Computational Science Education

Mr. Michaud
www.nebomusic.net
Hierarchy of Technology Skills

1. **Passive Reception**: From creator to receiver with no action on receiver to alter flow of content. (watching a video)

2. **Active Research**: From creator to receiver with receiver selecting path and specific points of content. (Surfing the web)

3. **Interactive 1 way**: From creator to receiver through game model. Receiver must master a skill to progress through activity. (Skill and Drill)

4. **Communication / Expression**: User creates content and art with technology. Verbal / Text Images Plot Web

5. **Interactive 2 way**: creator and users interact. (Email, Blogs, Multi-User Virtual Environments)

6. **Data manipulation and Analysis**: User uses computer to interact and manipulate the content and data.

7. **Software creation / programming**: Users encodes logical thought and algorithms into computer. User tells computer what to do. Provides platforms and engines for above skill sets.

8. **Hardware creation**: User designs and assembles hardware to run, input, display software.
Why is Computational Science Important?

• Just as **Reading** and **Writing** allow us to encode and share our verbal thoughts . . .

• And **Math** allows us to model the world through number systems and operations . . .

• And **Music/Arts** allow us to encode our non-verbal thoughts into a dynamic system that changes over time . . .

• **Computational Science** allow humans to encode their logical/algorithmic thought in a dynamic system that changes over time.
In a way, CS combines Writing, Math, and Arts . . .

• Computational Science employs the
  • act of writing with the
  • language of mathematics in a
  • dynamic and changing system like music
  • to create algorithms to solve problems or engage in self expression.
Why the word ‘Computational’?

• **Computer Science** puts too much focus on the machine we picture as a screen, keyboard, touchscreen, mouse . . .

• **Computational Science** embodies the human activity of analyzing, designing, implementing, and testing hardware and software systems.

• As teachers – our job is teaching human beings how to be better humans through learning the art of Computational Science.
Goals for CS Activities for Classroom Teacher

• Algorithmic Solving
  • Build a series of steps and have computer/system follow
  • Build a series of steps to solve a problem

• Creative and Interactive Expression
  • Run ideas and interact

• Communication and Content Delivery
Software System Goals for CS Education with younger students

• Web based
• Visual Programming – limited typing/keyboarding
• Object Orientated
• Problem Solving Activities
• Create Exploration Activities
How can we add Computational Science during the instructional day?

• Leverage device usage already available to students (BYOD, classroom computers, labs)

• Set aside time for computational activities within the framework of other subjects (math, science, art)

• Finally – empower students to pursue CS activities during their non-classroom time.
Four Online CS Teaching Systems

• Scratch: [http://scratch.mit.edu](http://scratch.mit.edu)
  • Tile Based Programming
  • One of the Original CS Teaching Environments
  • Creative Tool

• Code.org: [http://code.org](http://code.org)
  • Tile Based Programming
  • Problem Solving Exercises

• EarSketch: [http://earsketch.gatech.edu](http://earsketch.gatech.edu)
  • Text Based Programming (Python)
  • Music Mixing

• SketchPad: [http://sketchpad.cc/](http://sketchpad.cc/)
  • Text Based Programming (Java and JavaScript)
  • Graphic Programming
Scratch
Scratch Description

• Launched in 2007

• Scratch is a free, block-based programming environment created by MIT to produce media and teach programming concepts.

• Join together block icons representing code, students create games, stories, and simulations in which programmable objects called "sprites" move and interact.

• Scratch's graphic block-based environment provides an excellent platform for introducing computer science concepts such as objects, methods, and conditional statements
Advantages for Scratch

- Easy and Fun!
- Visual – way we think
- Object orientated
- Flexible – low floor, high ceiling
- Runs on the Web
- Encourages Open Source model
- Free!!!
Scratch User Interface:
http://scratch.mit.edu/projects/editor/

World: Where the program takes place

Sprite: Any character or object in the program.

