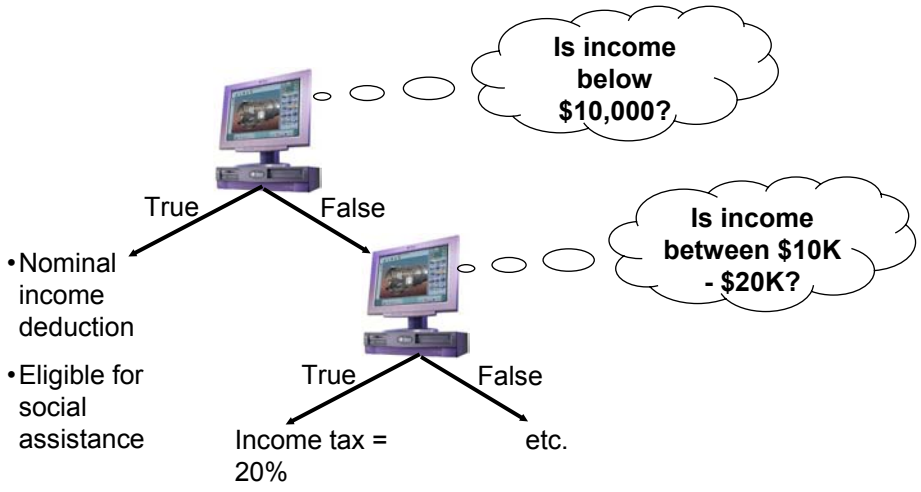
Programming: Part II

In this section of notes you will learn more advanced programming concepts such as branching and repetition as well as how to work with multimedia files (images and sounds).

Why Is Branching/Decision Making Needed?

- When alternative courses of action are possible and each action may result in a different result.
- Branching/decision making can be used in a program to deal with alternative.

High Level View Of Decision Making For The Computer



Decision-Making In Python

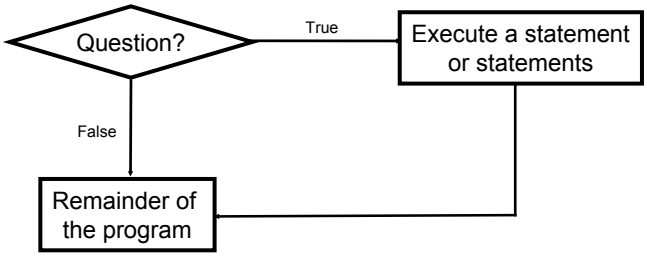
Decisions are questions with answers that are either true or false (Boolean) e.g., Is it true that the variable 'num' is positive?

The program branches one way or another depending upon the answer to the question (the result of the Boolean expression).

Decision making/branching constructs (mechanisms) in Python:

- If
- If-else
- If-elif-else

Decision Making With An 'If'



The 'If' Construct

Decision making: checking if a condition is true (in which case something should be done).

Format:

(General format)

if (*Boolean expression*):
    body

(Specific structure)

if (*operand relational operator operand*):
    body

Boolean expression

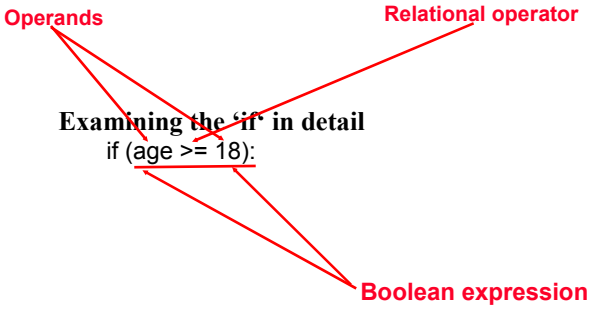
Note: Indenting the body is mandatory!

A Boolean expression is a question that evaluates to a Boolean value (answers true or false)

The 'If' Construct (2)

Example: Available online and is called "decision1.py"

```
def ifExample ():
    age = input ("Enter your age: ")
    if (age >= 18):
        print "You are an adult"
```



Allowable Operands For Boolean Expressions

If (**operand** relational operator **operand**) then:

Some operands

- integer
- real numbers
- String

Make sure that you are comparing operands of the same type!

