Exercises

• Create a DizzyTurtle class
  – That turns a bit to the left and goes forward when asked to go forward
  – And turns a bit to the right and goes backward when asked to go backward

• Create a SlowTurtle class
  – That only goes forward and backward by 50 instead of 100 if you don’t tell it how much to go forward or backward

• Create a StubbornTurtle class
  – Has a 50% chance of doing what you ask
Today

- A brief review
  - Python
  - Java (compared with Python)
- Next Steps
- Survey
- Course Evaluation
Today

- A brief review
  - Python
  - Java (compared with Python)
- Next Steps
- Survey
- Course Evaluation
- Scaling pictures
- Sound
- Scaling sound is frequency shifting
- Recursion
PROBLEMS

- Scaling pictures
- Sound
- Scaling sound is frequency shifting
- Recursion
Scaling

Scaling a picture (smaller or larger) has to do with sampling the source picture differently

- When we just copy, we sample every pixel
- If we want a smaller copy, we skip some pixels
  - We sample fewer pixels
- If we want a larger copy, we duplicate some pixels
  - We over-sample some pixels
Scaling the picture down

```python
def copyPictureHalfAsBig(file):
    # Set up the source and target pictures
    pic = makePicture(file)
    canvasFile = getMediaPath("7inX95in.jpg")
    canvas = makePicture(canvasFile)
    # Now, do the actual copying
    sourceX = 45
    for targetX in range(100,100+((200-45)/2)):
        sourceY = 25
        for targetY in range(100,100+((200-25)/2)):
            color = getColor(getPixel(pic,sourceX,sourceY))
            setColor(getPixel(canvas,targetX,targetY), color)
            sourceY = sourceY + 2
            sourceX = sourceX + 2
    show(pic)
    show(canvas)
    return canvas

>>> barbFile = pickAFile()
>>> setMediaPath()
>>> smallPic = copyPictureHalfAsBig(barbFile)
```
PROBLEMS

• Scaling pictures
• Sound
• Scaling sound is frequency shifting
• Recursion
How sound works: Acoustics, the physics of sound

- Sounds are waves of air pressure
  - **Sound comes in cycles**
  - **The frequency of a wave is the number of cycles per second (cps), or Hertz**
    - (Complex sounds have more than one frequency in them.)
  - **The amplitude is the maximum height of the wave**
Sine wave

**recipe 70**

def sineWave( freq, amplitude ) :
    mySound = getMediaPath("sec1silence.wav")
    buildSin = makeSound(mySound)
    sr = getSamplingRate(buildSin)  # sampling rate
    interval = 1.0 / freq           # interval of sample
    samplesPerCycle = interval * sr  # samples / cycle
    maxCycle = 2 * pi
    for pos in range( 1, getLength( buildSin ) + 1 ) :
        rawSample = sin(( pos / samplesPerCycle) * maxCycle)
        sampleVal = int( amplitude * rawSample )
        setValueAt( buildSin, pos, sampleVal )
    return buildSin
Square wave

**recipe 72**

```python
def squareWave( freq, amplitude ) :
    mySound = getMediaPath(“sec1silence.wav”)  
    square = makeSound(mySound)  
    samplingRate = getSamplingRate(square)  # sampling rate  
    seconds = 1  
    interval = 1.0 * seconds / freq  # interval of sample  
    samplesPerCycle = interval * samplingRate  # samples / cycle  
    samplesPerHalfCycle = int(samplesPerCycle / 2)  
    sampleVal = amplitude  
    i = 1  
    for s in range( 1, getLength( square ) + 1 ) :  
        if (i > samplesPerHalfCycle):  
            sampleVal = sampleVal * -1  
            i = 0  
            setSampleValueAt( square,s, sampleVal )  
        i = i + 1  
    return square
```
def triangleWave( freq ):
    amplitude = 6000
    samplingRate = 22050  # sampling rate
    seconds = 1
    triangle = makeEmptySound( seconds )  # create a sound object (the book uses “sec1silence.wav”)
    interval = 1.0 * seconds / freq  # interval of sample
    samplesPerCycle = interval * samplingRate  # samples / cycle
    samplesPerHalfCycle = int(samplesPerCycle / 2)
    increment = int( amplitude / samplesPerHalfCycle )
    sampleVal = -amplitude
    i = 1
    for s in range( 1, samplingRate + 1 ) :
        if (i > samplesPerCycle):
            increment = increment * -1
            i = 0
        sampleVal = sampleVal + increment
        setSampleValueAt( triangle, s, sampleVal )
        i = i + 1
    return triangle  # return the sound (the book says play)
Adding

