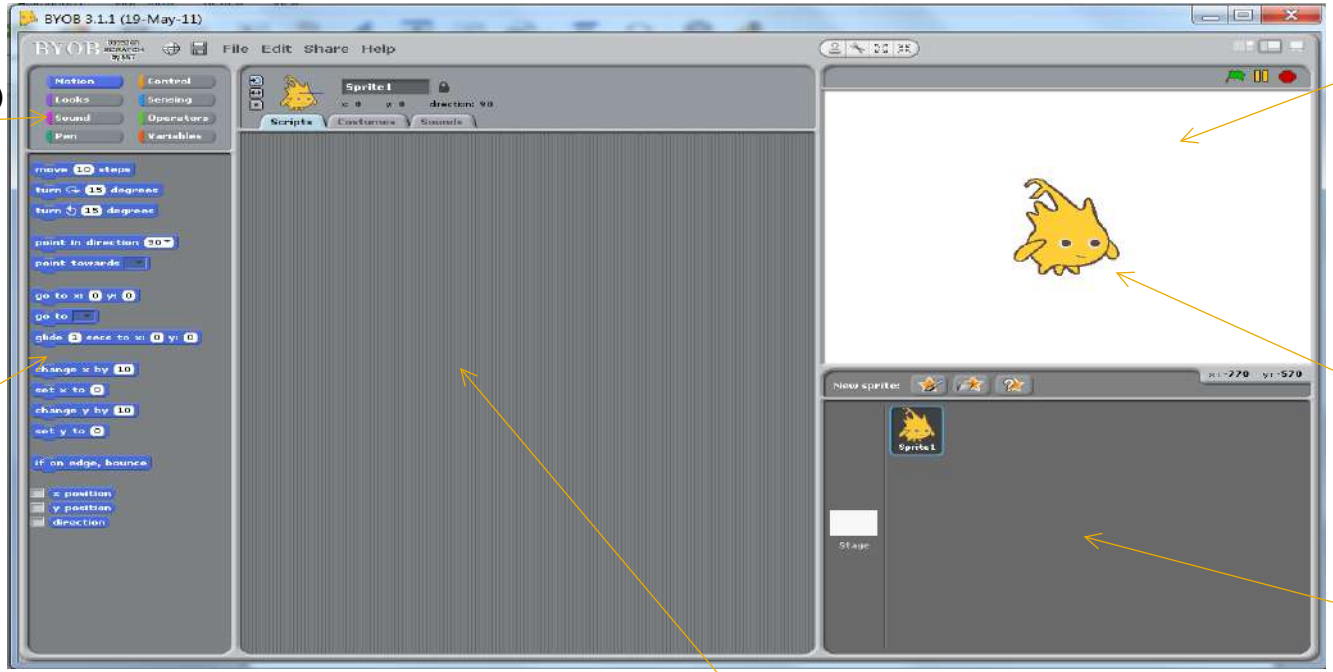
Mrs. Chapman

# AP Computer Science

# The BYOB Window / IDE

Tabs
(Block Categories)

Commands
Available to use




Stage

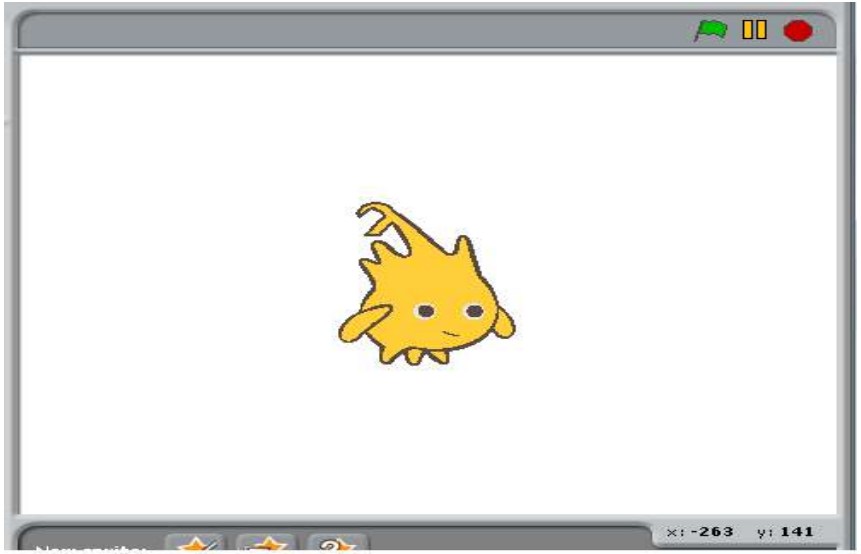
Sprite

All sprites in
this project

Script Area where you type your code

# The Stage / Editor Area

- The Stage is where the action happens
- You will add objects called sprites to the stage and they will act based on their scripts
- Notice the coordinates in the lower-right



