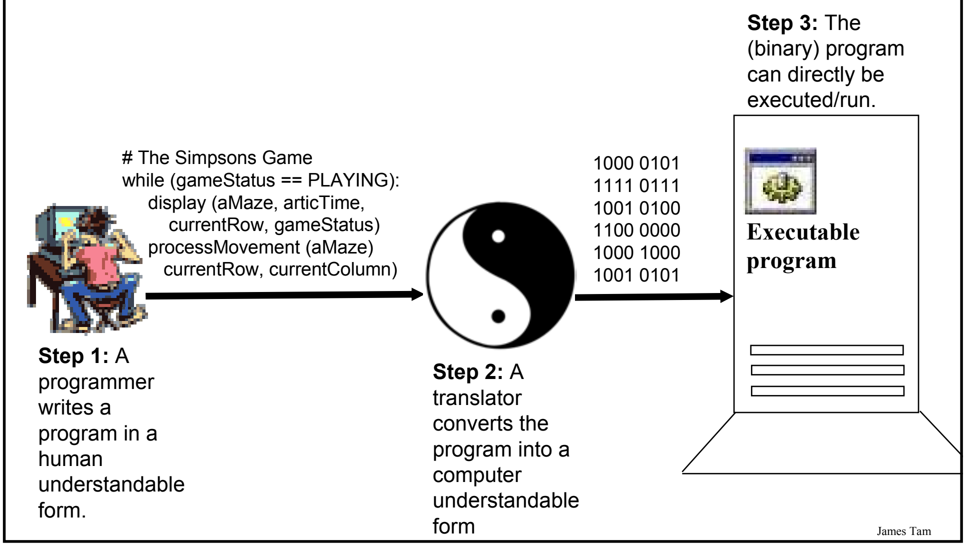
Programming: Part I

In this section of notes you will learn about how to write simple programs using JES.

Translators

The (greatly simplified) process of writing a computer program.



Step 3: The (binary) program can directly be executed/run.

```
# The Simpsons Game
while (gameStatus == PLAYING):
    display (aMaze, articTime,
        currentRow, gameStatus)
    processMovement (aMaze)
       currentRow, currentColumn)
```

```
1000 0101
1111 0111
1001 0100
1100 0000
1000 1000
1001 0101
```

Step 1: A programmer writes a program in a human understandable form.

Step 2: A translator converts the program into a computer understandable form

What Your Will Be Writing/Translating Your Programs With In This Class

Writing programs this semester:

- Programming language: Python (actually it's a modified version of Python called JPython or the JES version of the Python programming language)
  - Starting tutorial: http://docs.python.org/tutorial/
  - Online documentation: http://www.python.org/doc/)
  - My old CPSC 217 notes: http://pages.cpsc.ucalgary.ca/~tamj/2008/217W/index.html
- Software to write/translate your Python programs: JES
  - Quick introduction: http://www.cs.ucr.edu/~titus/JesIntro.pdf

Where You Can Run JES

Lab computers:

- JES is installed and running on all the computers in both 203 labs

Installing on your own computer (Windows, Mac, Linux)[1]:

- http://coweb.cc.gatech.edu/mediaComp-plan/94

OR

- http://www.cc.gatech.edu/classes/AY2006/cs1315_summer/software.html

[1] Java is needed run JES so if you don't have it installed then you should download it as well

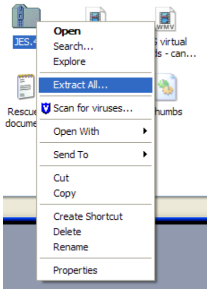
Resources

My completed example programs can be found on the course web page under the following link:

http://pages.cpsc.ucalgary.ca/~tamj/203/topics/programming.html:

Getting Started At Home[1]

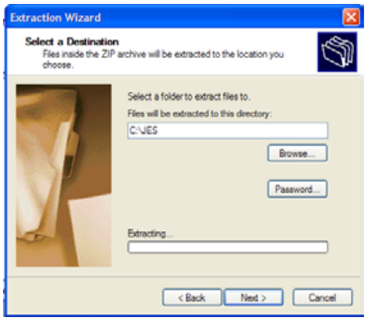
- Step 1: Download the version appropriate to your computer (Windows, Mac, Linux).
- Uncompress the compressed (zip format) file using the appropriate program (in Windows it's built into the operating system). Right click on the downloaded file:



[1] Note: It is NOT required for this course that you install JES at home. No warranties or guarantees of service are provided for this program (i.e., we aren't responsible if you inadvertently damage your computer during the installation process).

