category to repeat a task as many times as you need. This is iteration.

7. Go to the CONTROL category and pull out the **repeat** block. Fit the three blocks **move, turn,** and **wait** inside the **repeat** block. Decide how many times you want these three blocks to repeat. Type that number into the **repeat** block. Add a **pen down** block above the **repeat** block. Now reset, and run the script. What happens"?

Challenge:

- Reset and then make the cat draw a square around your birth date.
- Reset and then make the cat first draw a square around your birth date, and then move and draw a square around your partner's birth date.
- Make the cat draw a colorful star around your birth date and another around your partner's birth date. Use this script.

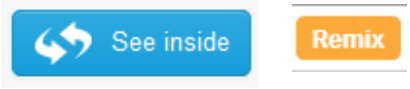
Calendar Cross Out



Look at the calendar page on the left. The cat has crossed out the first row using a thick pen, and the cat is now located at the start of the second row.

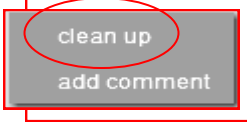
Open a new Calendar Activity.
https://scratch.mit.edu/projects/52455886/

Click See Inside and Remix.



1. Create a reset button to make the cat return to the first block pointing right. Include the **clear** and **pen up** blocks in your reset script. Be sure to use the correct order.

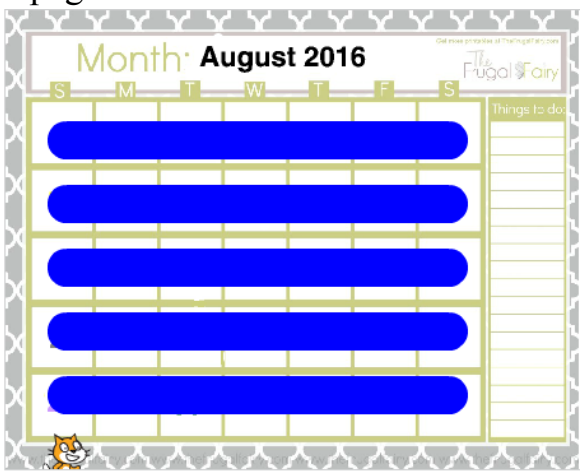
Often the Script Area is full of scripts and unorganized. Clean up and organize this area. Right click in the gray color of the Script Area. A box will appear. Select clean up.



2. Use the blocks below to make the line shown above on your calendar and stop the cat at August 7. You may have to use some blocks more than once.

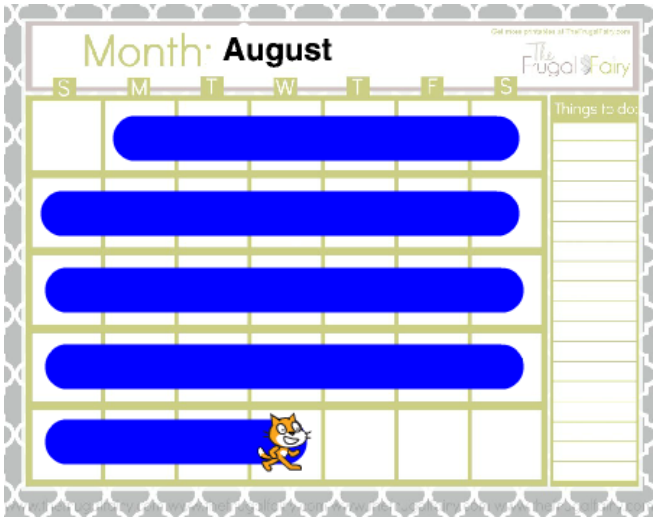
3. Think about the sequence for the script. Does it matter? Test the script.

4. Use the script you wrote above for the blue line and the **repeat** block to cross out every row on your calendar. Your calendar page will look like the calendar on the right. Test your script.



*Make the cat move left without changing the way it is pointing by using the **move** block and inserting a **negative number instead of a positive number**.*

Bugs, Glitches, and Problems in Scripts



Open a new Calendar Activity.
https://scratch.mit.edu/projects/52455886/

The script on this page was written to make the cat create the calendar on the right.

The cat has drawn lines through only the dates. It has left the empty squares blank.

However, when you run the script on this page, it does not work correctly. It does not recreate a calendar page where only the dates are blocked. This script has errors in it. It has bugs in it. That means that the script is not working correctly. There are 3 things wrong with this script.

1. First create a reset button for the cat.
2. Then recreate this script as it is written.
3. Can you find what is wrong with this script and debug or fix it to make it recreate the calendar shown on this page?

Copy this script after you create it. Right click on the script. Select duplicate. If you do that before you begin making changes, you can return to the original script after you see if the changes you made were correct.

4. Test your new script to see if you found all the bugs. Pull the blocks apart to test the script in small sections.

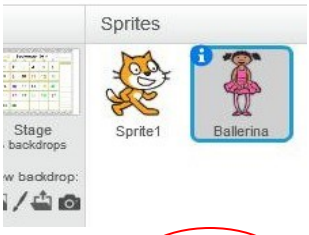
5. Does your new script create the calendar shown above? What did you change?

Ballerina

Open a new Calendar Activity.
https://scratch.mit.edu/projects/52455886/

Notice the ballerina sprite shown below the calendar next to the cat sprite. This sprite has four different costumes. Costumes control the way a sprite looks.

1. Click the ballerina. Go to the Costumes tab.

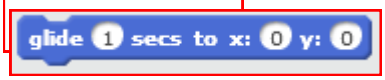


2. Click each costume. As you click each costume, the ballerina changes costumes, or positions. Notice each costume is labeled as ballerina-a, ballerina-b, ballerina-c, or ballerina-d.

3. Now quickly click each costume. As you do this, the ballerina appears to change her position. The quicker you click, the faster she seems to move from one position to the next, making it look like she is dancing. This is like creating a movie by rapidly changing several different still shots to make it appear like motion.

4. Now look at the calendar. The ballerina is hidden. You cannot see her. Make her appear. First click the ballerina sprite, then go to her scripts. Notice the ballerina has one script already written. It uses these blocks. This script kept the ballerina hidden while you worked with the cat. Now you need to make her appear on the calendar. Pull out the **show** block. Click it. Where does the ballerina appear? This is the ballerina's starting position. Create a reset script for the ballerina to show at this location.

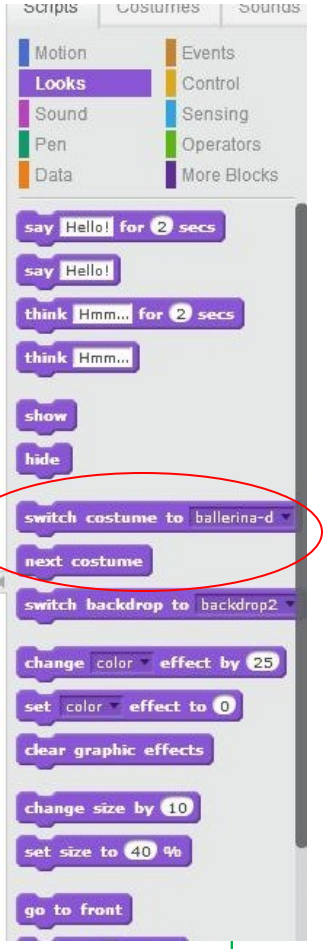
*Remember, to make a sprite move left input negative numbers into the **move** block. Or make the ballerina glide to any location using this block.*



5. Move her around the calendar. Use the reset script to make her return to her starting position and show.

6. Write a script that will make the ballerina show, wait for a second, and then move to August 30. Test it.

7. Make the ballerina appear to dance by creating a script that will make her switch costumes rapidly.



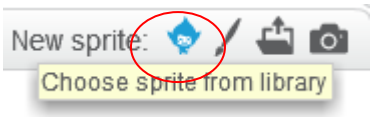
- To do this, pull out the **switch costume to** block. Notice there is a drop down menu which allows you to select which costume you want to use.
- Create a script that makes the ballerina appear in her first costume for a second, then appear in her second costume for a second, then appear in her third costume for a second, and finally, appear in her fourth costume for a second.
- Test this script.
- Does it seem like it takes too long to wait between costume changes? How can you make this wait time shorter?
- Make the ballerina appear to dance by running this script over and over and over, or tell the computer to repeat this costume change over and over by pulling out the **repeat** block. Repeating things over and over is called **iteration.** Place the **repeat** block around your script.

- The same effect can be obtained by using the single block **next costume** inside of the repeat block. This cycles through all of the costumes of a sprite.
- Make a copy of this script to use in the Challenges below. Right click the top of the script. Select duplicate.