# Class / Sprites

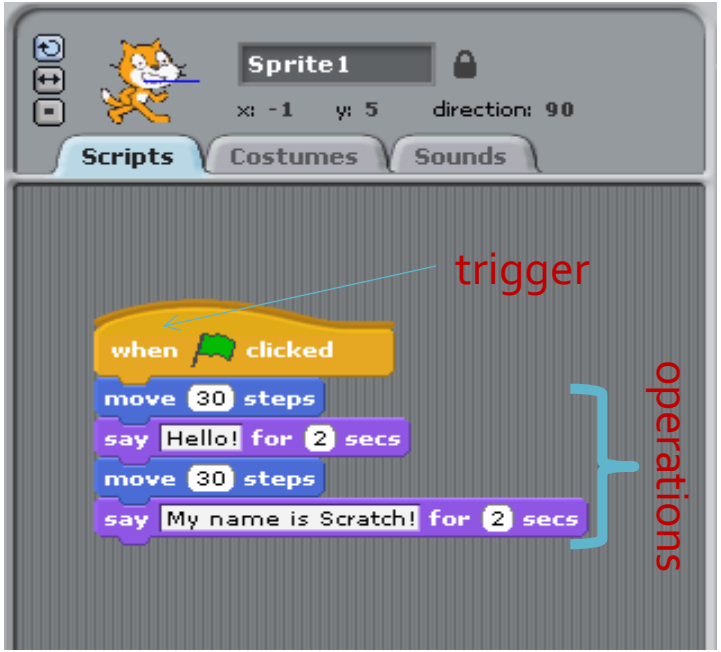
A class is a template in programming used to create an object.

A class in SNAP is a Sprite. You choose the sprite class and create an object from it. These are the main elements of your BYOB programs

- Each object created has one or more commands (methods) that determine how the object acts under different circumstances.
- Objects can have different costumes that change their appearance or attribute.
- BYOB provides a bunch of built-in sprites and costumes, or you can design your own.

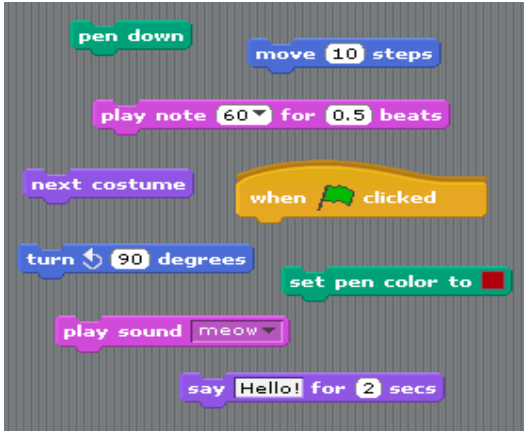
# Creating a Method

- A method is a group of code that performs a certain action.
- Methods control an objects actions.
  - You can also write methods to change the stage area.
- All methods in SNAP must have a trigger or event handler and some operations
- When the event handler happens, the object will perform the commands in order. - Sequential order one after the other.
- What does this method do?



# Blocks

- The different types of blocks are listed in the upper left of the BYOB window
  - "Motion" blocks cause objects to move
  - "Looks" blocks change an object's appearance
  - Etc.
- Each block corresponds to one "action"
- Notice the shapes!
- When a script runs, the actions occur in the order in which the blocks appear in the script.



# Using blocks to create methods

- *Exercise: Write an algorithm to do the following:*
  - *Move the sprite 25 steps,*
  - *Have the sprite say your name for 3 seconds*
  - *Turn the sprite around and move it back where it came from*
  - *Have the sprite say "I love BYOB!" for 5 seconds*

# Drawing

- Blocks in the Pen category allow your sprites to draw things on the stage
- The 

block puts a sprite's pen down on the stage. The 

block picks the pen up.
  - When a sprite's pen is down, moving will cause it to draw.
- You can also change the pen's color, size, or shade.