Getting Started At Home (2)

Pick a location to extract the files to that you will remember:



Files inside the ZIP archive will be extracted to the location you choose.

Select a folder to extract files to.

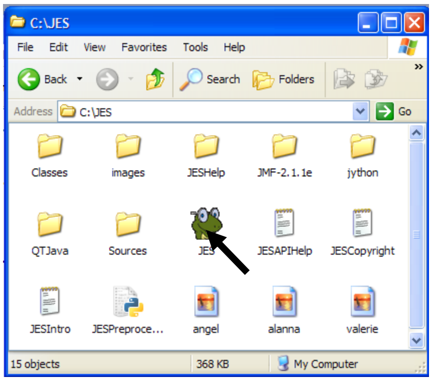
Files will be extracted to this directory:

C:\JES

Note: to keep it simple any data files (e.g., images) that you need for your programs should be stored in this folder/directory.

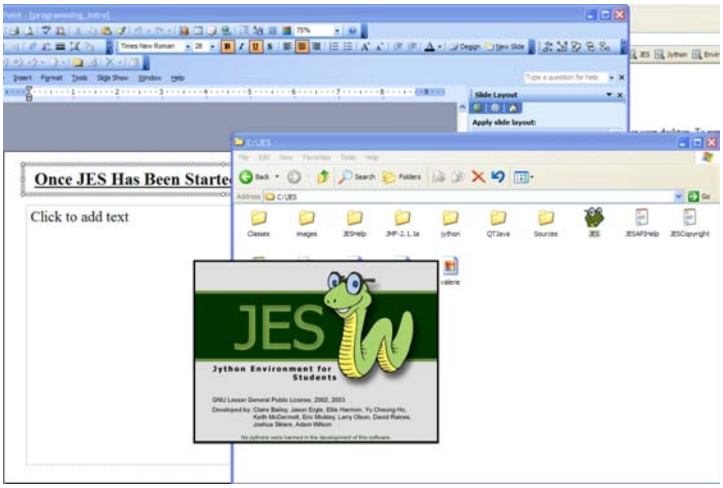
Getting Started At Home (3)

To start the program click on the 'JES' icon:

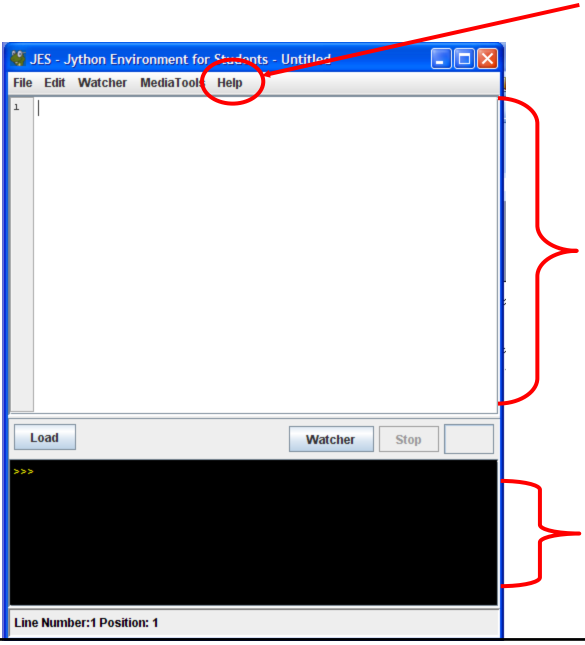


Once JES Has Been Started (Home Or In The Lab)

The splash screen will first load (it may stay on a few seconds even with a fast computer)


Once JES Has Been Starte
Click to add text
C:\JES
Classes
images
JESHelp
JMF-2.1.1e
Jython
QTJava
Sources
JES
JESAPIHelp
JESCopyright
JES
Jython Environment for Students
GNU Lesser General Public License, 2002, 2003
Developed by: Claire Bailey, Jason Ergle, Ellie Harmon, Yu Cheung Ho, Keith McDermott, Eric Mickley, Larry Olson, David Raines, Joshua Sklare, Adam Wilson
No pythons were harmed in the development of this software