Challenge:

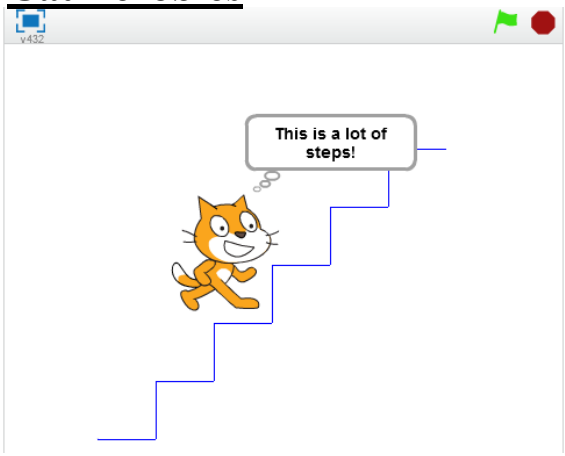
- Make a script for the ballerina to appear in her starting position, move to August 30, and then dance forever. Which **repeat** block did you use?
- Make a script for the ballerina to appear in her starting position, move to your birth date, and dance several times.
- Make a script for the ballerina to appear in her starting position, move to your birth date, dance several times, and then move to your partner's birth date and dance forever. Make her stop and wait a second each time she makes a turn on her way from your birth date to your partner's birth date.
- Click Choose sprite from library located under the stage. Double click the sprite called 1080 Hip-Hop. This sprite has many costumes. Write a script to make him move around the stage, dance, say things, change size, and change colors. Test your script. Does he dance, look, and move the way you want him to dance, look, and move?


1080 Hip-Hop



Cat Aerobics

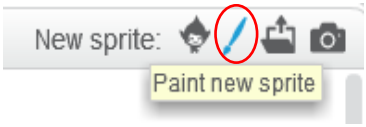
In this activity you will make the cat climb stairs.

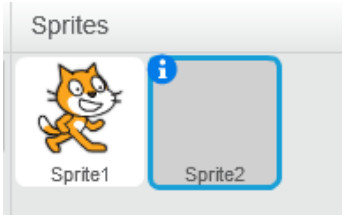


Open a new Scratch file.

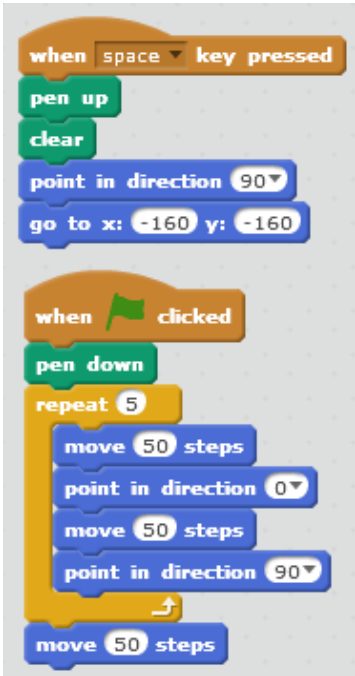
1. First, make the stairs. You will draw the stairs with a new sprite that is so tiny you cannot see it!

2. To create a tiny sprite click the paintbrush in the New sprite row. The paint editor will open under the Costume tab.





3. Click once on the line tool and click once on the paint field. A new sprite called Sprite2 will appear next to the cat sprite. You cannot see the sprite, but it is there. It is too tiny to see.



4. Now click the Script tab.

5. Here are the scripts you will write for your new, tiny Sprite2. Before you create these scripts, make a prediction about what will happen when you run them.

6. Why is the **move 50 steps** block outside of the repeat loop at the bottom of the script? What if it were put it inside the repeat? Explain why this does not produce what you wanted? What would you do if you wanted the steps to begin with a vertical line at the very bottom, and not the horizontal line?

Shrink or grow the size of any sprite by clicking the arrow keys above the Tabs. Then click the sprite whose size you want to change.

Challenge:

- Make a longer staircase with more stairs for the cat to climb.
- Make the cat wait to walk up the stairs until it is clicked with the cursor. Look in the EVENTS category for this block.
- Make the cat walk partway up the stairs, and think, "This is a lot of steps!" before he continues to the top. When he gets to the top, make him say, "I made it!"

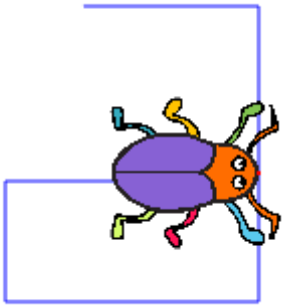
Talk About

- How has interation helped you in this activity, the ballerina activity, and the calendar activity?

Conditionals

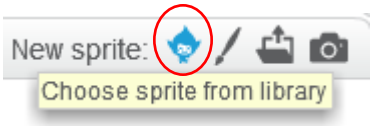
A conditional is a command that says, "If an event occurs, then another event must occur."

The Beetle

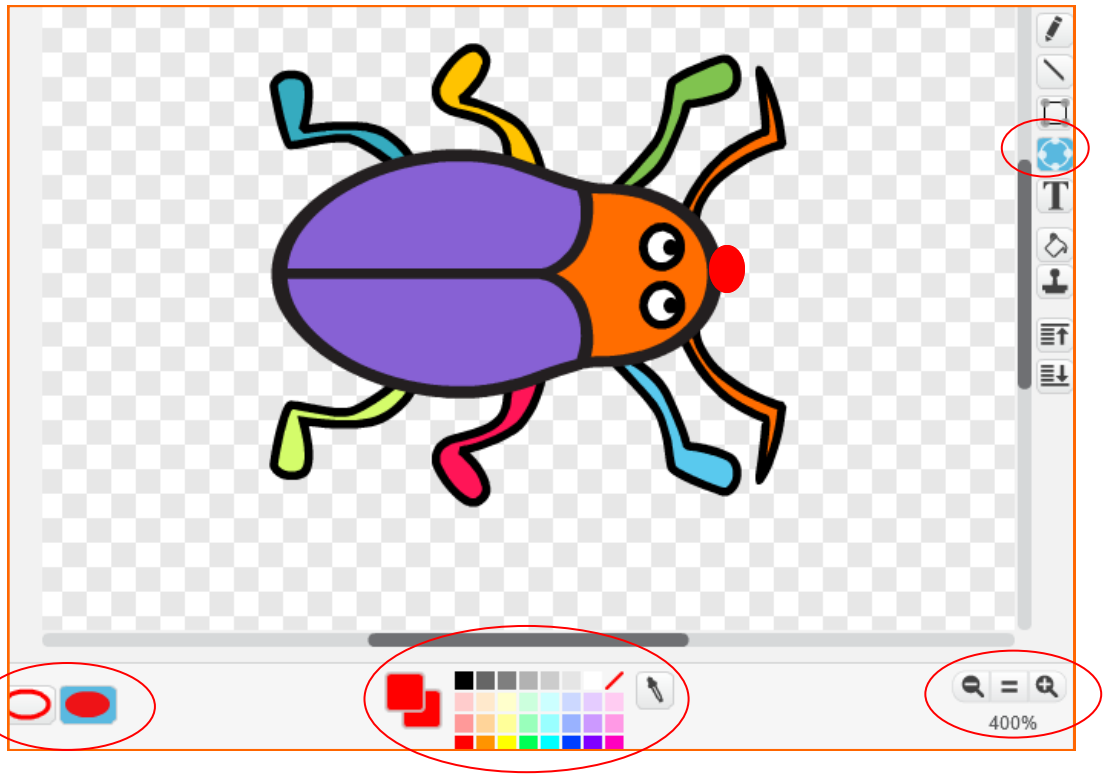


In this game the beetle draws a path as it moves across the stage. The beetles movement is controlled by the arrow keys on the keyboard. The beetle will move as long as it does not cross or touch a path that it has already drawn on the stage. If the beetle touches the path, the game ends.

1. Open a new Scratch file. Delete the cat sprite.
2. Get the beetle sprite. Click Choose sprite from library under the stage. Find the beetle. Double click the beetle.



3. The beetle will appear on the stage, and the sprite for the beetle will also appear selected below the stage. Click the Costume tab. This allows you to change the costume, or look, of the beetle. Create a red nose for this beetle. To make it easier to see, enlarge the beetle using the magnification button in the lower right corner. Select the ellipse tool from the drawing tools. Choose the filled ellipse in the lower left corner and select the red color. Draw in the red nose on the beetle. Be sure the outside and inside of the nose is red. When you have created the nose, click the beetle on the stage. The beetle should now appear with a large red nose.



4. You can control how the beetle moves with the arrows on the keyboard and four scripts. You can make the beetle point right with the right arrow. Make the beetle point left with the left arrow. Make the beetle point down with the down arrow. Make the beetle point up with the up arrow.

   To do this, use the EVENT block **when __ key pressed**. Notice this block has a drop down menu. In this menu, there is an up arrow choice. Combine this block with the **point in direction** block to make the beetle change directions when the up arrow is pressed.

   Why does the **point in direction** block use a 0 for this script?

5. Make a copy of this script. Change the settings in this copy to make the beetle point down when the down arrow is clicked. How did you change the **point in direction** block to make the beetle point down?

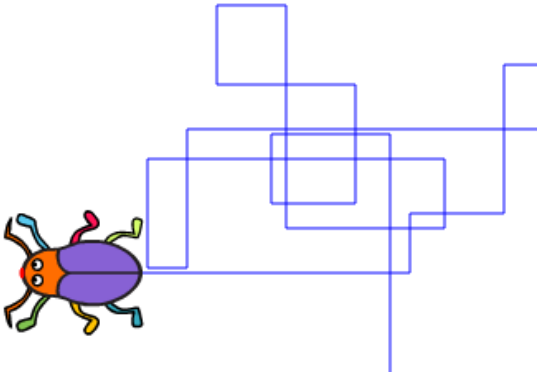
6. Make two more copies of the script for the beetle. One of these scripts should make it point left, and the other should make it point right when the left or right arrows are used on the keyboard. Test these. Does your beetle point in the correct direction when each arrow is clicked?

