Python/JES Quick Reference
CSE Bridge Workshop Summer 2009

Variables, expressions, and output

Numbers:	integers (0, 12, 17, …), floats (0.0, 3.14, 6.02e23)
Operators:	+ - * / % (modulus or remainder) ** (exponentiation)
Strings:	"anything in double quotes" or 'anything in single quotes'
Assignment:	*variable* = *value*
Output:	print *value*

Variable names can be almost anything that begins with a letter and contains letters, digits, and underscores (_). Exception: reserved words that have special meaning to Python like if, def, and return can't be used as variable names. JES and other Python-savvy editors will help out by displaying reserved words and normal identifiers (variable names) in different colors.

Math Functions

Python's standard library includes many common math functions like sqrt(x), sin(x), cos(x), log(x), abs(x), max(list), min(list), etc., as well as the constants pi and e. To access these items in regular Python, put the statement

```
from math import *
```

at the top of your file, or type it if you are running Python interactively. Most of these functions are available in JES without an explicit import, however.

Functions

Definition:

```
def function_name ( optional_parameters ) :
    statements that make up the function body
```

The statements are not executed until the function is called.

The function can return values with a return statement (syntax: return *value*)

Loops and conditionals

Most of our loops use for, which repeats statements for each value in some collection (like all of the pixels in an image):

```
for variable in collection :
    statements to repeat
```

To repeat statements for a sequence of numbers, use range for the collection:

```
for variable in range(min, max) :
    statements to repeat
```

We can also repeat statements as long as some condition remains true (the condition is only tested once each time around the loop):

```
while condition :
    statements to repeat
```

A condition can include operations to compare numbers or other values (< > <= >= == !=) and operators to combine logical values (and or not).

Statements can be conditionally executed with if:

```
if condition :
    statements to execute if true
```

or

```
if condition :
    statements to execute if true
else:
    statements to execute if false
```

Files and Pictures

Recipe to open an image file and display it:

```
setMediaPath()                # optional, but sets default directory
file = pickAFile()
picture = makePicture(file)
show(picture)
```

Refresh the display to show changes after altering the picture in memory: repaint(*picture*)

To write a new copy of a (possibly altered) picture to a file (uses current setMediaPath() directory):

```
writePictureTo(picture, filename)      # filename is a string like “pic.jpg”
```

Picture Functions

Display a color chooser to look at red, green, blue components of colors: pickAColor()

To show a picture variable in a window where you can examine colors and pixel values and magnify the image, select PictureTool… from the JES MediaTools menu.

Get a list of all pixels in a picture (usually to use in a for loop): getPixels(*picture*). Example:

```
for pixel in getPixels(picture):
   process each pixel variable in turn
```

Get the number of rows or columns in a picture: *h* = getHeight(*picture*) *w* = getWidth(*picture*)

Get the pixel at a particular location in an image: *pixel* = getPixel(*picture*, *xpos*, *ypos*)

Given a pixel, find out its coordinates: *x* = getX(*pixel*) *y* = getY(*pixel*)

Get or set a pixel’s colors: r = getRed(*pixel*) setRed(*pixel*, *value*) . Value is 0-255. Similarly for green, blue.

Get or set a pixel’s colors with a color triple: *color* = getColor(*pixel*) setColor(*pixel*, *color*)

Create a color from red, green, blue values: *color* = makeColor(*red*, *green*, *blue*)

Create a blank picture: *pic* = makeEmptyPicture(*width*, *height*)

Look in the JES function menu for these and other functions; select a function name for help.