# Drawing

- *Exercise 1: Write an algorithm to make your sprite draw a square with 50 step sides.*
- *Exercise 2: Once you've done that, write an algorithm to draw two squares next to each other*
  - *The squares **should not** be connected by a line*

# Triggers/Controls

- Triggers tell an object to start executing
- There are four types of triggers:
  - When green flag is clicked
  - When I am clicked
  - When <*some key*> is pressed
  - When I receive a message
    - more on messages later

# Triggers/Controls

- *Exercise: Rewrite your square drawing algorithm to draw a square whenever the space bar is pressed. Clear the screen before each new square.*

# Threads

- You can build multiple scripts in the script area for any sprite (or the stage)
- All these scripts run *at the same time* (assuming their triggers occur)
- Each script is called a “thread”

# Threads

- *Exercise: Write scripts to allow the user to move a sprite around the stage with the arrow keys.*

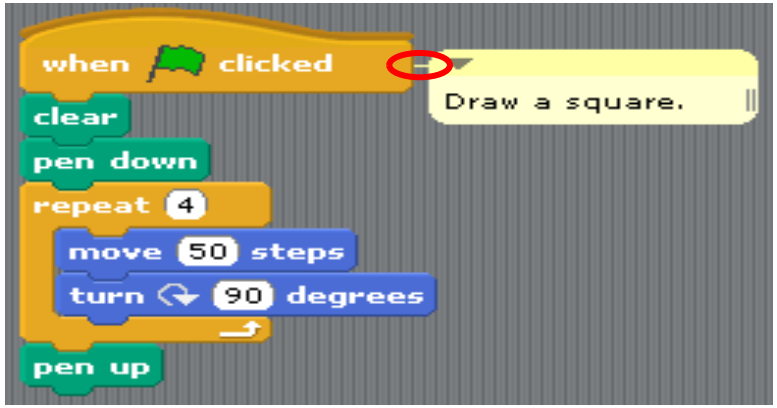
# Comments

- If you right-click on the script area, you get an option to "add comment"


- *Comments* are used to describe what's going on in the code. In BYOB, they look like this:
- They do not execute.

# Comments

- BYOB comments can be attached to blocks to indicate to what they are referring:

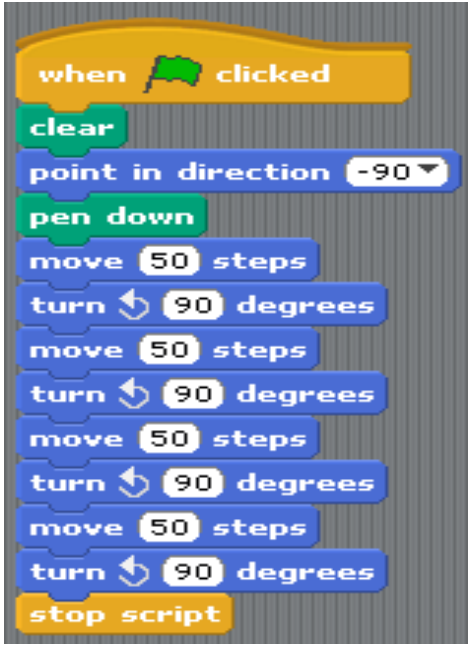


# Comments

- You should use comments to:
  - Describe the basic, high-level behavior of any script
  - Explain anything potentially unclear or tricky
  - Explain why you chose to do something a certain way if there were multiple options
  - Etc.
- Get in the habit of using comments now. You'll be graded on them (especially in Java)!

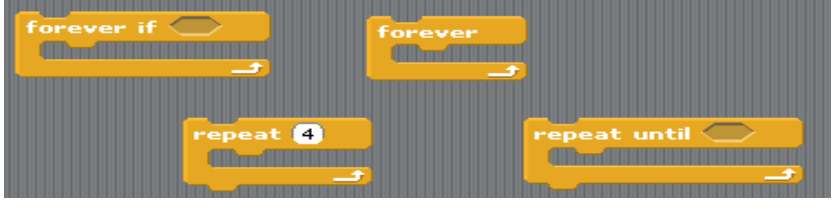
# Control Flow

- The order in which blocks are executed is called the "*control flow*" of the script
  - So far all our scripts have just executed blocks in the order given
- Consider our script to draw a square
  - Notice all the repeated code
  - Wouldn't it be nice if there were a way to simplify this?



# Loops

- *Loops* cause the object to repeat a certain set of commands multiple times without having to repeat the blocks
- Loop blocks in SNAP have the  symbol at their bottom right
  - This indicates that when the end of the loop is hit, the next block executed is back at the beginning
- There are several types of loops in SNAP:



# Loops

- *Exercise 1: Rewrite the script to draw a square using loops. Try not to repeat any code.*

- *Exercise 2: Now rewrite the script to draw two squares next to each other using loops. Again, try not to repeat code.*
  - *This is tricky!*

# Variables

- Thought exercise: How can we make objects move at different "speeds"?
- *Variables* allow us to store data and modify or retrieve it later
- Check out the Variables category
- Look around for built-in variables
  - What shape are variable blocks?