Allowable Relational Operators For Boolean Expressions

If (operand **relational operator** operand) then

Python operator	Mathematical equivalent	Meaning	Example
<	<	Less than	5 < 3
>	>	Greater than	5 > 3
==	=	Equal to	5 == 3
<=	≤	Less than or equal to	5 <= 5
>=	≥	Greater than or equal to	5 >= 4
<>	≠	Not equal to	5 <> 5
OR			
!=			5 != 5

If (Simple Body)

Body of the if consists of a single statement

Format:

```
if (Boolean expression):
    s1
s2
```

Body

Example:

```
if (num == 1):
    print "Body of the if"
print "After body"
```

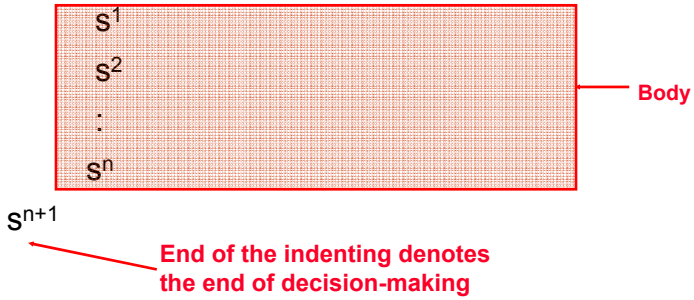
Indenting used to indicate what statement is the body

If (Compound Body)

Body of the if consists of multiple statements

Format:

if (Boolean expression):



If (Compound Body(2))

Example: Available online and is called "tax.py"

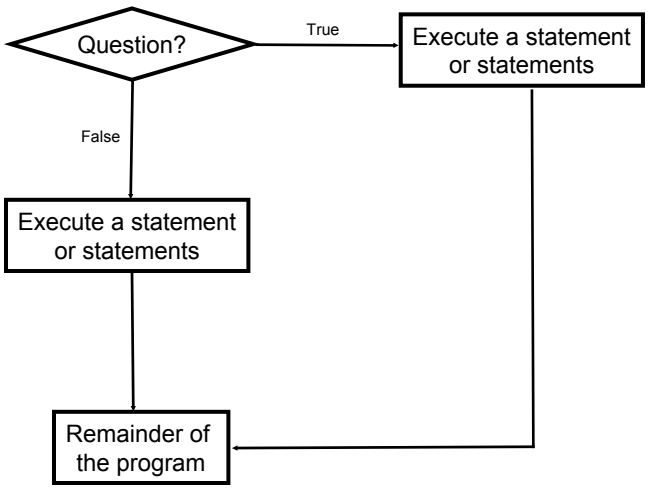
```
def tax ():
    taxCredit = 0
    taxRate = 0.2
    income = input ("Enter your pre-tax income: ")
    if (income < 10000):
        print "Eligible for social assistance"
        taxCredit = 100
    tax = (income * taxRate) – taxCredit
    print tax
```

Decision Making With An 'If": Summary

Used when a question (Boolean expression) evaluates only to a true or false value (Boolean):

- If the question evaluates to true then the program reacts differently. It will execute a body after which it proceeds to execute the remainder of the program (which follows the if construct).
- If the question evaluates to false then the program doesn't react different. It just executes the remainder of the program (which follows the if construct).

Decision Making With An 'If-Else'



The If-Else Construct

Decision making: checking if a condition is true (in which case something should be done) but also reacting if the condition is not true (false).