- recipe 71 (part 2)

```python
def addSounds(sound1, sound2):
    for index in range(1, getLength(sound1) + 1):
        s1Sample = getSampleValueAt(sound1, index)
        s2Sample = getSampleValueAt(sound2, index)
        setSampleValueAt(sound2, index, s1Sample + s2Sample)
    return sound2
```
**recipe 64**

```python
def echoes( soundFile, delay, num ) :
s1 = makeSound( soundFile )
ends1 = getLength( s1 )
ends2 = ends1 + (delay - num)
s2 = makeEmptySound(1 + int( ends2 / getSamplingRate(s1) ) )
echoAmplitude = 1.0
for echoCount in range( 1, num + 1 ) :
    echoAmplitude = echoAmplitude * 0.6  # each echo is 60% of previous
for posn1 in range( 1, ends1 ) :
    posn2 = posn1 + (delay* echoCount )
    values1 = getSampleValueAt( s1, posn1 ) * echoAmplitude
    values2 = getSampleValueAt( s2, posn2)
    setValueAt( s2, posn2, values1 + values2 )
return s2
```
PROBLEMS

- Scaling pictures
- Sound
- Scaling sound is frequency shifting
- Recursion
Shifting the frequency

- recipe 68, modified
- how sampling keyboards work....

```python
def shift( soundFile, factor ) :
    source = makeSound( soundFile )
    target = makeSound( soundFile )
    sourceIndex = 1
    sourceLength = getLength( source )

    for targetIndex in range( 1, sourceLength + 1 ) :
        setSampleValueAt( target, targetIndex, getSampleValueAt( source, int( sourceIndex )))
        sourceIndex = sourceIndex + factor
        if sourceIndex > sourceLength :
            sourceIndex = 1

    return target
```
Scaling pictures

Sound

Scaling sound is frequency shifting

Recursion
A very powerful idea: Recursion

- Recursion is writing functions that call *themselves*.
- When you write a recursive function, you write (at least) two pieces:
  - What to do if the input is the smallest possible datum,
  - What to do if the input is larger so that you:
    - (a) process one piece of the data (the “head”)
    - (b) call the function to deal with the rest. (“rest”)
Factorial -- the classic recursive function

def factorial( number ):
    if number == 1:
        return number
    else:
        return number * factorial( number - 1.0 )
Today

- A brief review
  - Python
  - Java (compared with Python)
- Next Steps
- Survey
- Course Evaluation
Hey, remember that stuff we used to make HTML do stuff?
# JavaScript vs. Python

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>Javascript</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>write code</strong></td>
<td><strong>JES</strong></td>
<td><strong>text editor</strong></td>
</tr>
<tr>
<td><strong>interpret/execute</strong></td>
<td><strong>JES</strong></td>
<td><strong>browser (IE, Firefox, ...)</strong></td>
</tr>
<tr>
<td><strong>context</strong></td>
<td><strong>execute in command area</strong></td>
<td><strong>in HTML only</strong></td>
</tr>
<tr>
<td><strong>variables</strong></td>
<td><strong>just use</strong></td>
<td><strong>declare</strong></td>
</tr>
<tr>
<td><strong>blocks</strong></td>
<td><strong>“:” + indent</strong></td>
<td><strong>“{“ + “}”</strong></td>
</tr>
<tr>
<td><strong>termination</strong></td>
<td><strong>new line</strong></td>
<td><strong>“;”</strong></td>
</tr>
</tbody>
</table>
JavaScript vs. Python

- JavaScript’s syntax is much like other programming languages.
- JavaScript can’t do everything that Python can.
- Python is a more full-featured programming language.
- But Python can’t be embedded inside of HTML.
  - (Well, not cross-platform. It can on Windows with a Python plugin for your browser.)
Be sure to review the lecture sides!
# Python vs. Javascript vs. Java