# Variables

- When we click "Make a variable" we get a dialog box
- The name can be anything you want


- "For all sprites" means all sprites/objects will be able to see and edit the variable. This is a global variable.
  - Why might this be useful? Why might it be dangerous?
- "For this sprite only" means only the current sprite can see and edit it. This is a local variable.

# Input

- You can ask the user for input using 

- The response is stored in 

  - Note that 

 is just a built-in variable
- Often, you'll be storing the input in a variable for later use

# Input

- *Exercise 1: Write an algorithm to do the following:*
  - *Ask the user for a number between 1 and 10*
  - *Draw that many squares*

- *Exercise 2: Write an algorithm to do the following:*
  - *Ask the user for a number between 1 and 10*
  - *Ask the user for a number between 1 and 255*
  - *Draw the first number of squares with the pen color set to the second number*

# Doing Arithmetic

- So far, we've only used simple numbers
- It would probably be nice if we could do some math
- Check out the Operators category
  - The first four blocks are your basic arithmetic operators
  - At the bottom are some more useful operations
  - As always, notice the shapes– where can we use these blocks?

# Doing Arithmetic

- *Exercise: Write an algorithm to do the following:*
  - *Ask the user for a number between 1 and 5*
  - *Draw twice that many squares*

# Booleans

- See the hexagon-shaped hole in 

? What goes there?
  - Look around for blocks with that shape. What does it look like they do?

- Hexagon-shaped blocks represent *Boolean expressions*
  - Named after 19th century English mathematician George Boole
- Boolean expressions evaluate to either *true* or *false*

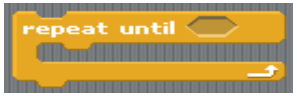
# Conditions

- Boolean expressions are used in *conditions*


- Conditions control whether or not certain blocks are executed
  - The blocks inside of an "if" are executed if and only if the condition is true
  - The blocks inside of an "else" are executed if and only if the condition is false
- Look at the "Sensing" category for lots of interesting things you can test

# Conditional Loops

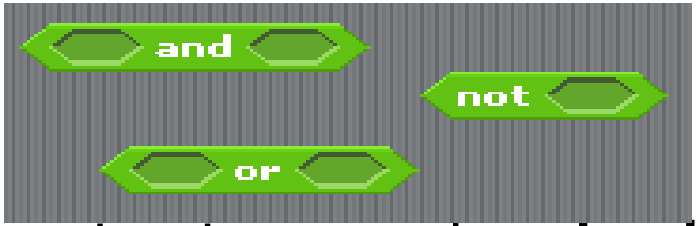
- You can also use conditions in loops
  - 

 is like forever, but the body only executes when the condition is true
    - Will stop then start again if things change
  - 

 loops until the condition is true, then moves on
- Play with these, as they can be quite useful

# Boolean Operators

- Boolean expressions can be combined in certain ways




- An *and* expression is true when **both** parts are true
- An *or* expression is true is when **at least one** part is true
- A *not* expression is true when the component expression is false

# Boolean Operators

- Truth Tables:

| AND | T | F |
|---|---|---|
| T | T | F |
| F | F | F |

| OR | T | F |
|---|---|---|
| T | T | T |
| F | T | F |

| | NOT |
|---|---|
| T | F |
| F | T |

# Conditions/Booleans

- *Exercise 1: Write an algorithm to do the following:*
  - *Generate a random number between 1 and 10*
  - *Draw a red square if the number is less than 6*
  - *Draw a blue square if the number is 6 or greater*
- *Exercise 2: Write an algorithm to do the following:*
  - *Generate two random numbers between 1 and 10*
  - *If both are less than 6, draw a red square*
  - *If both are 6 or greater, draw a blue square*
  - *Otherwise, draw a purple square*

# Events

- Sprites can cause each other to act in certain ways by using 

and 

- This sends out a message which can be picked up by other objects
- These messages are called *events*
- Events have unique names, and any sprite can broadcast or listen for any event

# Events

- *Exercise: Implement "Marco Polo"*
  - *Create one object at the center of the stage*
  - *Create another object at a random location and hide it*
  - *The arrow keys control the motion of the first object*
  - *When space is pressed, the first object should say "Marco" after which the second sprite should briefly show itself and say "Polo"*
  - *If the first sprite says "Marco" when it is touching the second, the second sprite should appear and say "Found me!"*