Format:

```
if (operand  relational operator  operand):
    body of 'if'
else:
    body of 'else'
additional statements
```


If-Else Construct (2)

Example: Available online and is called "decision2.py"

```
def decision2 ():
  age = input ("Enter your age: ")
  if (age >= 18):
      print "Adult"
  else:
      print "Not an adult"
  print "Tell me more about yourself"
```

If-Else (Compound Body(2))

Example: Available online and is called "tax2.py"

```
def tax2 ():
    income = input ("Enter your pre-tax income: ")
    taxCredit = 0
    if (income < 10000):
        print "Eligible for social assistance"
        taxCredit = 100
        taxRate = 0.1
    else:
        print "Not eligible for social assistance"
        taxRate = 0.2
    tax = (income * taxRate) - taxCredit
    print tax
```


Quick Summary: If Vs. If-Else

If:

- Evaluate a Boolean expression (ask a question)
- If the expression evaluates to true then execute the 'body' of the if.
- No additional action is taken when the expression evaluates to false.
- Use when your program evaluates a Boolean expression and statements are to execute only when the expression evaluates to true.

If-Else:

- Evaluate a Boolean expression (ask a question)
- If the expression evaluates to true then execute the 'body' of the if.
- If the expression evaluates to false then execute the 'body' of the else.
- Use when your program evaluates a Boolean expression and different statements will execute if the expression evaluates to true than if the expression evaluates to false.

Decision-Making With Multiple Expressions

The multiple expressions are joined with a logical operator (AND, OR)

Format:

```
if (Boolean expression) logical operator (Boolean expression):
   body
```

Example:

```
if (x > 0) and (y > 0):
    print "X is positive, Y is positive"
```


Decision-Making With Multiple Expressions (2)

Commonly used logical operators in Python

- or
- and
- not

Forming Compound Boolean Expressions With The "OR" Operator

Format:

```
if (Boolean expression) or (Boolean expression):
    body
```

Example:

```
if (gpa > 3.7) or (yearsJobExperience > 5):
    print "You are hired"
```


Forming Compound Boolean Expressions With The "AND" Operator

Format:

```
if (Boolean expression) and (Boolean expression):
    body
```

Example:

```
if (yearsOnJob <= 2) and (salary > 50000):
    print "You are fired"
```

Forming Compound Boolean Expressions With The "NOT" Operator

Format:

```
if not (Boolean expression):
   body
```

Examples:

```
if not (num > 0)
     print ("Num is zero or less")

if not ((response == 'y') or (response == 'Y')):
       print "Invalid response, enter only 'y' or 'Y'"
```


Quick Summary: Using Multiple Expressions

Use multiple expressions when multiple questions must be asked and the result of each expression may have an effect on the other expressions:

AND:

- All Boolean expressions must evaluate to true before the entire expression is true.
- If any expression is false then whole expression evaluates to false

OR:

- If any Boolean expression evaluates to true then the entire expression evaluates to true.
- All Boolean expressions must evaluate to false before the entire expression is false.

Decision-Making With Multiple Alternatives

if

Checks a condition and executes the body of code if the condition is true

if-else

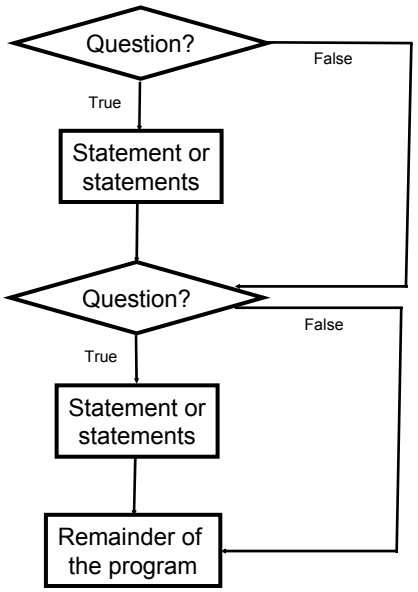
Checks a condition and executes one body of code if the condition is true and another body if the condition is false

Approaches for multiple (two or more) alternatives

Multiple if's

if-elif-else

Decision Making With Multiple If's



Multiple If's: Non-Exclusive Conditions

Any, all or none of the conditions may be true (independent)

Format:

```
if (Boolean expression 1):
    body 1
if (Boolean expression 2):
    body 2
        :
statements after the conditions
```


Multiple If's: Non-Exclusive Conditions (Example)

Example:

```
if (num1 > 0):
    print “num1 is positive”
if (num2 > 0):
    print “num2 is positive”
if (num3 > 0):
    print “num3 is positive”
```