Scripts Pane: Where you enter the instructions for the Sprite

Tiles: Commands for your sprite
Scratch Blocks:
Move, Looks, Sound
Scratch Blocks: Pen, Control, Sensing
Types of Scratch Activities

• Games
• Simulations / Interactive Art
• Storytelling

http://scratch.mit.edu/projects/1930908/

http://scratch.mit.edu/projects/2906171/

http://scratch.mit.edu/projects/25687700/
Samples: Interdisciplinary uses for Scratch: Science

• Molecule Project: 5th Grade Science

Lesson Link: [http://nebomusic.net/techlesson07-5.html](http://nebomusic.net/techlesson07-5.html)
Samples: Interdisciplinary uses for Scratch: Science

• Orbit Simulation: 4th Grade Science

Lesson Link: http://nebomusic.net/techlesson07-4a.html
Scratch Sample: Sample: http://scratch.mit.edu/projects/48298382/
Samples: Interdisciplinary uses for Scratch: Science

• Amplitude Project: 4th Grade Science

Lesson Link: http://nebomusic.net/AmplitudeScratch.html
Scratch Sample: Sample: http://scratch.mit.edu/projects/48294962/
Samples: Interdisciplinary uses for Scratch: Math

• Polygon Project:

Lesson Link: http://nebomusic.net/PolyRobotProject.html
Scratch Sample: Sample: http://scratch.mit.edu/projects/26689031/
Samples: Interdisciplinary uses for Scratch: Music

• Xylophone Project:

Lesson Link: http://nebomusic.net/scratchxylophone.html
Scratch Sample: Sample: http://scratch.mit.edu/projects/25687700/
Links for Scratch

• [http://nebomusic.net/scratch.html](http://nebomusic.net/scratch.html)
• [http://nebomusic.net/scratch2lessons/](http://nebomusic.net/scratch2lessons/)
• [http://nebomusic.net/picoboardlessons/Scratch_Overview.pdf](http://nebomusic.net/picoboardlessons/Scratch_Overview.pdf)
• [http://scratch.mit.edu/](http://scratch.mit.edu/)
• [http://nebomusic.net/picoboardlessons/Scratch_Overview.pdf](http://nebomusic.net/picoboardlessons/Scratch_Overview.pdf)
Code.org

• Launched in 2013
• Dedicated to expanding participation in computer science by providing free web-based lessons and activities
• Provides a balance of activities geared for both boys and girls
Code.org: Problem Solving

• Scaled from Pre-Readers to Late Elementary
• Pre Readers:
  • Mouse Skills
  • Workflow in Tile Based Programming Environment

http://studio.code.org/s/course1
Creativity in Code.org

• Artist Activities

http://studio.code.org/s/artist/stage/1/puzzle/1

• Playlab: Storytelling and Event Driven Games

http://studio.code.org/s/playlab/stage/1/puzzle/1
Code.org: Advantages

• Self Directed – “Teacher-proof”
  • Activities can be implemented with little teacher training or background in Computer Science
  • Students can create accounts to track progress

• Pedagogically Appropriate for Pre-Typing Students
  • Focus on movement in 2D Space
  • Shapes and Colors of tiles reach to non-readers and non-typists

• Web based and Free!
Code.Org: Disadvantages

• Low Ceiling: Platform does not support variables, arrays and other structures key to Computer Science
• Unable to import student created graphics or sound
• Process of “Coding” seen primarily as a means to solve other people’s problems – not create your own work or expression.
Code.org Summary

• Appropriate for first experience in tile based programming.
• Useful for students younger than 8 years.
• Will need an extension plan for more advanced students or for students older than 10 years.
• Must balance use of Code.org with expressive/creative computing activities.
EarSketch: Web Based Python Music Mixing
Description of EarSketch

• Launched in 2012
• Programming Environment
• Python Based
• Web Based App or Installed System of Software
• API built in Python for Music Mixing
• NSF Funded project to encourage computational interest through the mixing and sharing of music.
• Curriculum and Social Media Site
• [http://earsketch.gatech.edu](http://earsketch.gatech.edu)
EarSketch Workstation

