

Python

- The programming language we will be using is called *Python*
 - <http://www.python.org>
 - **It's used by companies like Google, Industrial Light & Magic, Nextel, and others**
- The *kind* of Python we're using is called Jython
 - **It's Java-based Python**
 - <http://www.jython.org>
- We use **Python** programming language with the aid of **JES** tools

Other Programming Languages

- Python


```
def hello():
    print "Hello World!"
```
- C


```
#include <stdio.h>
void main(){
    printf("Hello World !\n");
}
```
- Java


```
class HelloWorld {
    static public void main(String args[]){
        System.out.println("Hello World !");
    }
}
```
- Scheme

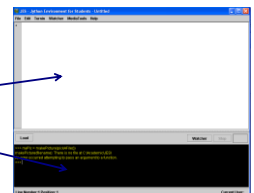

```
(define helloworld
  (lambda ()
    (display "Hello World !")
    (newline)))
```

Download JES

- JES
 - Create a 100 folder in C (C:\100)
 - Google search of 'JES download'
 - Download Windows version: jes-4-2-1-nojava.zip and select **'Save As'** to save it into C:\100
 - After download, right click on jes-4-2-1-nojava.zip, and select **'Extract All'** and **select 'C:\100' (not 'C:\100\jes-4-2-1-nojava')**
 - After unzipping it, you should have 'C:\100\jes-4-2-1-nojava' folder.
 - Go to the folder, and create a shortcut of JES.exe by right-clicking on JES.exe to desktop
- Download www.cs.uml.edu/~kim/100/mediasources.zip
 - Save into C:\100
 - Right click and 'Extract to mediasources\'

Python/JES

- JES IDE (Integrated Development Environment)
 - Incorporates editing environment
 - Program pane
 - Command pane
 - Watcher button to view debugging



Simple Computation in JES (+*/-%) in Command Pane

- Addition
3 + 4
- Multiplication
3 * 4
- Division
3 / 4
- Subtraction
3 - 4
- Negation
-4
- Modulo (Remainder)
10 % 2 and 11 % 2

Strings

- A sequence of characters
- Strings are defined with quote marks.
- Python actually supports three kinds of quotes:


```
>>> print 'this is a string'
this is a string
>>> print "this is a string"
this is a string
>>> print ""this is a string""
this is a string
```
- Use the right one that allows you to embed quote marks you want


```
>>> aSingleQuote = ""
>>> print aSingleQuote
.
```

Manipulating strings

- We can add strings and get their lengths

```
>>> hello = "Hello"
>>> print len(hello)
5
>>> mark = ", Mark"
>>> print len(mark)
6
>>> print hello+mark
Hello, Mark
>>> print len(hello+mark)
11
```

Getting parts of strings

- We use the square bracket "[]" notation to get parts of strings.
- string[n] gives you the nth character in the string
- string[n:m] gives you the nth up to (but not including) the mth character.

Getting parts of strings

```
>>> hello = "Hello"
>>> print hello[1]
e
>>> print hello[0]
H
>>> print hello[2:4]
ll
```

H	e	l	l	o
0	1	2	3	4

Start and end assumed if not there

```
>>> print hello
Hello
>>> print hello[:3]
Hel
>>> print hello[3:]
lo
>>> print hello[:]
Hello
```

Much of programming is about naming

- We name our data
 - Data: The "numbers" we manipulate
 - We call our names for data *variables*
- We name our recipes, call them *functions*
- Quality of names determined much as in Philosophy or Math
 - Enough words to describe what you need to describe
 - Understandable

Variables

- Name a value and re-use it
 - >>> x = 2*8
 - >>> x/2
 - >>> x
 - Whenever a variable name appears, it is **substituted** by its value
 - A variable name can be **reused, redefined**

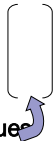
Naming Convention for both Variables and Functions

- MUST start with a letter followed by alphanumeric
 - aVar, cs100, ...
- **CASE** matters
 - 'Print' is not the same as 'print'
 - 'makePicture' is not the same as 'makepicture', nor as 'Makepicture'
- Multi-word name
 - First letter is capitalized
 - Convention, not absolute rule
- Use sensible, meaningful name

Functions

- A bunch of functions are pre-defined in JES for sound and picture manipulations

- pickAFile()
- makePicture()
- makeSound()
- show()
- play()



- Some of these functions accept *input values*

```
theFile = pickAFile()
pic = makePicture(theFile)
```

JES Functions

- Functions pre-defined in JES for image and sound manipulation
 - pickAFile(): allows a user to select a file
 - makePicture(): creates and returns a **picture object**
 - show(): display a picture in the argument
 - makeSound(): creates and returns an **audio object**
 - play(): plays out a sound
- **A function returns a value**
 - Like a quadratic $f(x) = x^2 + 5x - 10$, a $\sin(90)$, ...
- A variable vs. a function
 - **Variable** can be reused with **the same value** (until re-assigned)
 - **Function** can be reused to return **different values** according to arguments (some functions do not have arguments)

Sound Functions

- **makeSound(filename)** creates and returns a sound object, from the WAV file at the filename
- **play(sound)** makes the sound play
 - but doesn't wait until it's done
 - **blockingPlay(sound)** waits for the sound to finish
- We'll learn more later like **getSample** and **setSample**

Picture Functions

- `makePicture(filename)` creates and returns a picture object, from the JPEG file at the filename
- `show(picture)` displays a picture in a window
- We'll learn functions for manipulating pictures later, like `getColor`, `setColor`, and `repaint`

Values, names for those values, functions that return those values

- Completely INTERCHANGEABLE

- `>>> file = pickAFile()`
- `>>> print file`
- `C:\100\mediasources\barbara.jpg`

- `>>> show(makePicture(file))`
- `>>> show(makePicture(r"C:\100\mediasources\barbara.jpg"))`
- `>>> show(makePicture(pickAFile()))`

Put **r** in front of Windows filenames:
`r"C:\100\mediasources\pic.jpg"`