Multiple If's: Mutually Exclusive Conditions

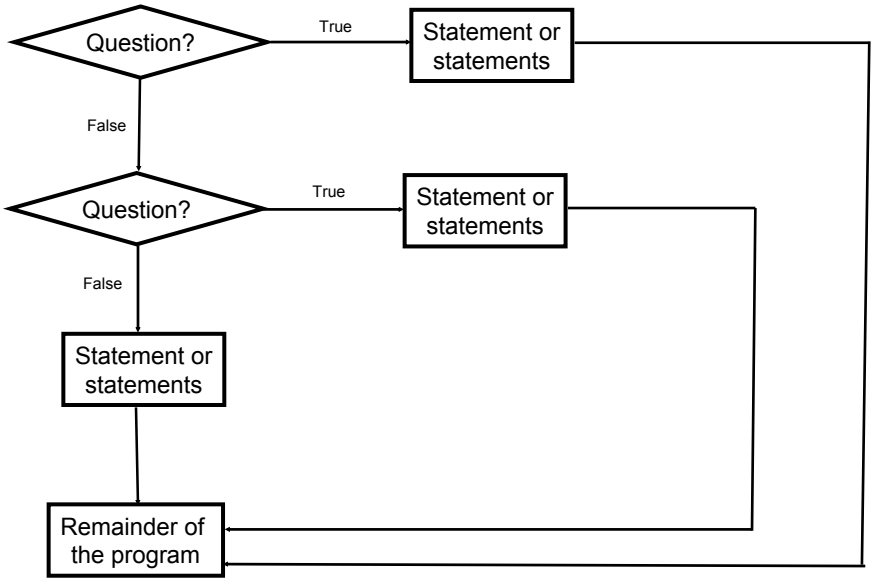
At most *only one* of many conditions can be true
Can be implemented through multiple if's

Inefficient combination!

Example: The example is available online and is called "grades.py"

```
def grades ():
    gpa = input ("Enter the grade point")
    if (gpa == 4):
        letter = 'A'
    if (gpa == 3):
        letter = 'B'
    if (gpa == 2):
        letter = 'C'
    if (gpa == 1):
        letter = 'D'
    if (gpa == 0):
        letter = 'F'
    print gpa, letter
```


Decision Making With If-Elif-Else



Multiple If-Elif-Else: Mutually Exclusive Conditions

Format:

```
if (Boolean expression 1):
    body 1
elif (Boolean expression 2):
    body 2
        :
else
     body n
statements after the conditions
```


Multiple If, Else-If's: Mutually Exclusive Conditions (Example)

Example: The complete example is available online and is called "grades2.py"

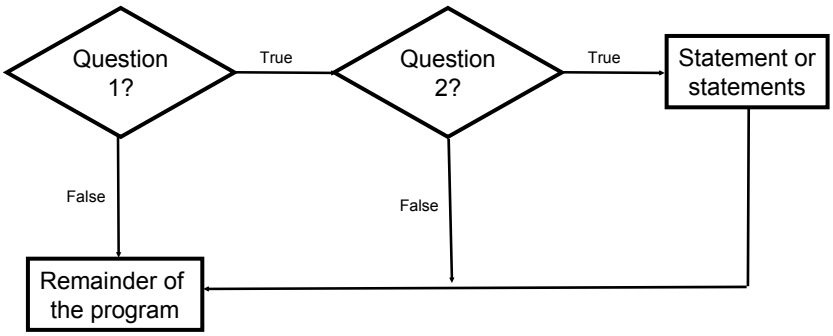
```
if (gpa == 4):
      letter = 'A'
   elif (gpa == 3):
      letter = 'B'

   elif (gpa == 2):
      letter = 'C';

   elif (gpa == 1):
      letter = 'D'

   elif (gpa == 0):
       letter = 'F'

   else:
       print "GPA must be one of '4', '3', '2', '1' or '1'"
       letter = 'NA'
   print gpa, letter
```

This approach is more efficient when at most only one condition can be true.