Music Sample Library

Music Viewer

Programing Area
Mixing Music

**Measures:** Time element in music. Starting and ending of clips

**Samples:** Musical Sounds placed on tracks starting and ending at measures.

**Tracks:** Layers of musical samples. Hold the different sounds.
Essential Elements we will use in Python:

- **Comments**
  
  # This is a comment – meant for Humans

- **Includes** – loading preset methods or data

  ```python
  from earsketch import *
  ```

- **Methods** – telling the computer “what to do”

  ```python
  fitMedia(drums, 1, 1, 5)
  ```

- **Variables** – Names for information stored by program

  ```python
  Beat1 = "0+++0+++0+0+0+++"
  ```
EarSketch Python Functions

- `insertMedia(file, track, measure, scale)`
- `insertMediaSection(file, track, location, start, end, scale)`
- `fitMedia(file, track, start, end)`
- `makeBeat(file, track, measure, beatString)`
- `randint(0,3) # returns a random integer`
“fitMedia” Function

fitMedia(file, track, start, end)

- Location of Media
- Which Track in Reaper
- Start measure.
- End Measure

Example:

```
fitMedia(HIP_HOP_DRUMS1_2M, 1, 1, 9)
```
Example:

```python
from earsketch import *
init()
setTempo(120)
fitMedia(HIPHOP_DUSTYPRECUSSION_002, 1, 1, 9)
finish()
```

Music Clip:
- **Track 1**
- **Start:** Measure 1
- **End:** Measure 4 (1 less than 5)

Place `fitMedia()` commands between the ‘setTempo()’ and ‘finish()’ commands.
Setting Volume Effects

- `setEffect(track, VOLUME, GAIN, level, start, level2, end)`

- Example

  ```
  setEffect(1, VOLUME, GAIN, -40, 1, 10, 5)
  ```
"makeBeat" Method

makeBeat(file, track, measure, BeatString)

- Location of Media Sound
- Which Track in Reaper
- What measure.

Example: “0+++0+++0+0+0+++”

Example:

makeBeat(drums, 1, 1, “0+0+0+++00-00+++”)


Other Effects: Delay

**DELAY**

DELAY creates a repeated echo-like delay of the original sound. A delay effect plays back the original audio as well as a delayed, quieter version of the original that sounds like an echo. After the first echo it plays an echo of the echo (even quieter), then an echo of the echo of the echo (still quieter), and so on until the echo dies out to nothing. With the delay effect, we can control how much time passes between each echo (delay time). If we set the delay time to match the length of a beat, we can create rhythmic effects with delay.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
<th>minValue</th>
<th>maxValue</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELAY_TIME</td>
<td>The time amount in milliseconds (ms) that the original track is delayed, and the time between successive repeats of the delay.</td>
<td>300.0</td>
<td>0.0</td>
<td>4000.0</td>
<td>1 setEffect(1, DELAY, DELAY_TIME, 1000.0)</td>
</tr>
<tr>
<td>DELAY_FEEDBACK</td>
<td>The relative amount of repeats that the delay generates. Higher values create more &quot;echoes&quot;. Be careful of applying &quot;too much&quot; feedback!</td>
<td>-3.0</td>
<td>-120.0</td>
<td>-1.0</td>
<td>1 setEffect(1, DELAY, DELAY_FEEDBACK, -20.0)</td>
</tr>
<tr>
<td>MIX</td>
<td>The percentage of the effected sound (wet) that is mixed with the original sound (dry). At its minimum value (0.0), no effect can be heard. At its maximum value (1.0), none of the original sound is heard - it is all effect.</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1 setEffect(1, DELAY, MIX, 0.4)</td>
</tr>
<tr>
<td>BYPASS</td>
<td>Whether the effect is &quot;on&quot; (1.0) or &quot;off&quot; (0.0). If the bypass of an effect is &quot;on&quot; (1.0), that means the audio going into the effect passes through, and comes out unaffected. Note that unlike other effect name/parameter pairs, the only valid values for BYPASS are 0.0 and 1.0.</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1 setEffect(1, DELAY, BYPASS, 1.0)</td>
</tr>
</tbody>
</table>