Remember, clean up and organize the Script Area by right clicking in the gray color of the Script Area. A box will

7. Use these blocks to make the beetle draw a path forever as it moves around the stage when the arrow keys are clicked.

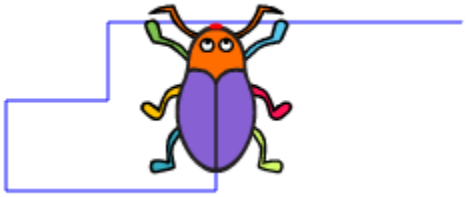
8. Set the pen size to 4 by pulling out the **set pen size to** block. Enter 4, and click the block. This controls the thickness of the line that the beetle will draw as it moves.

9. Slow the beetle down by changing the **move** block to 2 steps instead of 10. Test your script. Does the sequence of these blocks make a difference? Is the beetle's direction controlled by the arrow keys? Which block should you use to clear the path when you no longer need it?



10. Do you need to add a **wait** block?

Now you need a way to make the beetle stop moving if it crosses or touches the path it has drawn on the stage. This is a **conditional**: If the beetle touches the line of the path, then it stops.

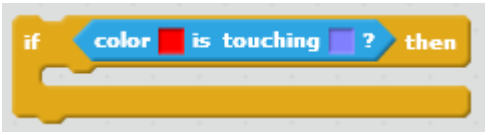


11. To create this conditional, pull out this block.

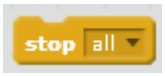
12. You want the beetle to stop, if its red nose is touching the blue path that it has drawn on the stage. Get this block.

13. Click the first color box in this block. As you drag the mouse across the beetle on the stage, notice this box changes colors. Click the red nose of the beetle, and this box will change to red. Now change the second box to the color of the pen path. Drop the **color** block into the space of the **if ___ then** block.

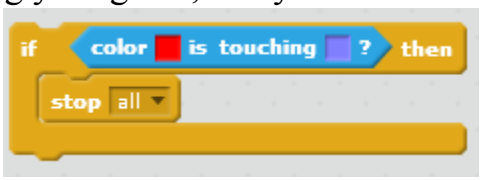
If the beetle's nose is too small to hover over, you can temporarily grow the beetle and then select the color.



14. This completes the first part of the conditional: If the beetle touches the path then. The second part of the conditional is: Stop. Pull out the **stop all** block. Where would you put this block to make the beetle stop if it touches the blue path?



15. Remember you want to forever test this conditional during your game, and you want the beetle to forever move. Where would you put this new block in the script you have created for the beetle?



16. Test your script. Does it work? If the beetle's red nose is too small, or if the pen size is too small, or if the beetle moves too many steps at once, it is possible that the beetle's nose will not contact the pen trail, and the beetle will just move right over it. Getting things to work may require a bit of adjusting of these values. If yours does not work, try making the step size smaller, or the nose or pen size larger.

Challenge:

- Change your script to make the path the beetle draws a new color. Think about all the changes you will make to your script. Make the changes and test your new script.
- Change the arrow keys so that the right arrow makes the beetle turn 5 degrees clockwise, and the left arrow makes it turn 5 degrees counterclockwise. Use the **turn** blocks found in the MOTION category.

Birthday Cake Game

Open a new Calendar Activity.

In this game, the cat is trying to get to its birthday cake. The knight is guarding the cake.

The conditions are: If the knight touches the cat, the game ends and the cat loses. Or, if the cat makes it to the cake without being touched by the knight, the game ends, and the cat wins.

1. Get a birthday cake sprite. Find the birthday cake in the Holidays category of the sprite library.
2. The cake is too large to fit on the calendar. Shrink the cake to fit on the calendar.

3. Get a knight sprite. Shrink it to fit. The knight will guard the birthday cake. Write a reset script to make the knight always start on August 17, pointing right, when the space bar is pressed.
4. **Click the cat sprite, which selects it, and makes it active.** Make a reset script for the cat. The cat should start in the first box, pointing right when the space bar is pressed. Use both of these scripts to reset the game.

5. Change the name of the cat from "Sprite1" to "cat" by clicking the *i* next to the cat. Replace "Sprite1" with "cat."

6. **Click the knight which makes it active.** Write a script to make the knight travel up and down along Wednesday's column on the calendar. The knight must travel up and down forever. Which blocks would you use to make the knight travel forever up and down along Wednesday's column? Use the green flag to start this script.

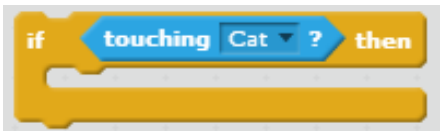
Make the knight bounce off the edge of the calendar and continue moving by using this block from the MOTION category.

6. Test your script to see if the knight moves up and down forever along the column for Wednesday.

7. **Click the cat sprite which makes it active.** In this game the cat is controlled by the arrow keys on the keyboard. Make these scripts in the same way that you made scripts to control the direction of the beetle with the arrows on the keyboard in the last activity.

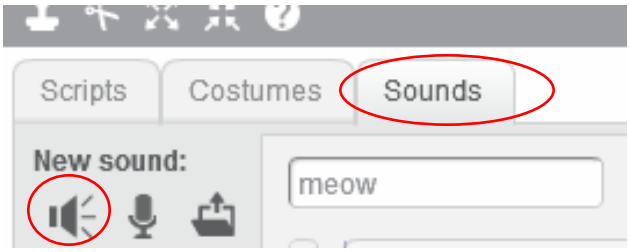
8. The game stops if the knight touches the cat. In this case, the cat will lose the game. The conditional is, "If the knight touches the cat, then the game ends." Signal the end of the game with a sound and by stopping the game. **Click the knight which makes it active.** Pull out the **if___then** block.

9. Get this **touching** block from the SENSING category. This block tells the knight to do something when it is touching something. Pull down the menu and find "Cat." Drop this block into the space of the **if__then** block.



10. Now you have the first part of the conditional finished. For the second part of the conditional, play a sound to indicate the cat lost when the knight touches it.

11. Go to the Sound tab. Under New Sound click the horn icon. Find the *laugh-male2* sound and double click it.



Remember, clean up and organize the Script Area any time by right clicking in the gray color of the Script Area. A box will appear. Select clean up.

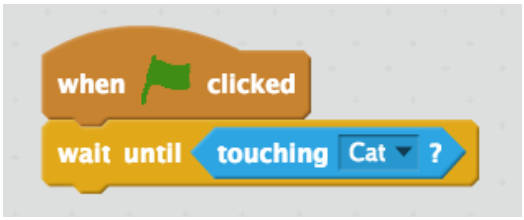
12. Go back to the knight script. Open the Sound tab. Pull out these blocks and create this script. Why would you place these two blocks inside the **if ___ then** block?

*If you need to make more room inside the **forever** block, it will expand.*

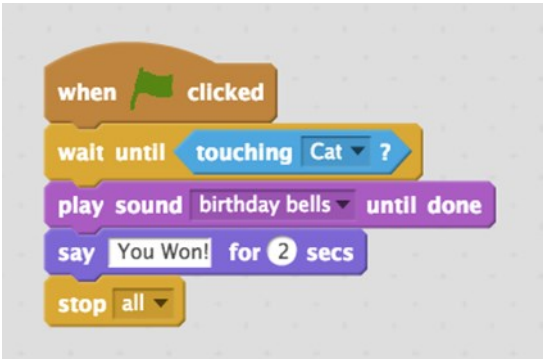
13. You want the knight to forever go up and down along the Wednesday column, and you forever want the knight to stop the game and play a sound if the knight touches the cat. Where would you put the **if__ then** block you just created in the script for the knight to make that happen? Test this. Does it work?

14. We'd like the game to end when the cat reaches the birthday cake. In order to do this, we'll program the cake to wait until the cat touches it, then play a birthday song and end the game. Click on the cake sprite which makes it active. Pull out the **wait until ___** block. Now look in the **SENSING** category. Which blue block would tell the cake to wait until it touches the cat? Drop it into the wait until block.

15. We need a way to run the script when the game starts. Pull out **when green flag clicked** from the CONTROL category, and attach it to the top of the **wait until** block that you just constructed. You should now have this:



16. Now go to the Sound tab and click the horn in New Sound. Pull out the birthday bells sound. Go to the Script tab. Pull out and combine the blocks to create the following:



17. Now test your script and make sure it works correctly.

Another way to end the Birthday Cake Game

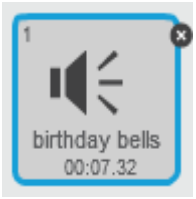
When the cat makes it past the knight and touches the cake, the cat wins. The cake broadcasts a message to the knight to let the knight know he lost the game. The cake also shows the cat won by saying, "The cat won!" and playing the sound, *birthday bells*.

- To write the cake's script, **click the cake which makes it active.** Pull out this wait until block and touching block. Select "cat" from the drop down menu. Drop the touching block into the wait until block. Now the cake will wait until it is touching the cake to do something.

- Look in the EVENTS category for a way to broadcast a message to so the knight knows it lost. Pull out this broadcast block, select "new message" from the drop down menu, and type "cat touching cake" to name the broadcast.