The body of the else executes only when all the Boolean expressions are false. (Useful for error checking).

Nested Decision Making

- Decision making is dependent.
- The first decision must evaluate to true before successive decisions are even considered for evaluation.



Nested Decision Making

One decision is made inside another.

Outer decisions must evaluate to true before inner decisions are even considered for evaluation.

Format:

```
if (Boolean expression):
    if (Boolean expression):
        inner body
```

Outer body

Inner body

Nested Decision Making (2)

Example:

```
if (income < 10000):
    if (citizen == 'y'):
        print "This person can receive social assistance“
        taxCredit = 100
tax = (income * TAX_RATE) - taxCredit
```


Recap: What Decision Making Constructs Are Available In Python/When To Use Them

Construct	When To Use
If	Evaluate a Boolean expression and execute some code (body) if it's true
If-else	Evaluate a Boolean expression and execute some code (first body) if it's true, execute alternate code (second body) if it's false
Multiple if's	Multiple Boolean expressions need to be evaluated with the answer for each expression being independent of the answers for the others (non-exclusive). Separate code (bodies) can be executed for each expression.
If-elif-else	Multiple Boolean expressions need to be evaluated but zero or at most only one of them can be true (mutually exclusive). Zero bodies (if) or exactly one body will execute. Also it allows for a separate body (else) to execute when all the if-elif Boolean expressions are false.

Recap: When To Use Compound And Nested Decision Making Constructs (2)

Construct	When To Use
Compound decision making	More than one Boolean expression must be evaluated before some code (body) can execute.
Nested decision making	The outer Boolean expression must be true before the inner expression will even be evaluated.

The Need For Repetition (Loops)

What if the program is to be executed multiple times but running the program for the first time is treated different from running the program the successive times:

- e.g. 1, A game that tracks your score each time that it's played.
- e.g. 2., A program that displays an introduction to the program only the first time that it's run.

It may also be necessary to repeat only *a part* of a program.

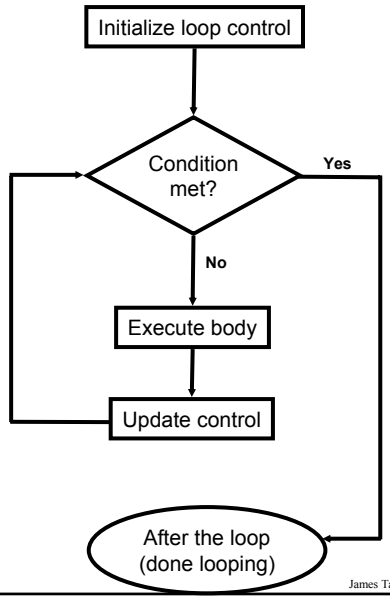
- e.g., the program keeps prompting the user for information until the information is entered in the correct format and/or the input falls within an acceptable range.

What Determines If A Loop Repeats

- Depending upon the type of loop it will repeat as long as some condition has been met (e.g., the player still wants to play the game) or the loop continues as long a condition has not been met (e.g., a loop that continues prompting the user for input until it's entered in the correct format). This condition for execution is referred to as "the stopping condition")
- Often the value of variable (called a "loop control") is what determines whether a loop repeats.

Loops: Basic Structure

1. Initialize loop control
2. Check if the stopping condition has been met
   a. If it's been met then the loop ends
   b. If it hasn't been met then proceed to the next step
3. Execute the body of the loop (the part to be repeated)
4. Update the loop control
5. Go to step 2



Types Of Loops In Python

For-loop:

- Typically used as a count controlled loop (count a fixed number of times) or to step through a sequence (items in a list, pixels in a picture).

While-loop:

- Used for any case where repetition is needed (usually can do what a for-loop does and more – although in some cases using a for loop may be simpler).

