

# Introduction to Computing and Programming in Python: A Multimedia Approach

## Chapter 2: Introduction to Programming



# Chapter Learning Objectives

## Chapter Learning Objectives

The media learning goals for this chapter are:

- To make and show pictures.
- To make and play sounds.

The computer science goals for this chapter are:

- To use JES to enter and execute programs.
- To create and use variables to store values and objects, such as pictures and sounds.
- To create functions.
- To recognize different types (encodings) of data, such as integers, floating-point numbers, and media objects.
- To sequence operations in a function.



# Installation

- Installing JES and starting it up
  - Go to <http://www.mediacomputation.org> and get the version of JES for your computer.
    - If you know that you have a Java compiler (e.g., a “JDK” or an “IDE”)
  - Windows users:
    - Just copy the folder
    - Double-click JES application
      - If trouble, try jes2.bat or jes-customjava.bat
  - Mac users:
    - Just copy the folder
    - Double-click JES application
- There is always Help
  - Lots and lots of excellent help





# 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
- Quality of names determined much as in Philosophy or Math
  - Enough words to describe what you need to describe
  - Understandable



# Naming our Encodings

- We even name our encodings
  - Sometimes referred to as *types*
  - Numbers without decimals are called *integers*.
  - Numbers with decimal points are called *floating point* or *floats*.
  - Collections of letters are called *strings*.
- Some programming languages are *strongly typed*
  - A name has to be *declared* to have a type, before any data is associated with it



# Examples of Types

**31,364**

**12**

**Integers**

**-12**

**Inside the computer,  
these are *all* just bits**

**34,654.01**

**1.0**

**Floats**

**-1.01**

**0.01**

**The quick brown fox...**

**Tim Capes**

**Strings**

**40 St. George Street**



# Our programs work with a variety of names

- You will name your *functions*
  - Just like functions you knew in math, like sine and gcd (Greatest Common Divisor)
- You will name your *data (variables)*
- You will name the data that your functions work on
  - *Inputs*, like the 90 in sine(90)
- Key: Names inside a function only have meaning while the function is being executed by the computer. (You'll see what we mean.)



# Names for things that are *not* in memory

- A common name that you'll deal with is a *file name*
  - The program that deals with those is called the *operating system*, like Windows, MacOS, Linux
- A file is a collection of bytes, with a name, that resides on some external medium, like a *hard disk*.
  - Think of it as a whole bunch of space where you can put your bytes
- Files are typed, typically with three letter *extensions*
  - .jpg files are JPEG (pictures), .wav are WAV (sounds)



# We will program in JES

- **JES:** Jython Environment for Students
- A simple *editor* (for entering in our *programs* or *recipes*): We'll call that the *program area*
- A *command* area for entering in commands for Python to execute.



# Tour of JES

- Save and Save As
- Cut/Copy/Paste with shortcut keys
- Help

**If JES runs slow, close other applications.**

**Web browsers (like Firefox or Internet Explorer)  
and iTunes and chat tools and... all take up memory.  
Closing some of them saves memory for JES.**



# Python understands *commands*

- We can name data with =
- We can print values, expressions, anything with **print**



# Names can be (nearly) whatever we want

- Must start with a letter
- Be careful not to use command names as your own names
  - `print = 1` won't work
- *Case matters*
  - “Print” is not the same as “print”
  - “myPicture” is not the same as “mypicture”



# Using JES

```
>>> print 34 + 56  
90
```

```
>>> print 34.1/46.5  
0.73333333333333334
```

```
>>> print 22 * 33  
726
```

```
>>> print 14 - 15  
-1
```

```
>>> print "Hello"  
Hello
```

```
>>> print "Hello" + "Tim"  
HelloTim
```

**Adding integers**

**Dividing floats**

**Multiplying integers**

**Subtracting integers**

**Printing a string**

**Adding (concatenating)  
two strings**



# Values and names with same value are interchangeable

```
>>> print 12 * 3
```

```
36
```

```
>>> value = 12
```

```
>>> print value
```

```
12
```

```
>>> print value * 3
```

```
36
```

```
>>> name = "Tim"
```

```
>>> print name
```

```
Tim
```

```
>>> print name * 3
```

```
TimTimTim
```

```
>>> print "Tim" * 3
```

```
TimTimTim
```



# Math may be surprising sometimes

**If you only use integers (numbers without decimal points),  
Jython thinks you only want integers.**

```
>>> print 1.0/2.0  
0.5  
>>> print 1/2  
0
```



# Command Area Editing

- Up/down arrows walk through *command history*
- You can edit the line at the bottom
  - Just put the cursor at the end of the line before hitting Return/Enter.