- The cake also indicates the cat won by saying, "The cat won!" Which block from the LOOKS category do you need? Pull out that block.

- Finally, the cake will play the sound, *birthday bells* when the cat touches it. Go to the Sound tab. Under New Sound click the horn icon. Find the *birthday bells* sound and double click it.



- 18. Combine all these blocks to make the cake:
  - Wait until the cat touches it
  - Broadcast "cat touching cake"
  - Play *birthday bells* until done
  - Say, "The cat won!" for 2 seconds
  - Use the green flag to start this script

When the knight receives the broadcast, "cat touching cake", the knight will stop traveling up and down, look at the cake, and say, "How did that cat get around me?"

- To write this new script for the knight, click the knight which makes it active**.** Start this new script with this block from the EVENTS category. With this block, the knight will wait to run this new script until it receives the "cat touching cake" broadcast.

- Pull out the stop all block and select "other scripts in sprite" from the drop down menu. This will stop the other script for the knight which makes him stop traveling up and down.

- Which three blocks will make the knight point toward the cake, say, "How did that cat get around me?", and stop all scripts for the game? Pull out these three blocks from the LOOKS, MOITION, and CONTROL categories. Add these to this new script for the knight.

- Test your game. Does it work correctly? If not, can you find the problem with your scripts? Do you need to add a wait block anywhere?

Challenge:

- What blocks would you add to your script, and where would you add them, to make the knight wait and say, "That cake looks great. I am going to guard that cake," before it begins to travel up and down. Test this.

- What other changes could you make to the Birthday Cake Game? Change the script to make the game follow your changes. Test your new script. Does it work?

Talk About

- What is a sequence?
- What is iteration?
- What are conditionals?
- Using sequence, iteration, and conditionals, what are some activities you would like to create in Scratch?

Getting User Input with Answer Boxes and Conditionals:

Scratch allows you to ask the people using your activity a question. Then, based on the answer, Scratch allows you to tell the computer to perform a task.

Can you guess on which date President Obama was born?

1. In this game, the person playing the game, or user, tries to guess the month and date President Obama was born. In order to create this game, you need to know the answer. President Obama was born on August 4, 1961.

Open a new Calendar Activity: https://scratch.mit.edu/projects/52455886/

The cat will give the user directions on how to play. Use this **say__ for __ secs** block and the words in the list below.

- *Hello, let's play a game.*
- *President Obama was born in 1961. Can you guess the month and date he was born?*
- *When you guess the month, use numbers for each month.*
- *For example, if you think he was born in January, use the number 1.*
- *If you think he was born in February, use the number 2 and so on.*

*You can link more than one **say___ for ___ secs** blocks together. You can also adjust the seconds. Make sure each direction stays on the screen long enough for the user to be able to read and understand all the directions.*

2. Now make the cat ask a question and wait until the user enters an answer. Pull out this block. Type, "*OK, let's start. In which month do you think President Obama was born?*"

You know that President Obama was born in August, the eighth month. The user does not know this, so s/he will choose any number. The number chosen will be either equal to 8 which is the correct answer, a number less than 8, or a number greater than 8. If the chosen number is less than 8 or a number greater than 8, the user's answer will be incorrect. There are only three possibilities.

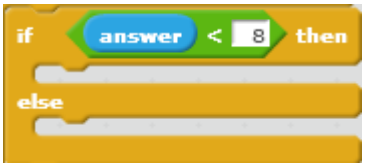
3. Assume that the user does not guess correctly on the first guess. The user does not choose 8. Then the user's number is less than 8, or it is a number greater than 8. This conditional has two outcomes. Pull out this **if ____then...else** block.

4. If the user's first answer is a number that is less than 8, that choice is any month before August. If this happens you need to write a script that says, "*if the number is less that 8*" say, *"No, he was born in a later month."*

To make this script, pull out these blocks.

5. Drop the **answer** block into the first part of the  block, and type 8 into the second part of that block.

6. Drop this new block into the space on the **if __then...else** block. The script now says, "If the answer is less than 8 then".



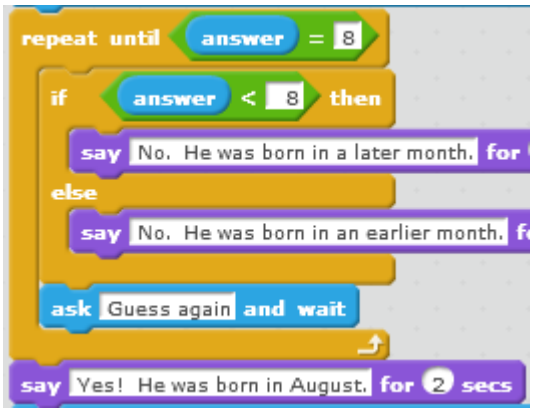
7. Tell the cat what to do if the guess is less than 8. Write a script that says if the answer is less than 8, say, *"No. He was born in a later month.."*

8. If the user does not pick 8 and does not pick a number that is less than 8, then s/he picks something else. The only thing left is a answer that is greater than 8. If it is something else, say, *"No. He was born in a earlier month."*

Notice how this script says this all. If the answer is less than 8 then say, "No. He was born in a later month." If the answer is something else say, "No. He was born in an earlier month." The script looks like this.

8. If the user guesses incorrectly, the cat must ask, *"Guess again."* Pull out the **ask____ and wait** block. Why do you place this block under the **if ____then...else** block?

9. Finally, you want this script to repeat until the user selects the number 8 for August. Pull out the **repeat until** block. Place the above script inside the **repeat until** block. Why put **answer = 8** in the top of the **repeat until** block?



10. Why add this **say** *Yes! He was born in August* **for 2 secs** block at the bottom of the script?

11. How would you attach this script to the directions for this game you created in question 1? Is sequence important?

12. Finish writing the script for this game. You still need to make the cat ask the user to guess on which date the President was born.

*You know that President Obama was born on the fourth day. The user could only choose the number 4 which is the correct answer, a number greater than 4, or something **else**: a number less than 4.*

13. Decide how you would make the cat tell the user that they guessed correctly when they guessed August 4? What block would you use? What would it say? Where would you place this block?

14. Test your script. Does it keep running until the user guesses August 4?

Challenge:

- Change the game. Make this game guess someone else's birthday. Add sound in your new game.

You do not have to rewrite the entire script to change the game. You could just change parts of the script or add new blocks to the script.

Variables

In programming, a variable is a name for a value that may change from time to time. Unlike school algebra, where a variable represents an unknown value that you could solve for, in computer programming variables are used as convenient names to represent numbers. For example *temperature* might represent the value 72 at one time, or 45 at another (depending on the weather). A variable can be thought of as a box in which numbers can be stored. In fact, variables are simply places in memory where the computer stores the values that you ask to be put there.

Variables are incredibly useful. The variable *gas* might represent the amount of fuel in a car, and you can write the program so that when *gas* is 0, the car can no longer move. Stopping the car at a fuel pump can increase the value of the variable *gas*. *Speed* might represent the speed of your car, which might change as well. In a computer game, variables such as *score, lives_left*, and *difficulty_level* are commonly used.

There are three things you can do with a variable: 1. Set or change its value. 2. Use its value in a calculation. 3. Test its value and have the program make decisions based on the value. All three of these ways of using a variable are used in this activity.

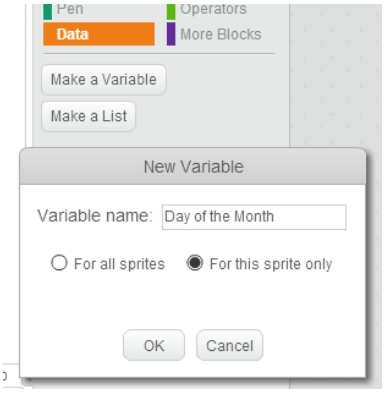
How Much Money?

Here is a problem to solve using two variables:

> The cat has one dollar. On the first of August, the cat's friend, Mick, decides to give the cat a gift. Mick plans to double the cat's money. In fact, Mick plans to double the amount of money the cat has every day of the month. So on the first day of August, the cat will have 2 dollars. On the second day, the cat will have 4 dollars. On the third day, the cat will have 8 dollars and so on until the last day of the month. How much money will the cat have on any day of the month? How much money will the cat have on the last day of the month?

Open a new Calendar Activity. https://scratch.mit.edu/projects/52455886/

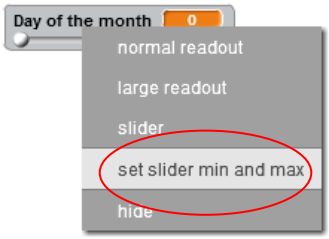
There are two variables in this problem. One is *money*. The amount of money the cat has each day will change or vary. The other variable is the *day of the month*. This variable will change from 1 to 31 depending on the date. So the script used to solve this problem will need two variables.



1. To create a variable, go to the DATA category and click Make a Variable Name this variable, *"Day of the month"*. Select For this sprite only.

2. Notice on the stage a gray box appears that says "*Day of the month 0*". This shows the name and current value of the variable. If you double-click on it, it will change to a different way to display the value (without the name). If you double-click again, a slider will appear. Create a slider now.