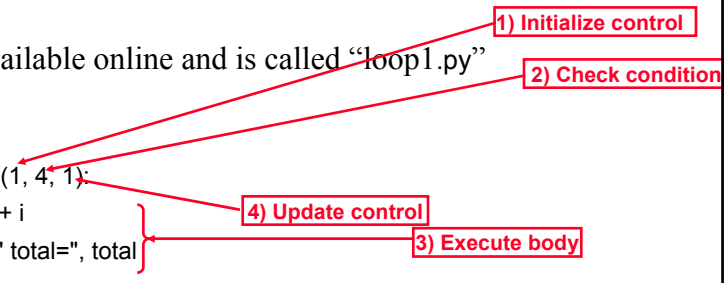
The For Loop

Format:

for <*variable*> in <*something that can be stepped through*>:
    body

Example: Available online and is called "loop1.py"

```
def loop1 ():
    total = 0
    for i in range (1, 4, 1):
        total = total + i
        print "i=", i, " total=", total
    print "Done!"
```



The While Loop

This type of loop can be used if it's not in advance how many times that the loop will repeat (most powerful type of loop).

Format:

(Simple)

```
while (Boolean expression):
    body
```

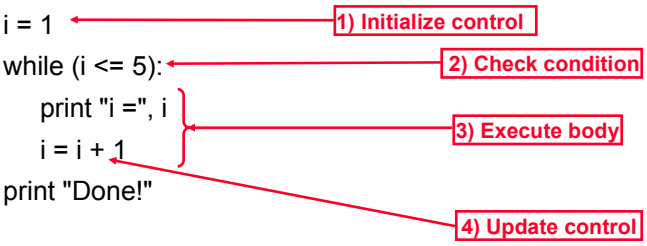
(Compound)

```
while (Boolean expression) Boolean operator (Boolean expression):
    body
```


The While Loop (2)

Example: Available online and is called "loop2.py"

```
def loop2 ():
    i = 1
    while (i <= 5):
        print "i =", i
        i = i + 1
    print "Done!"
```



Loop Increments Need Not Be Limited To One

While

```
i = 0
while (i <= 100):
    print "i =", i
    i = i + 5
print "Done!"
```

For

```
for i in range (105, 0, -5):
    print "i =", i
print "Done!"
```


Solving A Problem Using Loops

Problem: Write a program that will execute a game:

- The program will randomly generate a number between one and ten.
- The player will be prompted to enter their guess.
- The program will continue the game until the player indicates that they no longer want to continue.

Solution: The full program can be found online and is called "guessingGame.py"

Guessing Game

```
guess = 0
answer = 0
choice = "Y"
while choice not in ("q", "Q"):
    answer = random.randrange (10) + 1
    guess = input ("Enter your guess: ")
    if (guess == answer):
        print "You guessed correctly!"
    else:
        print "You guessed incorrectly"
    print "Number was", answer, ", your guess was", guess
    print "Play again?  Enter 'q' to quit, anything else to play again"
    print "Choice:",
    choice = raw_input()
    print ""
print "Exiting game"
```


Nested Loops

One loop executes inside of another loop(s).

Example structure:

Outer loop (runs n times)
    Inner loop (runs m times)
        Body of inner loop (runs n x m times)

Example program: It can be found online and is called "nested.py"

```
for i in range (1, 3, 1):
    for j in range (1, 4, 1):
        print "i = ", i, " j = ", j
print "Done!"
```

Picture/Image-Related Functions In JES

For more details look under the help menu under "Picture functions".

- addLine (picture, startX, startY, endX, endY)
- addRect (picture, startX, startY, width, height)
- addRectFilled (picture, startX, startY, width, height, color)
- addText (picture, xpos, ypos, text, color):

Explanation of the function values

•picture: the name of the picture that you want to edit

•startX: the starting pixel coordinate on the x axis

•startY: the starting pixel coordinate on the y axis

•endX: the ending pixel coordinate on the x axis

•endY: the ending pixel coordinate on the y axis

•width: the width of the rectangle in pixels

•weight: the height of the rectangle in pixels

•color: the color to fill the rectangle with (red, green, blue etc.) or the color of the text (specifying the color is optional in the case of text, if the color is not set then the default black color will be used)

James Tam

Example Of Editing A Picture

Available online and is called "picture1.py"

```
def picture1():
  lion = makePicture ("lion.jpg")
  addLine (lion,0,0,100,100)
  addRectFilled (lion,100,100,100,200,red)
  addText (lion,200,300,"Lion dance for the Lions Club")
  show (lion)
```

Important: Typically you want to show a picture only after you have finished all your edits!