JES

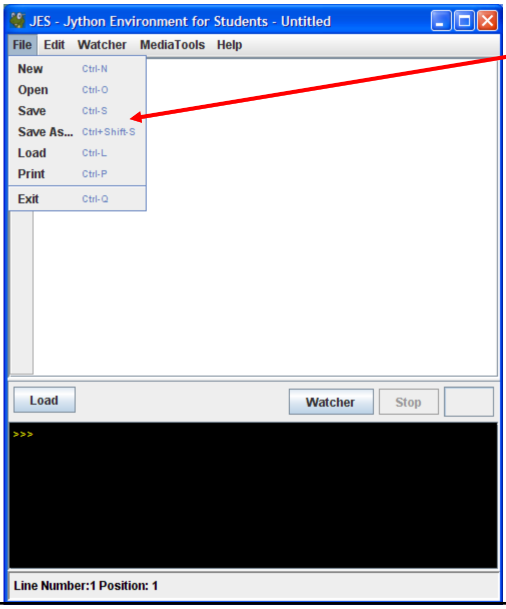

JES - Jython Environment for Students - Untitled
File
Edit
Watcher
MediaTools
Help
1
Load
Watcher
Stop
>>>
Line Number:1 Position: 1

Built in help for using JES and for some programming concepts in JPython

Type in your programs here

Type in your commands (e.g., run the program that you just entered) here

File Operations



Similar to a text editor (e.g., notepad) and are used in conjunction with the creation of your programs (load, save, print etc.)

The Basic Structure Of A Program In JES

Format:

def *program name* ():
   Body of the program[1]

Indent the body using three spaces (or at least make it consistent throughout).

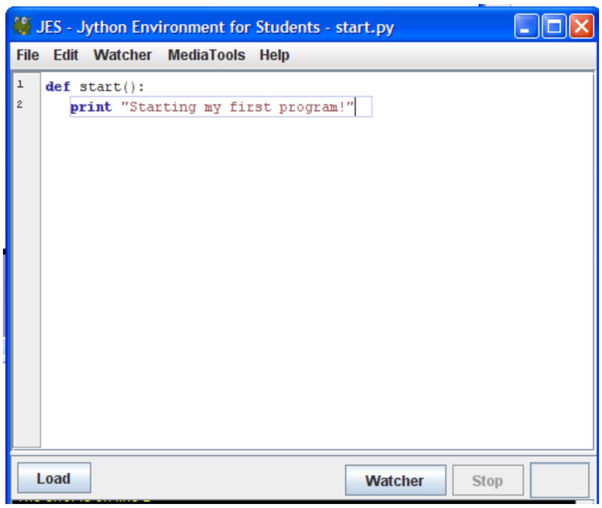
Example:

```
def start ():
   print "Starting my first program!"
```

[1] This is the part that actually completes actions such as: displaying messages onscreen, loading a picture from file, performing a calculation etc.

Creating And Running Your First Program

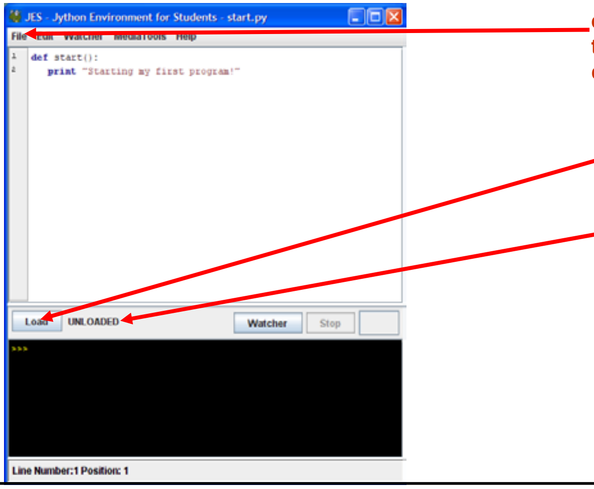
Step 1: Type in your program in the editing area:


JES - Jython Environment for Students - start.py
File Edit Watcher MediaTools Help
Load
Watcher
Stop

```
def start():
   print "Starting my first program!"
```

Creating And Running Your First Program (2)

Step 2: Load your program so it can be translated into binary. (Currently your program has been loaded into the editor but not loaded into the JES translator).


JES - Jython Environment for Students - start.py
File Edit Watcher MediaTools Help
Load
UNLOADED
Watcher
Stop
>>>
Line Number:1 Position: 1

```
def start():
   print "Starting my first program!"
```