3. This slider allows you to vary numbers from 0 to 100. Since there are only 31 days in August, right click on the orange box holding the value zero. Select slider min and max. What numbers should you use to set the slider's minimum and maximum amount for this problem? Set the slider for this problem. Now you can use this slider to select any day of the month.



4. Create the second variable for this problem. Name it *amount of money*. You do not have to create a slider for this variable.
5. Hide any variable you do not want to show on the stage. Right click it and select hide, or under the DATA category uncheck the checkbox in front of the variable's name. Hide the *amount of money* variable.

Now that you have created both variables needed for this problem, you have to set each variable to a value.

- The *day of the month* variable can be set with the slider. For example, if you want the day of the month to be 5, move the slider to 5.
- The *amount of money* variable needs to be set to show the amount of money for each day.

6. To set the *amount of money* variable, pull out this block. Because the cat has 1 dollar before Mick gives him any money, set the **set amount of money to 0** block to 1.

7. At the start of the problem, you want the amount of money to be equal to 1. However, each day of the month the amount of money must be set to double. To do this:
   - pull out **another** **set amount of money to 0** block
   - pull out a **multiplication** block
   - pull out an *amount of money* variable
   - Write a script that says, set the *amount of money* to the *amount of money times*. Create this script to double the money each day.

8. Each day the cat calculates the amount of money it has. It needs to say that amount each day. Since the amount of money varies, it needs to say the variable, *amount of money,* each day. Which block in the LOOKS category can you combine with the *amount of money* variable to make it say the *amount of money* each day? Pull out that block. Which block goes into the blank to make the cat say the value of the variable for *amount of money?*

9. When you run this three block script, the cat says, "2". $2.00 is the amount of money the cat has on day 1: It stopped doubling on the first day because you only ran the script once.

10. The script doubles the amount of money the cat has, but you want it to execute repeatedly, once for each day. You can do this using a **repeat** block. Which of the blocks in this current script go inside of the repeat block? Why leave the **set amount of money to 1** block out of the **repeat** block?

*Think about the first block, **set amount of money to 1**. You want this on the first day to show the cat starts with $1, but do you ever need to set the amount of money to $1 again?*

11. Notice this **repeat** block is set to 10. This is fine for the 10th day of the month, because it runs the script 10 times, doubling the money 10 times. But, you want the date to change. You want the number of times the script repeats to match the date. So you need to use a variable.

12. Every time the date changes with the slider, you want the number of times the script runs to also change. Pull out the variable **Day of the Month** block from the DATA category. Drop this block into the **repeat** block to make the script repeat the same number of times as the date. Your script looks like this. Test your script. Does the cat say the amount of money on each date?

13. How would you change the script to make the cat only say the amount of money it would have on a certain date and not say all the amounts of money it would have up to that date. Test your changes. Do they work?

Challenge:

- Write a script that makes the cat travel to your birth date, stop on that date, and say the amount of money it would have on that day.
- Change this script to answer this problem:

The cat still starts with $1.00, but each day instead of doubling the money, its friend, Anna, gives it $1,000 each day? Will the cat get more money from Mick or from Anna? Use the two scripts to complete the chart below.

Day of the Month	Amount from Mick	Amount from Anna
1	$2	$1,001
2	4	2,001
3	6	3,001
4	8	4,001
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		

Talk About

- Why are variables important and helpful when you write scripts?
- Make a list of all the things you learned during these activities.
- What are some things that you need more help understanding?

Variables and Generalizations

Coders often look for ways to reuse parts of specific scripts in a more general way. They find patterns, use variables to represent changes in those patterns, and create a more general script. Finding and using general scripts saves coders time and effort.

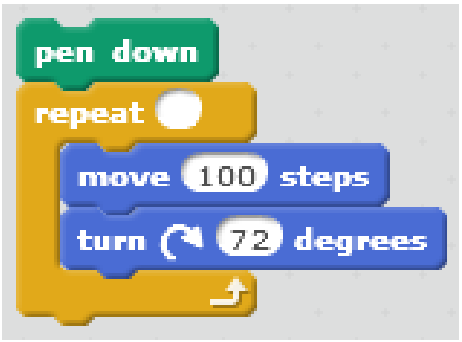
Regular Polygons

1. Open a new Scratch file. Shrink the cat sprite.
2. Look at the script on the left. Predict what shape it draws, then create and test the script. What shape does it draw?

3. Look at this script on the right.
   - How is it similar to the script above?
   - How is it different from the script above?
4. Predict what shape it will be draw, then create and test this script. What shape does it draw?

Remember, right click on a script and select duplicate to make a copy of the script.

5. Look at the script on the right.
   - How is it similar to the scripts above?
   - How is it different from the scripts above?
6. Decide which is the smallest number to input into the **repeat** block so the script makes a regular polygon Create and test this script. What regular polygon does it make?

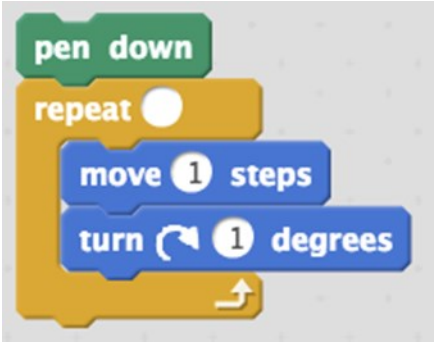


Think about the number of degrees needed for a full circle turn.

7. Look at this last script.
   - How is it similar to the scripts above?
   - How is it different from the scripts above?
8. Decide which number to input into the **turn** block so that the script makes a regular polygon and returns to the starting position, without tracing over its path more than once.

9. The script on the right does not have a number for the **repeat** block or for the **turn** block. Decide what the smallest number possible is to put into the repeat block and decide which number you would use in the turn block to create a regular octagon. Create and test your script.

10. Notice the sequence in this script is different. If you change the sequence in the other scripts in this activity would they still work? Why or why not? This script draws a polygon that looks like a circle, but it's really a polygon with many small sides. Decide which number you would use in the **repeat** block to create a closed polygon. Create and test your script.



11. Look back at the scripts you created for this activity and complete this table.

Repeat	Degrees	Shape
3	120	Triangle
4	90	Square
	72	
6		
		"Circle"

12. What patterns do you notice in the table?

Create Any Regular Polygon Using Generalizations and Variables

You can use the script that makes a regular polygon, and generalize it, or change it, by using variables to make a script that will make **any** regular polygon. You found that when you make a script for regular polygons only two things in the scripts change. These two things are the variables. To create a generalized script for **any** polygon, you need variables for those two things. One variable is for the number of polygon sides and another variable is for the number of degrees the cat turns. Open a new Scratch file.

You can see the polygons easier if you shrink the cat.

1. To make a new variable click the DATA category. Click make a variable. Name the variable *number of sides*.

2. Make the *number of sides* variable into a slider. A polygon must have a minimum of three sides, so set the slider min max to a minimum of 3. Decide on a maximum number and set the slider min max to that maximum.

3. Generalize the triangle script to create the script for **any** polygon.
   - Drop number of sides variable into the **repeat** block.
   - Now you can use the slider to choose the number of sides for your polygon.

To draw a triangle the cat turns 120 degrees three times for a total of 360 degrees (the number of degrees in a full circle turn).
- To draw a square the cat turns 90 degrees four times.
- To draw a pentagon the cat turns 72 degrees five times.
- To draw a hexagon the cat turns 60 degrees six times.

The number of degrees in the turn is 360 divided by the number of sides of the polygon.

4. Write a script that sets the variable for the number of degrees the cat turns to 360 divided by the number of sides. Pull out this block.

5. Make this block say, *360 / (number of sides)*.

6. Where would you drop this block into your triangle script to make a script for any polygon? Test your script. Does it create any polygon?

Spiral Polygons

Open a new Scratch file. Shrink the cat sprite. Make it very small.

Look closely at these two figures

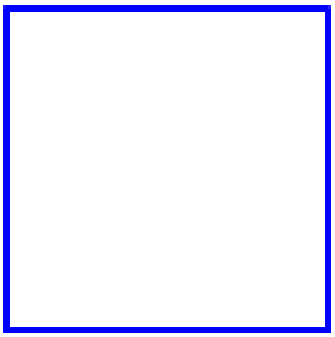


Figure 1

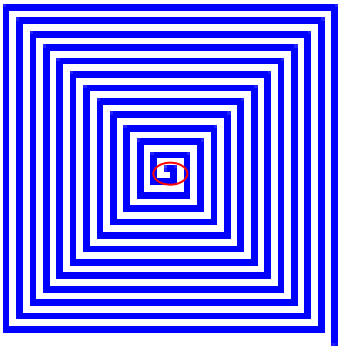


Figure 2

1. Notice that all four sides in Figure 1 are the same length. Look carefully at the line segments in Figure 2, starting with the small, center line segment (circled in red), the segments get longer and longer with each 90 degree turn. If each segment stays the same length, a square is formed. However, in Figure 2, because each segment gets a little longer, a spiral is formed in the shape of a square.