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>Javascript</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>compile or interpret ?</td>
<td>interpret</td>
<td>interpret</td>
<td>compile</td>
</tr>
<tr>
<td>by ?</td>
<td>JES</td>
<td>browser</td>
<td>DrJava</td>
</tr>
<tr>
<td>develop &amp; debug in?</td>
<td>JES</td>
<td>text editor - no debugger</td>
<td>DrJava</td>
</tr>
<tr>
<td>Command line test?</td>
<td>&gt;&gt;&gt;</td>
<td>code view</td>
<td></td>
</tr>
<tr>
<td>performance</td>
<td>slow</td>
<td>not an issue</td>
<td>fast</td>
</tr>
</tbody>
</table>

Note: there are other Python and Java environments. Later Java course will probably use “Eclipse”.
## Syntax and Appearance

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>Javascript</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>end of line of code?</td>
<td>new line</td>
<td>&quot;,&quot;</td>
<td>&quot;,&quot;</td>
</tr>
<tr>
<td>blocks defined</td>
<td>indentation: &lt;tab&gt; or &lt;spaces&gt;</td>
<td>{ ... }</td>
<td>{ ... }</td>
</tr>
<tr>
<td>appearance</td>
<td>wide</td>
<td>like HTML: messy, wide, &amp; tall</td>
<td>tall</td>
</tr>
</tbody>
</table>
## Variable rules

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>Javascript</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>declare?</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>declaration examples</td>
<td>var x; var x = 10;</td>
<td>int x; int x = 10;</td>
<td></td>
</tr>
<tr>
<td>scoping</td>
<td>by class or global</td>
<td>w/in enclosing block</td>
<td>w/in enclosing block</td>
</tr>
<tr>
<td>can contain “.”?</td>
<td>all use “dot” notation to refer to a method in a class so “.” is NOT allowed in a name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Python</td>
<td>Javascript</td>
<td>Java</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------</td>
<td>--------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Functions?</td>
<td>yes</td>
<td>yes</td>
<td>sort of</td>
</tr>
<tr>
<td>functional</td>
<td>yes</td>
<td>not exactly</td>
<td>no</td>
</tr>
<tr>
<td>programing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td>allowed, but not required</td>
<td>required, mostly built-in (“document”)</td>
<td>required</td>
</tr>
<tr>
<td>Methods</td>
<td>allowed, but not required</td>
<td>required</td>
<td>required</td>
</tr>
<tr>
<td>Object-oriented</td>
<td>supported</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
# More syntax

<table>
<thead>
<tr>
<th>Use of &quot;=&quot;</th>
<th>Python</th>
<th>Javascript</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>varName =</td>
<td>flexible</td>
<td>type varName = expression</td>
<td></td>
</tr>
<tr>
<td></td>
<td>var varName = expression</td>
<td>Class varNam = expression</td>
<td></td>
</tr>
<tr>
<td>++ / --</td>
<td>allowed</td>
<td>often used</td>
<td>often used</td>
</tr>
<tr>
<td>Uniqueness of variable name</td>
<td>var, class, function(), method() overlay</td>
<td>var, class, method, method(params) are different</td>
<td>var, class, method, method(params) are different</td>
</tr>
<tr>
<td></td>
<td>class = consMethod</td>
<td>class = consMethod</td>
<td></td>
</tr>
</tbody>
</table>
## Even more objects & methods

<table>
<thead>
<tr>
<th>Definition</th>
<th>Python</th>
<th>Javascript</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>defining a Class</td>
<td><code>class ClassNam(ofClass):</code></td>
<td><code>public class ClassNam</code> <code>{ ... }</code></td>
<td><code>public class ClassNam</code> <code>{ ... }</code></td>
</tr>
<tr>
<td>constructor</td>
<td><code>def __INIT__/Class(self, arguments):</code></td>
<td><code>public ClassNam()</code> <code>{ ... }</code></td>
<td><code>public ClassNam()</code> <code>{ ... }</code></td>
</tr>
<tr>
<td>polymorphism</td>
<td>all support inheritance and polymorphism so that special methods w/ same name can be created for each class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom-up</td>
<td>pretty good</td>
<td>good</td>
<td>not so good</td>
</tr>
<tr>
<td>top-down</td>
<td>OK, but keep it simple</td>
<td>not so good</td>
<td>better</td>
</tr>
</tbody>
</table>
# Loops Compared