**setEffect(1, DELAY, DELAY_TIME, 500, 1)**

From [http://earsketch.gatech.edu/category/learning/reference/every-effect-explained](http://earsketch.gatech.edu/category/learning/reference/every-effect-explained)
Other Effects: Distortion

DISTORTION creates a “dirty” or “fuzzy” sound by overdriving the original sound. This compresses or clips the sound wave, adding overtones (higher frequencies related to the original sound). It is common to distort an electric guitar sound by “overdriving” the guitar amplifier. Modern music sometimes uses distortion to add a grungy or gritty effect or feel to the composition.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
<th>minValue</th>
<th>maxValue</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISTO_GAIN</td>
<td>The amount of overdrive of the original sound.</td>
<td>20.0</td>
<td>0.0</td>
<td>50.0</td>
<td>1 setEffect(1, DISTORTION, DISTO_GAIN, 25.0)</td>
</tr>
<tr>
<td>MIX</td>
<td>The percentage of the effected sound (wet) that is mixed with the original sound (dry). At its minimum value (0.0), no effect can be heard. At its maximum value (1.0), none of the original sound is heard — it is all effect.</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1 setEffect(1, DISTORTION, MIX, 0.4)</td>
</tr>
<tr>
<td>BYPASS</td>
<td>Whether the effect is “on” (1.0) or “off” (0.0). If the bypass of an effect is “on” (1.0), that means the audio going into the effect passes through, and comes out unaffected. Note that unlike other effect name/parameter pairs, the only valid values for BYPASS are 0.0 and 1.0.</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1 setEffect(1, DISTORTION, BYPASS, 1.0)</td>
</tr>
</tbody>
</table>

setEffect(1, DISTORTION, DISTO_GAIN, 30, 1)

From (http://earsketch.gatech.edu/category/learning/reference/every-effect-explained)
Other Effects: PAN

PAN affects the audio mix between the left and right channels. For example, if you were wearing headphones, changing the panning would affect whether you heard something from the left earcup or the right.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
<th>minValue</th>
<th>maxValue</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT_RIGHT</td>
<td>Specifies the left/right location of the original sound within the stereo field (0.0 is center, -100.0 is fully left, 100.0 is fully right).</td>
<td>0.0</td>
<td>-100.0</td>
<td>100.0</td>
<td><code>setEffect(1, PAN, LEFT_RIGHT, -50.0)</code></td>
</tr>
<tr>
<td>BYPASS</td>
<td>Whether the effect is &quot;on&quot; (1.0) or &quot;off&quot; (0.0). If the bypass of an effect is &quot;on&quot; (1.0), that means the audio going into the effect passes through, and comes out unaffected. Note that unlike other effect name/parameter pairs, the only valid values for BYPASS are 0.0 and 1.0.</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td><code>setEffect(1, PAN, BYPASS, 1.0)</code></td>
</tr>
</tbody>
</table>

`setEffect(1, PAN, LEFT_RIGHT, 100, 1)`

From [http://earsketch.gatech.edu/category/learning/reference/every-effect-explained](http://earsketch.gatech.edu/category/learning/reference/every-effect-explained)
Other Effects: PITCHSHIFT

**PITCHSHIFT**

PITCHSHIFT simply lowers or raises the sound by a specific pitch interval (PITCHSHIFT_SHIFT). It can be useful in helping multiple sound files sound better together or, contrastingly, to add a little bit of dissonance, if desired. It can also be used to create a harmony part for the same track by keeping the PITCHSHIFT_MIX setting somewhere towards its central position (0.5), to balance the volume between the effect and the original sound.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
<th>minValue</th>
<th>maxValue</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PITCHSHIFT_SHIFT</td>
<td>Specifies the amount to adjust the pitch of the original sound in semitones (and fractions of a semitone, given by values after the decimal point). 12 semitones equal 1 octave.</td>
<td>0.0</td>
<td>-12.0</td>
<td>12.0</td>
<td>setEffect(1, PITCHSHIFT, PITCHSHIFT_SHIFT, 0.6, 1)</td>
</tr>
<tr>
<td>MIX</td>
<td>The percentage of the effected sound (wet) that is mixed with the original sound (dry). At its minimum value (0.0), no effect can be heard. At its maximum value (1.0), none of the original sound is heard – it is all effect.</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>setEffect(1, PITCHSHIFT, MIX, 0.4)</td>
</tr>
<tr>
<td>BYPASS</td>
<td>Whether the effect is “on” (1.0) or “off” (0.0). If the bypass of an effect is “on” (1.0), that means the audio going into the effect passes through, and comes out unaffected. Note that unlike other effect name/parameter pairs, the only valid values for BYPASS are 0.0 and 1.0.</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>setEffect(1, PITCHSHIFT, BYPASS, 1.0)</td>
</tr>
</tbody>
</table>