2. Look closely at these two figures.

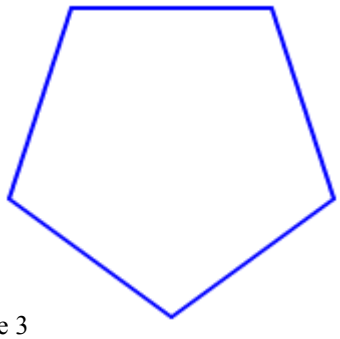


Figure 3

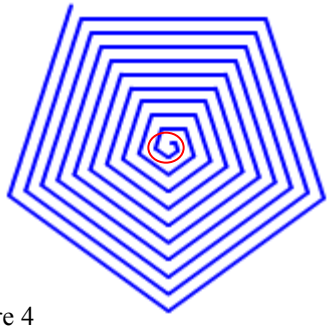


Figure 4

3. Notice that all five sides in Figure 3 are the same length. Because all five sides are the same length, a pentagon is formed. However, in Figure 4, starting with the small, center line segment (circled in red), the segments increase in length at each turn. This creates a spiral in the shape of a pentagon.

4. Figures 5 and 6 show two more spirals.

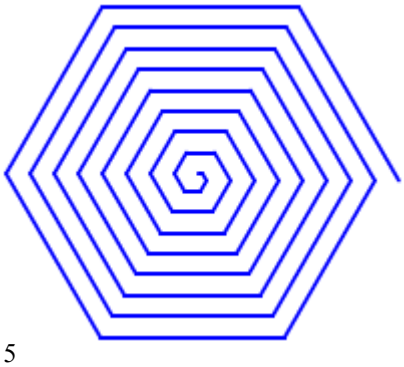


Figure 5

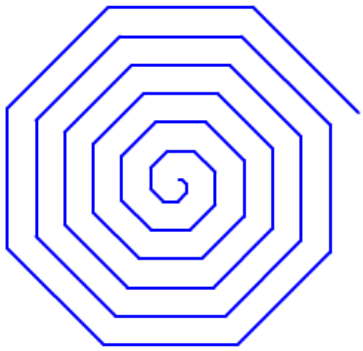
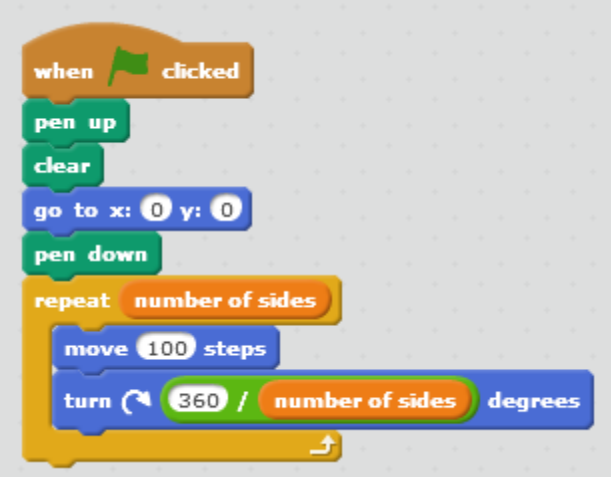


Figure 6

5. The line segments in Figures 5 and 6 get longer with each turn. Increasing the segment length each turn creates a spiral in the shape of a polygon. We can call these shapes spiral polygons.

6. Look at the regular polygon script again. The script at the right is an example of how it may have looked.



7. Make a remix of the regular polygon script. The remix creates any spiral polygon instead of any regular polygon. To do this, think about what needs to be different in the two scripts, and think about what should be similar in the two scripts. For both, you need a variable, with a slider, used to change the *number of sides*. In both, you also need this block from the OPERATORS category to tell the cat the number of degrees to turn. Create this block and this variable with a slider.

8. In the regular polygon script, you use the **move __ steps** block to tell the cat how long to make each side. In the spiral polygon, each side lengthens by 2 steps after each turn. Since the sides **change** length, a variable called, *side length* is needed. Create a *side length* variable.

10. The side length variable should be set to 2 for the first, small, center line segment in the spiral polygon. Use this block.

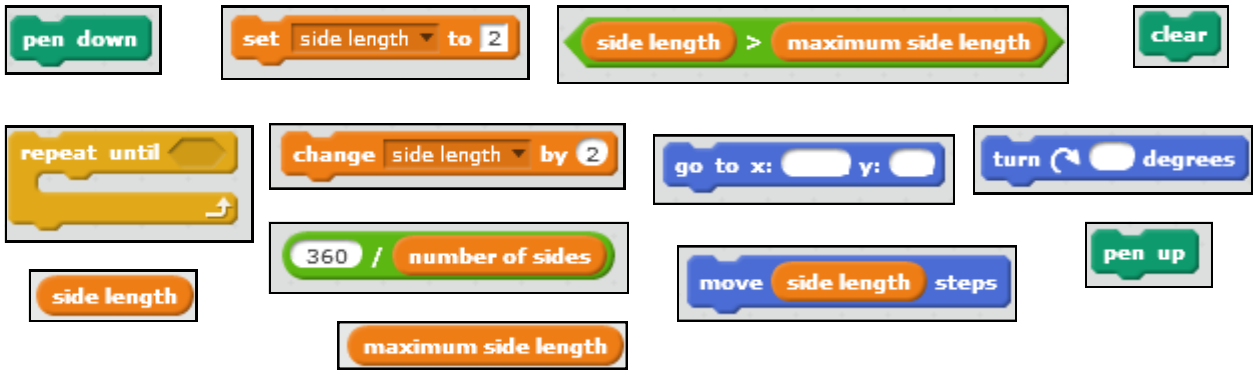
11. The regular polygon script used the **repeat** block and the variable number of sides to tell the cat how many line segments to draw. For the spiral polygon, the cat continues drawing line segments that get longer and longer. Instead of using the **repeat** block, use the **repeat, until __** block to stop the cat.

12. You have to stop the cat if the spiral is getting too big. Decide on a maximum side length. Since this could **change** depending on how large or small you want the spiral polygon to be, create another variable with a slider. Call it *maximum side length.*

13. You want the cat to stop drawing line segments when the *side length* is greater than the *maximum side length* you set with the slider. Use this block, and drop the variable names into the block. Then drop this new block into the **repeat until__** block.

14. Set the first, small, center-line segment with the **set side length to 2** block. After that first line segment is drawn, you want every other line segment in a spiral polygon to increase by 2. Use this block.

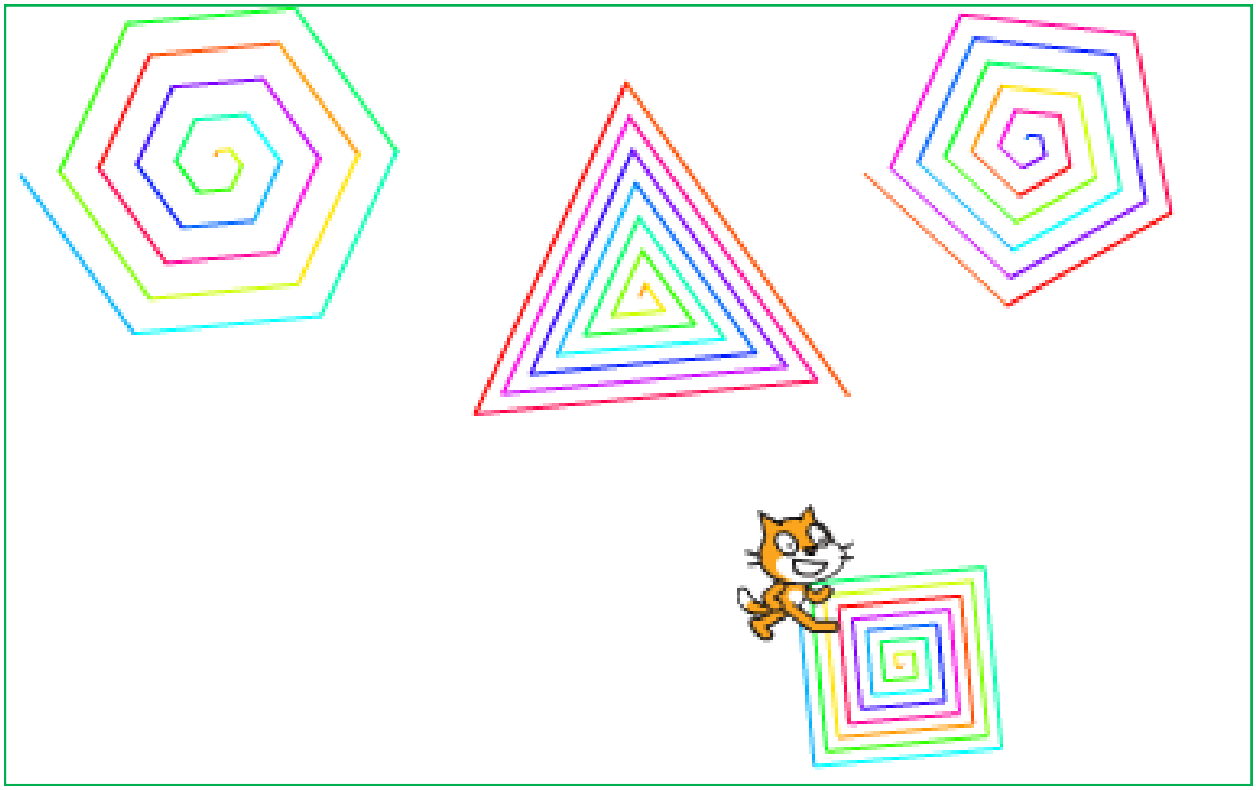
15. Use these to make a script that will create **any** spiral polygon.