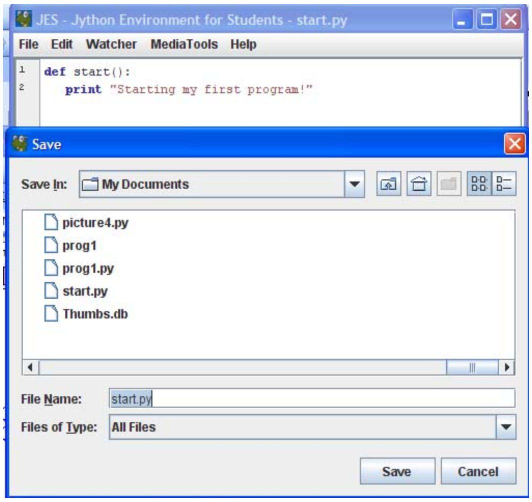
The file menu affects the editor and not the translator (e.g., open doesn't load the program)

Load your program here

Program hasn't been 'loaded' yet

Creating And Running Your First Program (2)

Step 3: When you run your program JES will ask if you want to save it. Save it with a file name that ends in dot-py (e.g., start.py)



```
def start():
    print "Starting my first program!"
```

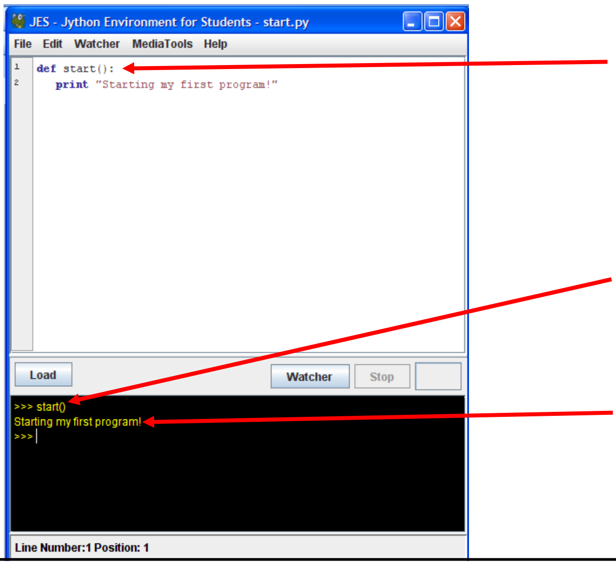
picture4.py
prog1
prog1.py
start.py
Thumbs.db

File Name: start.py

Files of Type: All Files

Creating And Running Your First Program

Step 4: Run your program in the command area


JES - Jython Environment for Students - start.py
File
Edit
Watcher
MediaTools
Help
Load
Watcher
Stop
Line Number:1 Position: 1

```
def start():
    print "Starting my first program!"
```

```
>>> start()
Starting my first program!
>>> 
```

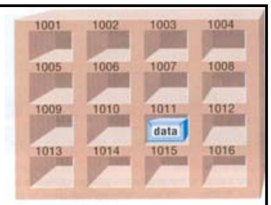
IMPORTANT: the name that you type in to run the program must match the program name specified here

Running program called ‘start’

The effect of running your program is displayed immediately after you run it.

Storing Information

- When writing a program there will sometimes be a need to store information e.g., to perform a calculation.
- Information is stored in a computer program by using a variable.



Variables

Set aside a location in memory

Used to store information (temporary)

- This location can store one ‘piece’ of information
- *At most* the information will be accessible as long as the program runs

Some of the types of information which can be stored in variables:

- Integer (num = 10)
- Real numbers (num = 10.5)
- Strings (message = “Not happy! >-<”)

Picture from Computers in your future by Pfaffenberger B

Arithmetic Operators

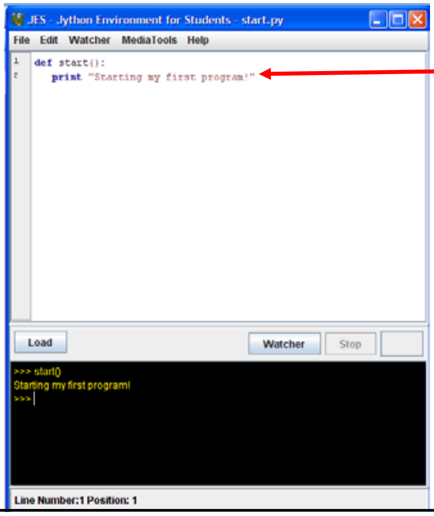
Operator	Description	Example
=	Assignment	num = 7
+	Addition	num = 2 + 2
-	Subtraction	num = 6 - 4
*	Multiplication	num = 5 * 4
/	Division	num = 25 / 5
%	Modulo	num = 8 % 3
**	Exponent	num = 9 ** 2

Displaying Output

- When there's a need to display information onscreen:
  - Status messages or instructions
  - Contents of variables

Displaying String Output

- Displays information or messages in the command area of JES.
- Example:


JES - Jython Environment for Students - start.py
File Edit Watcher MediaTools Help
Load
Watcher
Stop
Line Number:1 Position: 1

```
def start():
    print "Starting my first program!"
```

```
>>> start()
Starting my first program!
>>>
```

Print:

- A command to display information
- In this case, everything between the quotes will appear in the command area (string = series of characters).