**setEffect(1, PITCHSHIFT, PITCHSHIFT_SHIFT, 0.6, 1)**

From [http://earsketch.gatech.edu/category/learning/reference/every-effect-explained](http://earsketch.gatech.edu/category/learning/reference/every-effect-explained)
Other Effects: RINGMOD

RINGMOD multiplies the signals from two sounds together: your original sound and a pure sine wave (that sounds like a tuning fork). The effect of this multiplication sounds different at every frequency of the original sound, which creates a completely artificial-sounding result, as this type of sound could never occur naturally. Some parameter settings for this effect will likely produce recognizable-sounding effects similar to ones used in old science-fiction movies. It is useful experimenting with since there are a wide range of sounds that can be generated from your original sound.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
<th>minValue</th>
<th>maxValue</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>RINGMOD_MODFREQ</td>
<td>The frequency (in Hz) of the sine wave oscillator that is being multiplied into your original sound.</td>
<td>40.0</td>
<td>0.0</td>
<td>100.0</td>
<td>1 setEffect(1, RINGMOD, RINGMOD_MODFREQ, 70.0)</td>
</tr>
<tr>
<td>RINGMOD_FEEDBACK</td>
<td>The amount of effected sound that is fed-back into the effect. High values create more robotic-type sounds and sonic artifacts.</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
<td>1 setEffect(1, RINGMOD, RINGMOD_FEEDBACK, 30.0)</td>
</tr>
<tr>
<td>MIX</td>
<td>The percentage of the effected sound (wet) that is mixed with the original sound (dry). At its minimum value (0.0), no effect can be heard. At its maximum value (1.0), none of the original sound is heard – it is all effect.</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1 setEffect(1, RINGMOD, MIX, 0.4)</td>
</tr>
<tr>
<td>BYPASS</td>
<td>Whether the effect is “on” (1.0) or “off” (0.0). If the bypass of an effect is “on” (1.0), that means the audio going into the effect passes through, and comes out unaffected. Note that unlike other effect name/parameter pairs, the only valid values for BYPASS are 0.0 and 1.0.</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1 setEffect(1, RINGMOD, BYPASS, 1.0)</td>
</tr>
</tbody>
</table>

From [http://earsketch.gatech.edu/category/learning/reference/every-effect-explained](http://earsketch.gatech.edu/category/learning/reference/every-effect-explained)
Music Mixing Techniques

• Instrumentation: Bass, Drum, Melodic, Accompaniment
• Richard Devine Clips designed to work together by sub folder
• Keep clips in related keys (relative Major / minor)
• Tempos: (Rough Guide)
  • 88-92: Hip Hop / Funk
  • 94-110: Pop Music
  • 110 to 120: Marches (British: 110, American: 120)
  • 120 to 144: Techno
  • Faster than 144: Fast Jazz/Swing
• Less is more – vary the texture
• Music is organized in groups of 4 (Beats, measures, form)
• Organize clips into lists in code.
• Use makeBeat() and fills every 4 measures for variety and drive
Advantages to EarSketch

• Uses Python – a ‘real’ programming language that transfers to what we use in universities and the industry.
• Feels more ‘authentic’ for older students.
• Non-graphic expression. Some students connect with music quicker than with graphics/games. Studies show greater engagement in minority communities with EarSketch.
• Supported by extensive online curriculum.
Disadvantages to EarSketch