James Tam

The 24 Bit Color Model

Each pixel's color is specified with 24 bits:

- 8 bits (256 combinations) for the red component
- 8 bits (256 combinations) for the blue component
- 8 bits (256 combinations) for the green component

In JES the color value is an integer ranging from 0 – 255.

- Smaller numbers result in darker pixels.
- Larger numbers result in lighter pixels.

JES' Pixel-Related Functions

Get functions: find the color level of a particular sub-pixel

- getRed: returns the red level (0 – 255) of a pixel
- getBlue: returns the blue level (0 – 255) of a pixel
- getGreen: returns the green level (0 – 255) of a pixel

Set functions: change the color of a particular sub-pixel

- setRed: change the red level of a pixel (to a value from 0 – 255)
- setBlue: change the blue level of a pixel (to a value from 0 – 255)
- setGreen: change the green level of a pixel (to a value from 0 – 255)

Example: Seeing The Color Values Of A Picture

Available online and is called "picture2.py". It also requires that you download and save the picture "smallLion.jpg" into the folder that you run JES from.

```
def picture2 ():

  aPicture = makePicture ("smallLion.jpg")
  show (aPicture)
  allPixels = getPixels (aPicture)
  path = raw_input ("Enter the location and name for the modified
   picture")

  for pixel in allPixels:
    red = getRed (pixel)
    blue = getBlue (pixel)
    green = getGreen (pixel)
    print "RBG:", red, blue, green
```


Example: Changing The Color Values Of A Picture

Available online and is called "lighten.py". It also requires that you download and save the picture "mediumLion.jpg" into the folder that you run JES from.

```
def lighten ():
  path = raw_input ("Enter the location and name of the file to save the
   changes to (make sure it ends with the proper suffix: ")
  aPicture = makePicture ("mediumLion.jpg")
  show (aPicture)
  allPixels = getPixels (aPicture)
```

Show the original picture loaded from file.

Example: Changing The Color Values Of A Picture (2)

```
for pixel in allPixels:
  red = getRed (pixel)
  blue = getBlue (pixel)
  green = getGreen (pixel)
  if ((red + 50) <= 255):
    setRed (pixel, (red+50))
  if ((blue + 50) <= 255):
    setBlue (pixel, (blue+50))
  if ((green + 50) <= 255):
    setGreen (pixel, (green+50))
repaint (aPicture)
writePictureTo (aPicture, path)
```

Step through each pixel in the image

Set each pixel's sub-pixel to a lighter value (larger number).

Show the original picture after it has been manipulated.

Save the modified image to the location and filename specified by the user.

Loading Media Files Into JES

JES can load, play and modify multimedia files:

- Sounds in the form of ".wav" files
- Videos in the form of ".avi" or ".mov" (usually associated with the Quicktime player). You must have Quicktime for Java installed on your computer.

Example: Playing A Sound In JES

This program is available online and is called "soundExample.py". This program should be able to play the sounds using any "wav" file that isn't too large (~less than 100 KB) but I included a sound file that I know will work so you can immediately start using this program.

```
def soundExample ():
  path = pickAFile ()
  print "You want to play the sound ", path
  aSound = makeSound (path)
  play (aSound)
```


You Should Now Know

What are the three decision making constructs available in Python:

- If
- If-else
- If-elif-else
- How does each one work
- When should each one be used

How to evaluate and use decision making constructs:

- Tracing the execution of simple decision making constructs
- How to evaluate nested and compound decision making constructs and when to use them

When and why are loops used in computer programs

How to properly write the code for a loop in a program

What are nested loops and how do you trace their execution

You Should Now Know (2)

How to work with graphics in JES

- How images are represented using the 24 bit color model
- How to use the functions for working with pictures and other graphical objects in JES

How to load and play sound files in JES