16. Test your script.

Challenge:

- Make and run a script that fills the stage with colorful spirals like the image below. How can you use this block?



Talk About

- How can variables and generalization be used in Scratch?
- What are some of the things you learned during these activities?

Modularization

When creating activities, scripts can become long and complex. Instead of writing longer and longer scripts that are hard to debug if a problem occurs, use modularization to simplify the scripts. This process breaks the scripts down to manageable "chunks."

Blocks and More Blocks

Open a new Scratch file.

Study the script below. Notice how each section of this script creates a different, regular polygon.

The script inside this **repeat** block creates a triangle.

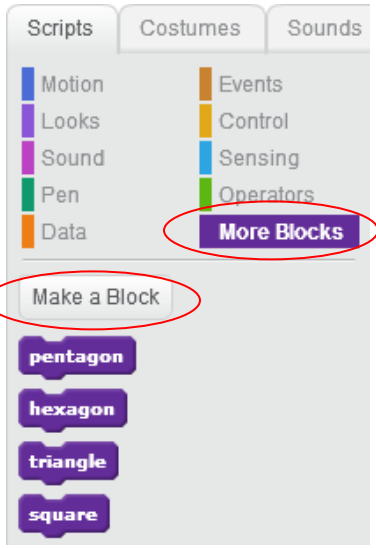
The script inside this **repeat** block creates a pentagon.

The script inside this **repeat** block creates a square.

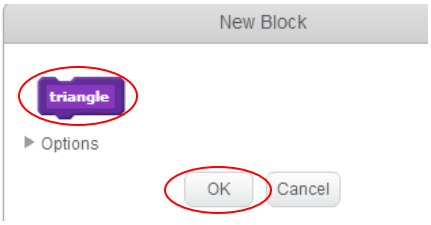
The script inside this **repeat** block creates a hexagon.

The script above is long and complicated. The script on the right is short and easy to understand. Both of these scripts create the same drawing. They both draw a triangle, pentagon, square and hexagon.

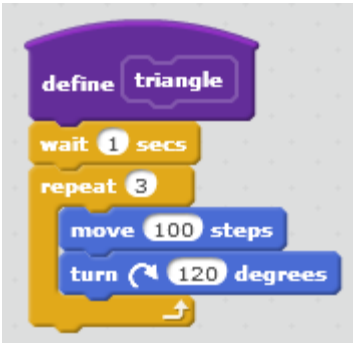
This simpler script is easier to understand and create. You can create a new "chunk" block for a "chunk" of code that you use over and over or for a section of code that does a specific task.



1. Click the MORE BLOCKS category. Then click Make a Block. Name the new block, **triangle.**

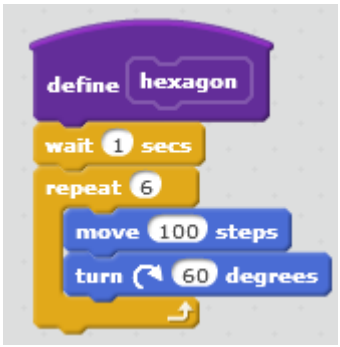


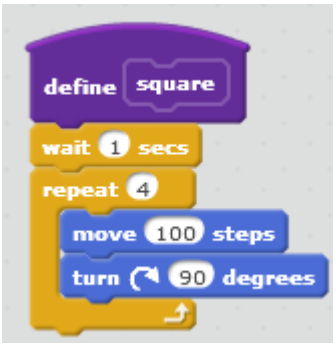
2. When you click "OK", a **define triangle** block appears in the Script Area.



3. Connect the script that draws a triangle to the new **define triangle** block. Now **triangle** is defined. The computer knows whenever you use **triangle** in a script, it should use this definition.

4. In the same way, create and define a block for a pentagon, a hexagon and a square. Combine the **pentagon, hexagon, square,** and **triangle** blocks to create the script shown at the bottom on the previous page. Test this script.



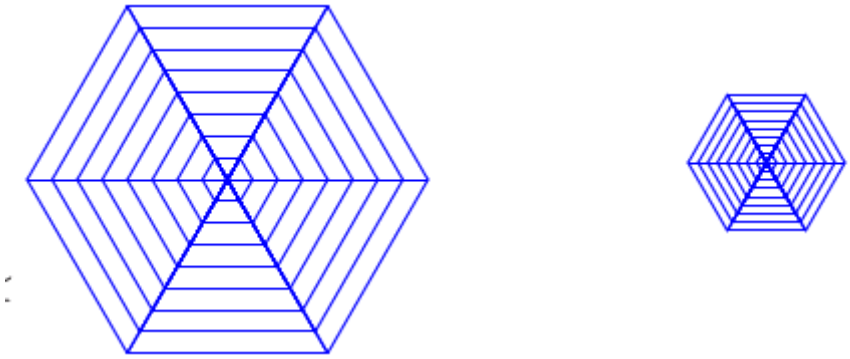


Challenge:

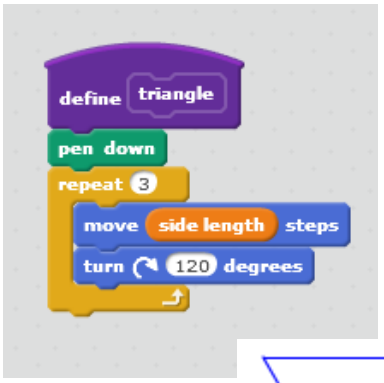
- Make a block that creates and defines an octagon.
- Make a block that creates and defines a spiral polygon.

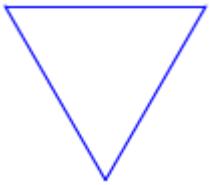
Spider Web

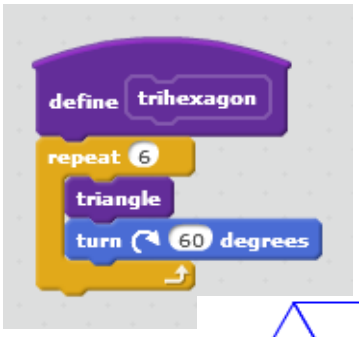
Open a new Scratch file. These spider webs were created using variables with sliders and modularization.

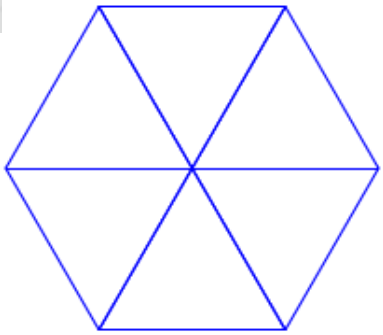


1. Recreate these scripts. One defines a block for **triangle.** The other uses **triangle** to create and define **tri hexagon,** a hexagon made of six triangles.





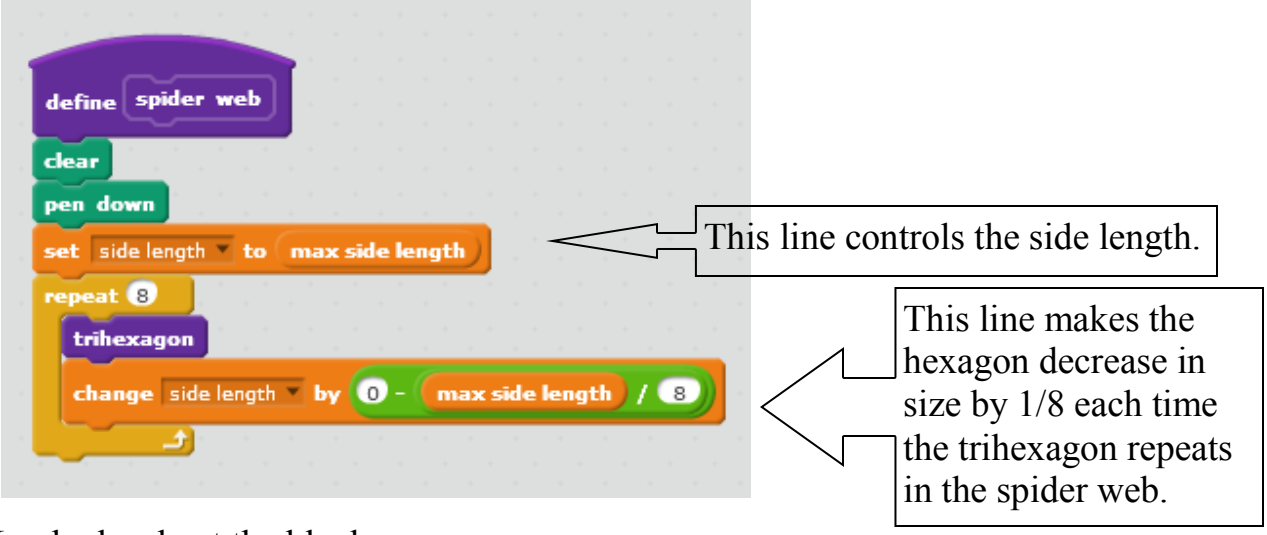




2. Create a variable for side length and another for max side length. Create a slider for each.

These variables for **side length** and for **max side length** allow you to vary the size of the spider web.