• Music comes with a lot of ‘baggage’. Students have strong musical tastes in middle and high school.
• Must be able to type and read. Requires a high level of attention to detail.
• Very difficult to import your own musical clips.
• Some bandwidth issues with online delivery on slower internet systems.
Processing / Sketchpad
Processing

• Invented in 2001 at MIT for use in visualizing data in for feedback in learning programming.
• Used by CS education community as an entrance to text based programming.
• Based on Java programming Language – an industry and research language.
• Widely used in University settings for research.
• Can be installed on Computer or used in a web-based Environment.
Processing Java Topics

- Data Types
- Two Functions in Processing
- Comments
- Canvas
- Shape Drawing
- Pen
- Colors
- Control Structures
A Side Note: Comments

• Comments are for humans
• Computer ignores comments
• Use lots of them in programming
  • Graded on Comments
  • Good style
  • Allows code to be used by others
• // Single Line Comment

• /* Multi line comments
• */
Data Types: Numbers and Letters

• **int**: Whole Numbers

• **Float**: Decimal Point Numbers

• **String**: Characters and Letters
Data Types: Other

• **boolean**: 1 Bit value storing ‘true’ or ‘false’ (Sometimes ‘1’ or ‘0’)

• **char**: 16 Bit Unicode character ranging from \u0000 to \uffff (Hexidecimal)
  • This is a single Letter or character

• **String**: (This is actually a class)
  • java.lang.String
  • ‘string’ of characters: like “Mr. Michaud”
Declaring Data

• Example:

```java
int myAge = 14;
```

Other Examples:

```java
String greetings = "Hello Class";
char letter = "a";
boolean state = false;
```
Operators

- String name = "Bob";
  // Used to Assign a value

- name == "Bob"  // ‘is equal to’
- 4 * 5         // multiply
- 20 / 4        // Divide
- 4 + 5         // Addition
- 4 - 5         // Subtraction
- 5 % 4         // Modulo
- 5 > 4
- 4 < 5
- 5 >= 4
- 4 <= 5
Logic

• &&   // Means “and”
• ||    // Means “or”
Two Functions in Processing

- All Processing Programs must have these two functions:
  - `void setup()`
  - `void draw()`
Canvas
Canvas Functions

• `size(width, height);` // Sets the Size of the Canvas

• `background(red, green, blue);` // Sets the color of the canvas
Canvas Functions

```java
// Setup Function

void setup() {
    size(600, 400); // sets size
    background(0, 0, 255); // sets color
}

void draw() {
}
```
Shape Drawing

• Key Functions:
  • `rect(xPos, yPos, width, height);`
  • `ellipse(xPos, yPos, width, height);`
  • `triangle(xPos1, yPos1, xPos2, yPos2, xPos3, yPos3);`
  • `fill(red, green, blue);`
  • `stroke(red, green, blue);`
Shape Drawing

```java
// Setup Function

void setup() {
    size(600, 400); // sets size
    background(0, 0, 255); // sets color
    fill(0, 255, 0); // Green
    rect(50, 50, 100, 100); // Rectangle
    fill(255, 0, 0); // Red
    ellipse(200, 200, 50, 50); // Circle
}

void draw() {
}
```
For Loop

• Repeats section of code while counting up or down with an index variable

• Example

```java
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

Returns:
0 1 2 3 4 5 6 7 8 9
for (int i = 0; i < 10; i++) {}

• i++ means "i = i + 1"

• int i means "integer i"

• for (int i = 0; i < 10; i++) means "For index variable i starting at 0, while i is less than 10, count be 1."
Conditional Statements

• ‘if statement’: Checks if a given statement or expression is true and then executes a section of code

```java
if (score > 9) {
    textScore.setText("You Win");
}
```
References

• Key Vocabulary:
  (http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html)

• Another Good Reference:
  http://processing.org/reference/

• Processing Lessons on nebomusic.net
  http://nebomusic.net/processinglessons/
Question and Answer time . . .