Displaying String Output

Format:

print "*<string of characters>*"

Example:

```
def outputExample ():
    print "Please don't display this message!"
```

Displaying The Contents Of Variables

Format:

print <*name of variable*>

Example:

```
def example1 ():
  num = 10
  print num
```

Displaying Mixed Output

Strings and the contents of variables can be intermixed with a single print statement.

Format:

print "<*string*>", <*variable name*>...

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

```
def profit ():
   income = 2000
   expenses = 1500
   profit = income - expenses
   print "Income: ", income, " Expenses: ", expenses, " Profit: ", profit
```

Working With Picture Variables

- One of the strengths of JES is the ease at which multimedia files can be incorporated in a computer program.
- Example: Available online and is called “picture1.py”, requires that you also download and save the image called “lion.jpg” to the folder where you are running JES from).

```
def picture1():
   picture = makePicture ("lion.jpg")
   show (picture)
```

Getting Input

- In JES it can be done as the program runs.
- Example: How to specify the name of the images as the program runs. (Available online and is called “picture2.py”):

```
def picture2 (file1,file2):
   picture1 = makePicture(file1)
   show(picture1)

   picture2 = makePicture(file2)
   show(picture2)
```

- To run this program you must enter the name of two images as you run the program in the command area
  - E.g., picture2("angel.jpg","valerie.jpg")

Getting A File Dialog Box

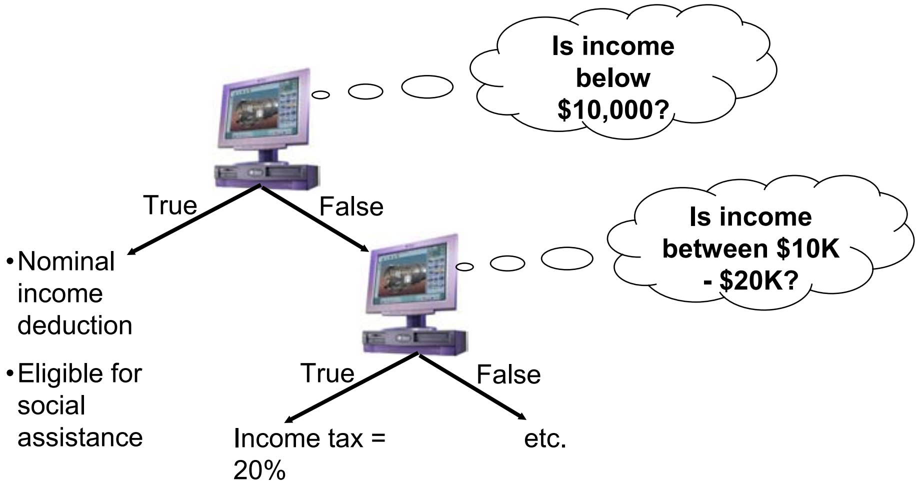
•Example: Available online and is called “picture3.py”

```
def picture3():
    filename = pickAFile()
    myPicture = makePicture (filename)
    show (myPicture)
```

What To Do If A Program Needs To Choose Among Alternatives?

- Employ branching/decision-making
- Each alternative involves asking a true/false (Boolean) question
- The answer to the question determines how the program reacts

Example Of The Need For Branching



Is income below $10,000?

True

False

- Nominal income deduction
- Eligible for social assistance

Is income between $10K - $20K?

True

False

Income tax = 20%

etc.

Some Types Of Branching Mechanisms In JES

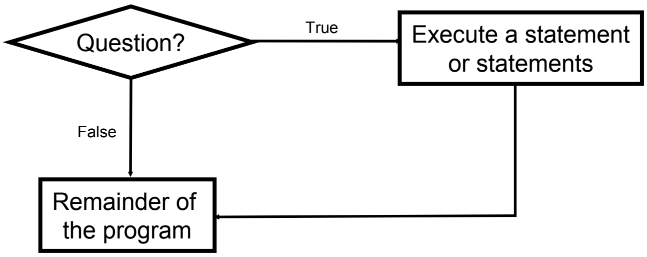
- If
- If-else
- Compound decision making
- Nested decision making

If

To be used when a true/false question (Boolean expression) is asked and:

- The program reacts one way if the question evaluates to true
- Example: If person is a senior citizen then give a 25% discount.