3. Write a script that uses **tri hexagon** to create and define **spider web**.



4. Look closely at the block shown here.

It is created using the **subtraction** and **division** blocks from the OPERATORS category. The **division** block is dropped into the second part of the **subtraction** block.

5. Create and define a block for **spider web.** Test your script. Does it work?

Challenge:

- The number 8 is used twice in this script. What happens if the number 8 is changed to another number?

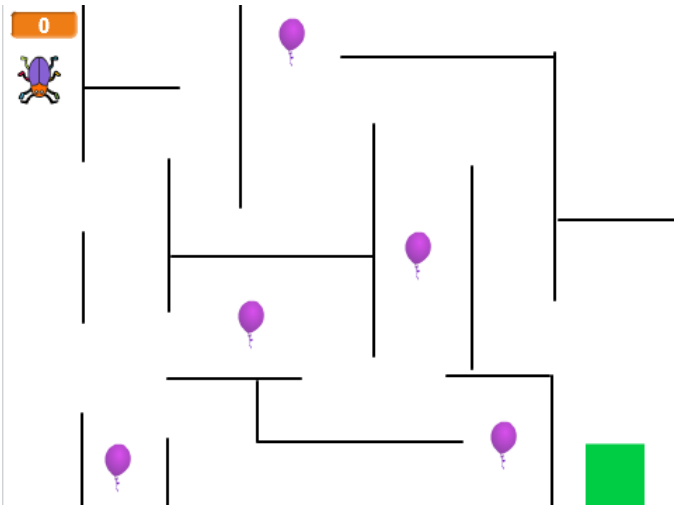
Talk About

- How is modularization helpful and important in Scratch?
- Look back at previous scripts you created. Could any of those scripts be made simpler by using modularization?

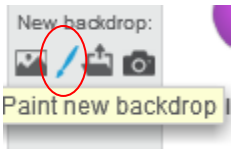
Remix

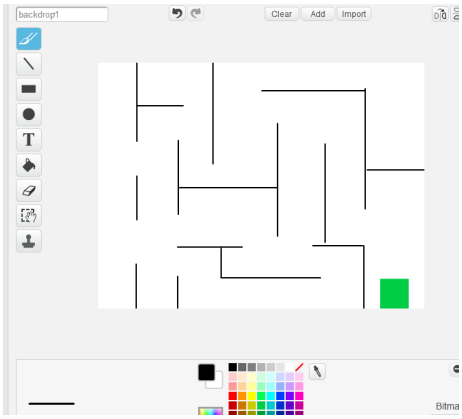
Open a new Scratch file.

The object of this game is to get the beetle to pop each balloon by touching it, and then get the beetle to the green block to win the game. The beetle is controlled by the arrow keys on the keyboard. Each time a balloon is touched by the beetle, it flashes off screen and increases the score by one point. When the beetle reaches the green finish line an announcement is made, "You win!"

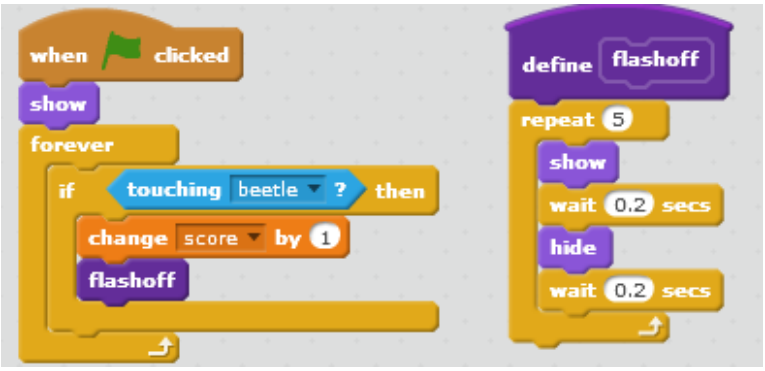


1. Create a background maze for your game. Click the paint brush under the Stage Backdrops to Paint new backdrop. Use the drawing tools to create a maze. Don't forget to include a green box for a finish line on your stage background. To create straight horizontal or vertical lines for the maze, press and hold the shift key while you place the line segments for your maze on the backdrop





2. Import at least five balloon sprites and place them in your maze. Create these scripts for each balloon.

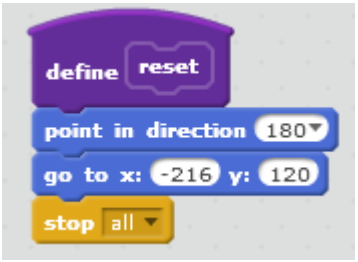


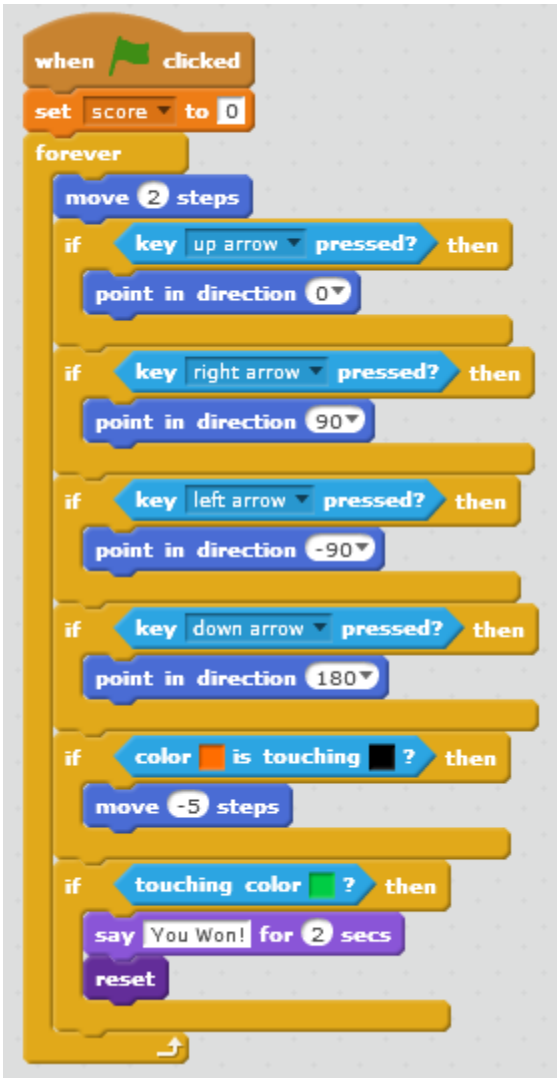
If you want two sprites to have the same script, left click on the first sprite's script you want to copy, drag it and hover over the second sprite below the stage. When you stop hovering, the script goes back to the Script Area for the first sprite. When you select the second sprite, the script will appear in its Script Area.

3. Create a beetle sprite. Delete the cat sprite.

4. Create a variable for score. Then create these scripts for the beetle.

*When you create and define **reset,** remember to use the location you want your beetle to start as the (x,y) location for this script.*





5. Test your game. Does it work? Does the beetle make it to the green finish line? Do the balloons flash off after they are touched by the beetle?

Challenge:

- Add a timer to your game. Look in the SENSING category. Make the beetle say the time every 0.5 second.
- Make the beetle go back to the starting location any time it touches a line of the maze.
- Use the (__ > __) block from the OPERATORS category to make the game end if the beetle is too slow in reaching the finish line.
- Add a sprite that appears and disappears at random locations on the maze. Shrink this sprite to fit your maze. If the beetle is touched by this sprite, make the beetle go back to the starting location.
- Think of other ideas that would make your remix fun. Create and test the scripts you need for your new ideas.

Show What You Know!

Well done! You've worked hard, and you've learned the basic elements of computer programming (sequence, iteration, conditionals, variables, and modularization)! Now is the time to get creative and make and share your own projects. What will you make? A game? A story? It's up to you!

Learn More

You've only Scratched the surface. There are many directions to go. Here are some features of Scratch to explore next:

- Use and change interesting backdrops (in the LOOKS category).
- Import, play, and control video (in the SENSING category)
- Import or create and control your own sound effects (in the SOUND category)
- Create new sprites automatically while the program is running, using "clones" (in the CONTROL category)
- Interact with the mouse (in the SENSING category)
- Create your own blocks to do whatever you can imagine! (in the MORE BLOCKS category)
- Create and use lists to store and manipulate data (in the DATA category)
- Learn about writing *recursive* programs. This is a technique, not a particular block. You'll have to read up on this on your own:
  - ◊ http://wiki.scratch.mit.edu/wiki/Recursion
  - ◊ http://wiki.scratch.mit.edu/wiki/Recursion_and_Fractals

Explore many more Scratch ideas at http://scratched.gse.harvard.edu.

Scratch Offline Editor

To work on projects without an internet connection, you can install the Scratch Offline Editor. With this you can create a new Scratch activity, edit projects, and save them to your local computer or to a thumb drive. Get the Scratch 2 Offline Editor here.

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

You can download a project from the Scratch site to your computer so you can work with it offline:

- Click File at the top of the stage.
- Select Download to your computer.
- Decide where you want to save this on your computer or thumb drive.
- Name the activity.
- Now you can open this activity and work offline. Notice when you open your activity you see "Scratch 2 Offline Editor" at the top of the window.