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>For each</strong></td>
<td><code>for (Pixel pxlObj: pixlArray) { }</code></td>
<td><code>for pxl in getPixel(pic):</code></td>
</tr>
<tr>
<td><strong>For index in range</strong></td>
<td><code>for (int i=0; i &lt; pixelArray.length; i++ ) { }</code></td>
<td><code>for i in range(0, getLength(pic)+1):</code></td>
</tr>
<tr>
<td><strong>For index in range for source &amp; target</strong></td>
<td><code>for (int sX=0, tX = 0; sX &lt; pixelArray.length; sX++, tX++ ) { }</code></td>
<td><code>tX = 0</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>for sX in range(0,getLength(pic)+1):</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>......</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>tX = tX+1</code></td>
</tr>
<tr>
<td>swapBackground &amp; chromakey</td>
<td>Java</td>
<td>Python</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>overall structure</td>
<td>name, init, loop(s)</td>
<td></td>
</tr>
<tr>
<td>loop structure</td>
<td>nested nested</td>
<td></td>
</tr>
<tr>
<td>algorithm</td>
<td>swapBackground: test each pixel to see if same as background only, replace with new background</td>
<td>chromakey: test each pixel to see if green+red &lt; blue, replace with new background</td>
</tr>
</tbody>
</table>
## Python & Java differences

<table>
<thead>
<tr>
<th>swapBackground &amp; chromakey</th>
<th>Java</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>where</strong></td>
<td>methods in class Picture</td>
<td>not defined as method</td>
</tr>
<tr>
<td><strong>arguments</strong></td>
<td>( background, newBackground )</td>
<td>( source, background, newBackground )</td>
</tr>
<tr>
<td><strong>return result?</strong></td>
<td>operate on “this” object</td>
<td>return changed source</td>
</tr>
<tr>
<td><strong>syntax</strong></td>
<td>declare variables for loops blocks in curly braces { }</td>
<td>no type declaration for loops blocks indented</td>
</tr>
</tbody>
</table>
Java on the Final

- Take Home Portion
  - Java problem: LeaderTurtle & FollowerTurtle
  - due on Dec 17, @ 2:00 PM
- In 316 McBryde (our regular room) Dec 17, 4:25-6:25
- Be sure JES works!
- open book, open computer
Java Turtle problem

Create two new kinds of turtles, LeaderTurtles and FollowerTurtles

FollowerTurtles move whenever Leaders move

detailed instructions to be posted on website on Thursday
Today

- A brief review
  - Python
  - Java (compared with Python)
- Next Steps
- Survey
- Course Evaluation
So this course was kinda cool ...

- in CS:
  - Soon: Make an appointment with CS advisors in McBryde
  - “Creative Computing Track”

- out of CS:
  - Computer Music & Multimedia Design (MUS)
  - New Media Theory (A&AH)
  - “Cyberart Studio” (CS, A&AH, ???)
CREATIVE COMPUTING

- Basic CS BS program plus:

  - CS Courses:
    - GUI Programming (new course)
    - Multimedia / Hypertext (CS 3624)
    - CyberArt or Game Design capstone (cs 4644)

  - Electives:
    - CS 4204 Computer Graphics
    - CS 4634 Design of Information
    - CS 4804 Intro to AI

- MUS 4?? Computer music & multimedia design
- Art 4804 New Media Theory
- Art 2704 Intro to 3D Animation *
- Art 3704 Topics in 3D Animation*
- Com 2054 Introduction to Film
- Com 3194 Film Production
- Com 4014 Media Effects
- Com 4034 Functions of Popular Culture
Coming Attractions

- HW 10 - oche
  - due on Thursday
  - look at Python echo recipe
    - what is different?
- Thursday (2-3 PM) in 110 McB
  - Open House
  - Learn about game design, animation, multimedia, cyberart
  - FOOD!
- Final, Wednesday 12.17
  4:25-6:25 in 316 McBryde
TODAY

- A brief review
  - Python
  - Java (compared with Python)
- Next Steps
- Survey
- Course Evaluation
THANK YOU!
You’ve been a great class. Good luck on the final.
A brief review

- Python
- Java (compared with Python)

Next Steps

Survey

Course Evaluation
COURSE OBJECTIVES

- Learn fundamental manipulations of digital media.
- Learn basic software engineering principles and programming skills.
- Simple data types, control structures, array and string data structures and algorithms, testing and debugging.