Decision Making With An 'If'



Question?

True

Execute a statement or statements

False

Remainder of the program

The ‘If’

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

Format:

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

Note: Indenting the body is mandatory!

The ‘If’ (2)

Example: Available online and is called “ifExample1.py”

```
def ifExample1 (age):
    if (age >= 18):
        print "Adult"
    print "Tell me more"
```

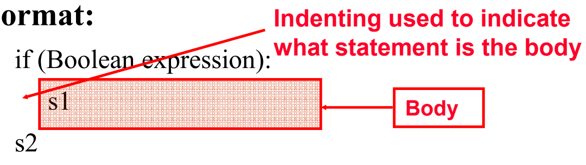
Types Of Comparisons

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
```

Indenting used to indicate what statement is the body

Body

Example:

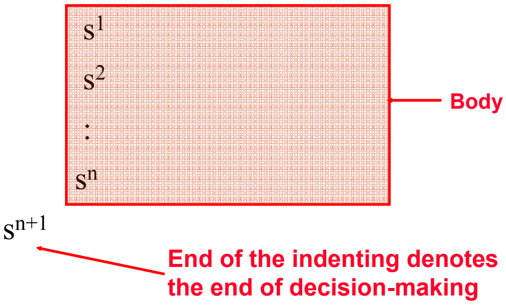
```
if (num == 1):
    print "Body of the if"
print "After body"
```

If (Compound Body)

Body of the if consists of multiple statements

Format:

if (Boolean expression):



s^1

s^2

:

s^n

Body

s^{n+1}

End of the indenting denotes the end of decision-making

If (Compound Body(2))

Example: Available online and is called “ifExample2.py”

```
def ifExample2 (income):
    taxCredit = 0
    taxRate = 0.2;
    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):

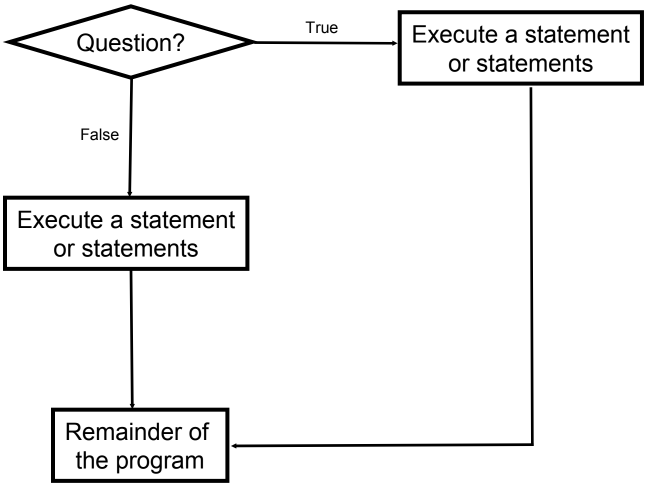
- If the question evaluates to true then the program reacts differently. It will execute a body after which it proceeds to execute the remainder of the program (which follows the if construct).
- If the question evaluates to false then the program doesn't react different. It just executes the remainder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

If-Else

To be used when a true/false question (Boolean expression) is asked and:

- The program reacts one way if the question evaluates to true
- The program reacts another way if the question evaluates to false
- Example: If income is greater than $10,000 then calculate taxes owed, otherwise calculate subsidy to be given back to the person.

Decision Making With An ‘If-Else’


Question?
True
Execute a statement or statements
False
Execute a statement or statements
Remainder of the program

The If-Else

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 (Boolean expression):
    body of 'if'
else:
    body of 'else'
additional statements
```

If-Else Construct (2)

Example: Available online and is called “ifExample3.py”

```
def ifExample3 (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 “ifExample4.py”

```
def ifExample4 (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 code will be executed 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 code will execute if the expression evaluates to true than if the expression evaluates to false.

Decision-Making With Multiple Expressions

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
```

Example:

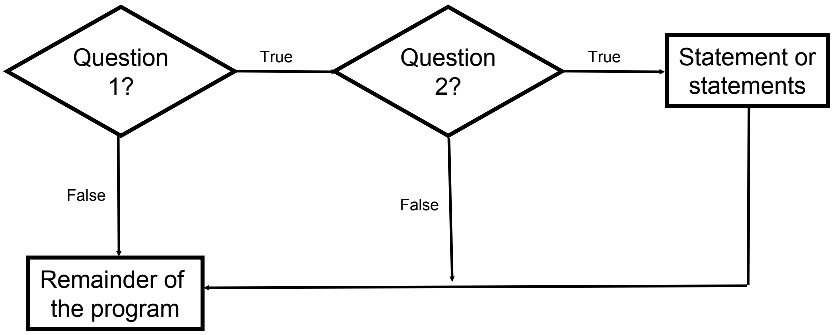
```
if not (x == 0):
    print “X is anything but zero”
```

