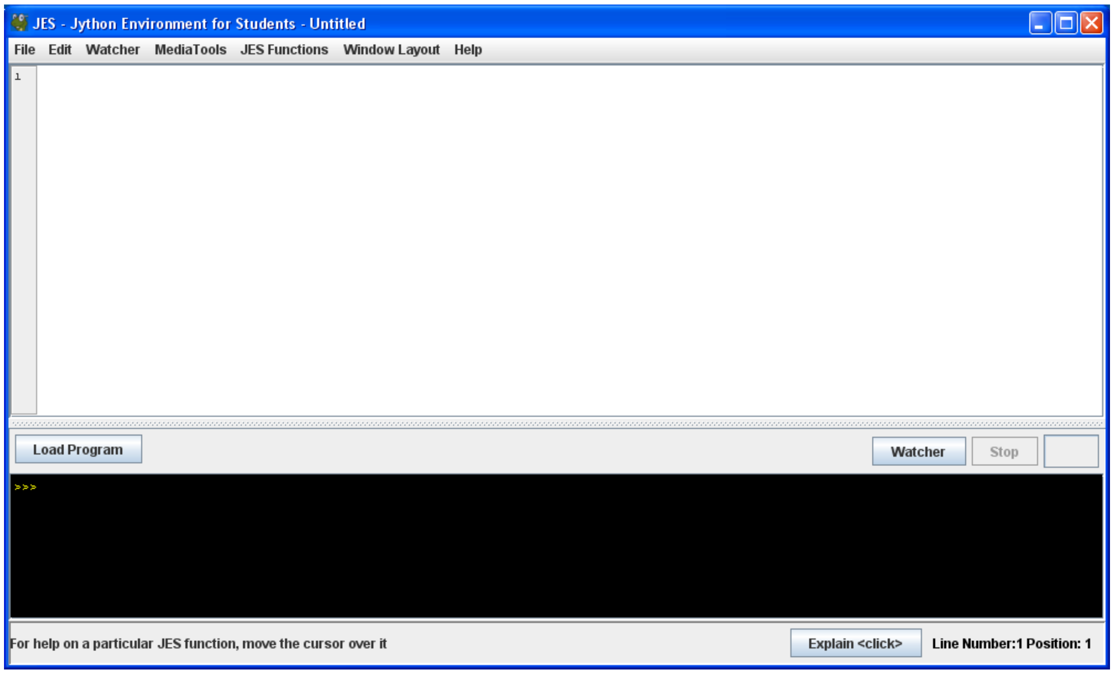
Programming in Python and JES

Throughout these notes, we will be using and referring to a tool called JES (Jython Environment for Students) to write programs with. But wait – I thought you said we were programming in Python?! It's ok, we are. Jython is an implementation (*i.e.*, a version) of the Python programming language written in Java. It allows Python code to work easily with Java code. We should also make note that some of the media names we will work with were developed to work with JES, but are not part of a normal Jython distribution.

Introduction to JES

As we just mentioned, we will be using a tool called JES to write and test our programs. This is what JES looks like when you run the application:


JES - Jython Environment for Students - Untitled
File
Edit
Watcher
MediaTools
JES Functions
Window Layout
Help
Load Program
Watcher
Stop
>>>
For help on a particular JES function, move the cursor over it
Explain <click>
Line Number:1 Position: 1

The top (white) part is the **program area**. This is where we will write our programs. The bottom (black) part is the **command area**. This is where will we command the computer to do something, such as run our programs.

Some examples of commands we could type into the command area are:

```
print 3 + 4
print "Hello class"
print "Hello" + "class"
print 34.1/46.5
print 1/2
```

(Note: Python understands many of the standard math operators.)

Exercise 1: Start up JES. Try entering the commands from the previous example into the command area in JES. Each command should be on a different line. So, for instance, the first line (in the black area) would look like:

```
>>>print 3 + 4
7
>>>
```

Follow-up Analysis: Were the results from this exercise what you expected? Why or why not? Was there a (small) difference between the 2nd and 3rd examples? If so, what was it? What happened in the last example? (You may need to discuss this with your instructor or class.)

We have just seen that we can use arithmetic operators and commands to make things happen in a program. When we used the `print` command, the word `print` is actually a **keyword** – a predefined word that has a specific meaning in the language. Programming languages often have a number of keywords. We will learn additional Python keywords as necessary.

In addition to using arithmetic operators and commands in a program, there are times when we will want to hold onto values for later use in a program. We will do this through the use of **constants** and **variables**. A **constant** is a value that does not change (*i.e.*, it is *constant*). Some examples of constants include 3, -15.2, "A", and "Hello". A **variable** is a value that can change throughout a program. In order to use variables, we need to give them identifiers (names). Identifiers can be almost anything (but cannot be the same name as a keyword), although good programmers try to give their variables names that reflect what the value represents. For example, if we have a variable that stores a file, we might name it *myFile*, or if we have a variable that represents a sum of numbers, we might name it *sum*. One of the features that makes Python easier to learn and use than some other languages is that you do not have to decide what type of data your variables will hold. Python takes care of those details behind the scenes. When you need to use a variable in a program, you just make up a name and use it. We do need to have some caution, though, because if you use the same name for two different values, the first value will be lost when you define the second.

Helpful Hint

When you need to use a variable in a program, you can just make up a name and use it. Be careful not to use the same name for two different values.

Aside: Rules and Conventions for Identifiers

- Names must start with a letter (usually lower case) or underscore character, but then may contain any number of letters, numbers, or other underscore characters.
- Names are case-sensitive: *Sum* and *sum* are different identifiers.
- It is a common convention that variable names that contain more than one word use underscores to separate the words, as in *my_file* or *average_of_three_numbers*, or use a capital letter at the beginning of words after the first, as in *myFile* or *averageOfThreeNumbers*.

It is now time to try out a few things. In this first mini-lab, you will learn some basics of JES and some basics for choosing and showing some pictures (and sounds).

Mini-Lab 1: Getting Started with JES