Nested Decision Making

- Used when there's a need to qualify questions.
- One or more questions are asked only if another question evaluates to true.
- Example: Looking for people to hire
  - Question 1: Did the candidate complete an advanced graduate degree (masters or doctoral degree)
    - (The following questions are asked only if the person does have a graduate degree)
    - Question 1A: Is the grade point 3.7 or above.
    - Question 1B: Does the person have 5 or more years of work experience.
- This is referred to as nested decision making because some questions are dependent on the answer to other questions (dependent questions are 'nested' inside another question).

Nested Decision Making

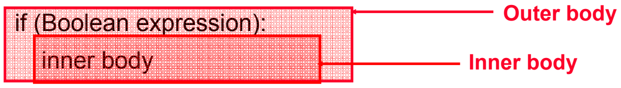
- Decision making is dependent.
- The first decision must evaluate to true before successive decisions are even considered for evaluation.



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: Available online and is called "ifExample5.py"

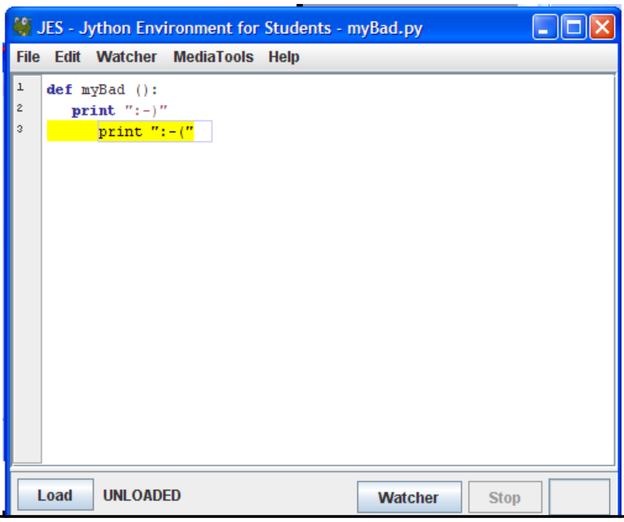
```
def ifExample5 (gradDegree, gpa, experience):
   status = "Not hired"
   if (gradDegree == 'y'):
      if (gpa >= 3.7):
         status = "Hired"
      if (experience >= 5):
         status = "Hired"
   print "Employment status: ", status
```

Review: What Decision Making Mechanisms Are Available/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
Compound decision making	More than one Boolean expression must be evaluated.
Nested decision making	The outer Boolean expression must be true before the inner expression will even be evaluated.

Common Errors

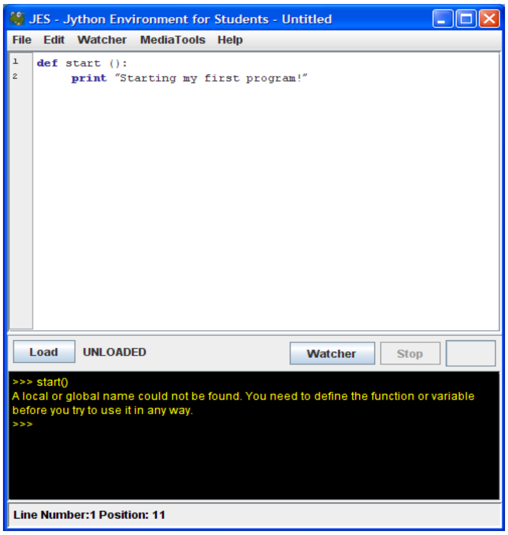
1. Inconsistent indenting (remember that you should only indent the 'body' of something).



```
def myBad ():
   print ":-)"
      print ":-("
```

Common Errors (2)

2. Forgetting to load your program before trying to run it.

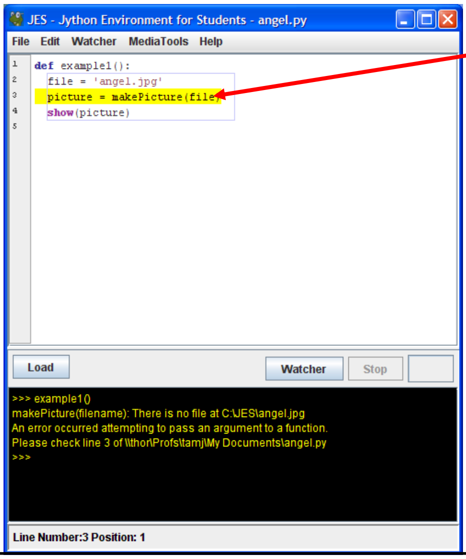

JES - Jython Environment for Students - Untitled
File Edit Watcher MediaTools Help
Load
UNLOADED
Watcher
Stop
Line Number:1 Position: 11

```
def start ():
    print "Starting my first program!"
```

```
>>> start()
A local or global name could not be found. You need to define the function or variable
before you try to use it in any way.
>>>
```

Common Errors (3)

3. Your program cannot find a file that it needs.


JES - Jython Environment for Students - angel.py
File Edit Watcher MediaTools Help
Load
Watcher
Stop
Line Number:3 Position: 1

```
def example1():
  file = 'angel.jpg'
  picture = makePicture(file)
  show(picture)
```

```
>>> example1()
makePicture(filename): There is no file at C:\JES\angel.jpg
An error occurred attempting to pass an argument to a function.
Please check line 3 of \\thor\Profs\tamj\My Documents\angel.py
>>>
```

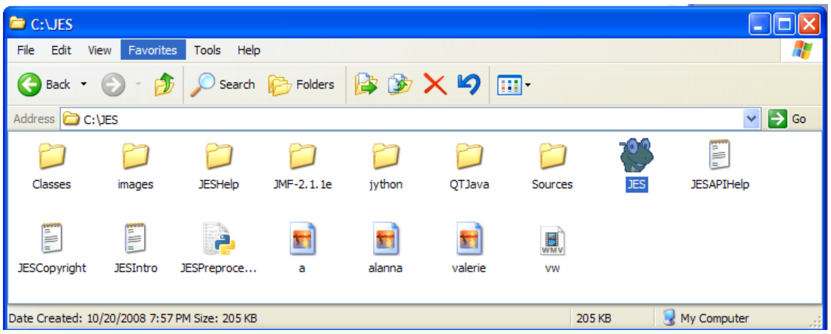
It can't find the file '*angel.jpg*' to make the picture

Common Errors (3)

3. Problem: Your program cannot find a file that it needs.

   Solution: Put all the files (pictures, sounds, videos) in the same folder where you put JES.

Example



Common Errors (4)

4. Forgetting required parts of the program
   - Forgetting the word "def" and/or the name of the program

(Erroneous version)

```
print "hello"
```

(Fixed version)

```
def programName ():
  print "hello"
```

- Forgetting the brackets and/or the colon:

```
def programName
  print "jello"
```

Common Errors (5)

4. Forgetting required parts of the program
   - Forgetting the quotes when displaying a string of characters:

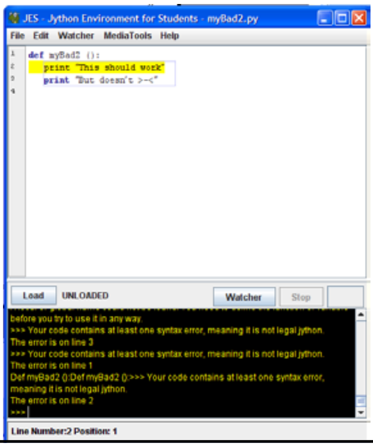
```
def programName ():
  print hello
```

   - Mismatching quotes:

```
def programName
  print “jello
```

Common Errors (6)

5. Typing in your JES program using a word processor or other program that allows for powerful text formatting. These errors can be very tough to find and fix because there is no visible error in your program (“computers suck!”)



```
def myBad2 ():
    print "This should work"
    print "But doesn't >-<"
```

```
before you try to use it in any way.
>>> Your code contains at least one syntax error, meaning it is not legal jython.
The error is on line 3
>>> Your code contains at least one syntax error, meaning it is not legal jython.
The error is on line 1
Def myBad2 ():Def myBad2 ():>>> Your code contains at least one syntax error,
meaning it is not legal jython.
The error is on line 2
>>>
```

Line Number:2 Position: 1

You Should Now Know

- How to create and run a program using JES
- How the print statement displays information
- What are variables and how to use them in a program
- How to load and display images in a program
- How to get input as a program runs
- The way in which decision making works with computer programs
- How to write and trace (determine the execution and output of) programs that employ: if, else-if, compound decision making and nested decision making mechanisms