# JES 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

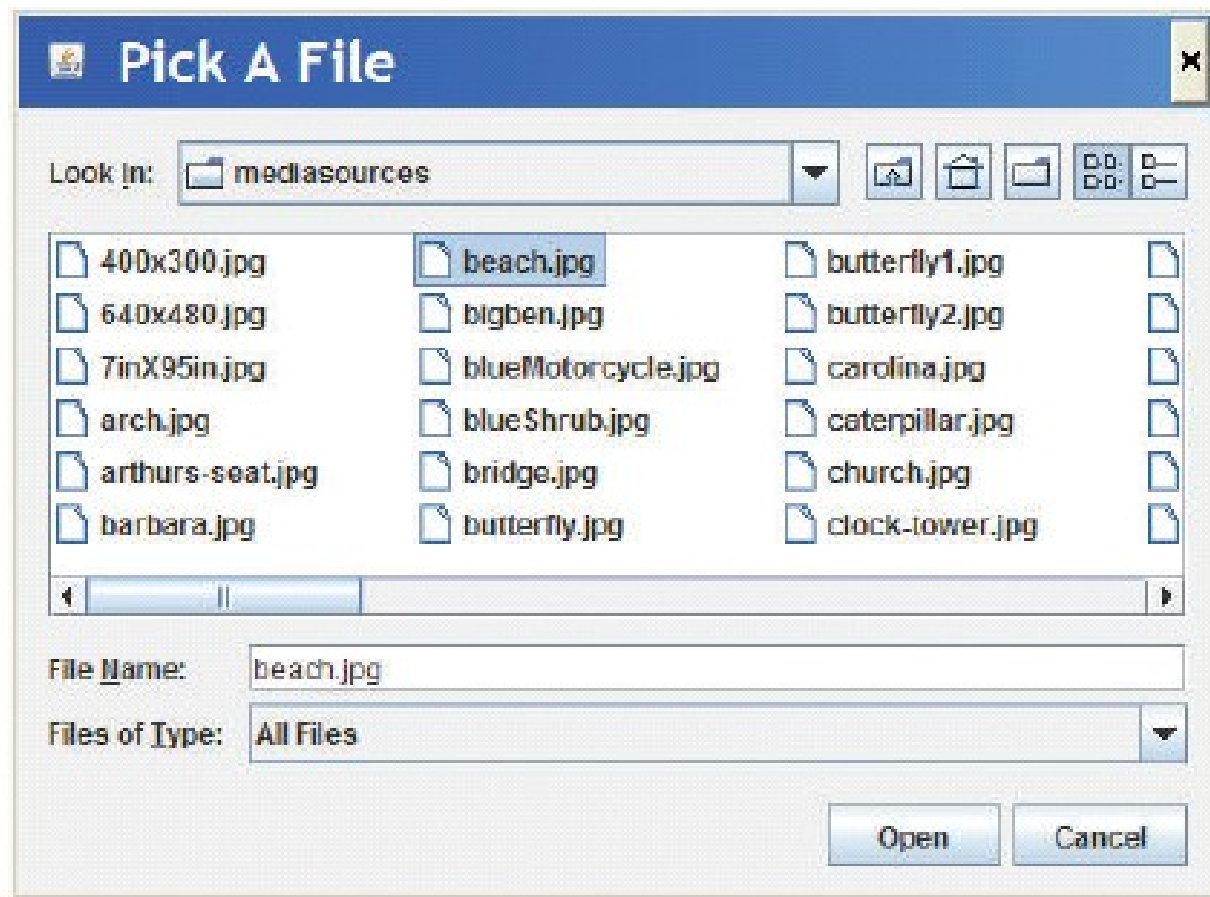


# What to do to show a picture

- 1. Find a file with a picture.
- 2. Pick it.
- 3. Get the bytes from that file into memory and label it as a type: “picture”
- 4. Show the picture



# pickAFile() leads to The File Picker!





# 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`



# 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`



# Demonstrating simple JES

```
>>> myfilename = pickAFile()
```

```
>>> print myfilename
```

```
/Users/guzdial/mediasources/barbara.jpg
```

```
>>> mypicture = makePicture(myfilename)
```

```
>>> print mypicture
```

```
Picture, filename /Users/guzdial/mediasources/barbara.jpg
```

```
height 294 width 222
```

```
>>> show(mypicture)
```



# Demonstrating simple JES

```
>>> print pickAFile()  
/Users/guzdial/mediasources/barbara.jpg  
>>> print makePicture(pickAFile())  
Picture, filename  
/Users/guzdial/mediasources/barbara.jpg height  
294 width 222  
>>> show(makePicture(pickAFile()))  
>>> print show(makePicture(pickAFile()))  
None  
>>> print pickAFile()  
/Users/guzdial/mediasources/hello.wav  
>>> print makeSound(pickAFile())  
Sound of length 54757  
>>> print play(makeSound(pickAFile()))  
None
```

***pickAFile()* returns a filename, which can be used as *input* to *makePicture()* to make a picture or *makeSound()* to make a sound.**

**Printing a picture just proves there's a picture there.**

***show()* and *play()* don't return anything, so they print *None*.**



## COMPLETELY THE SAME:

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

```
>>> file=pickAFile()
```

```
>>> print file
```

```
C:\Documents and Settings\Mark Guzdial\My  
Documents\mediasources\barbara.jpg
```

```
>>> show(makePicture(file))
```

```
>>> show(makePicture(r"C:\Documents and  
Settings\Mark Guzdial\My  
Documents\mediasources\barbara.jpg"))
```

```
>>> show(makePicture(pickAFile()))
```





# Picking, making, showing a picture



# Grabbing media from the Web

- Right-click (Windows) or Control-Click (Mac)
- Save Target As...
- Can *only* do JPEG images (.jpe, .jpg, .jpeg)

**Most images on the Internet are copyrighted. You can download and use them for your use *only* without permission. If you want to share your work with others use your own images.**



# Writing a recipe:

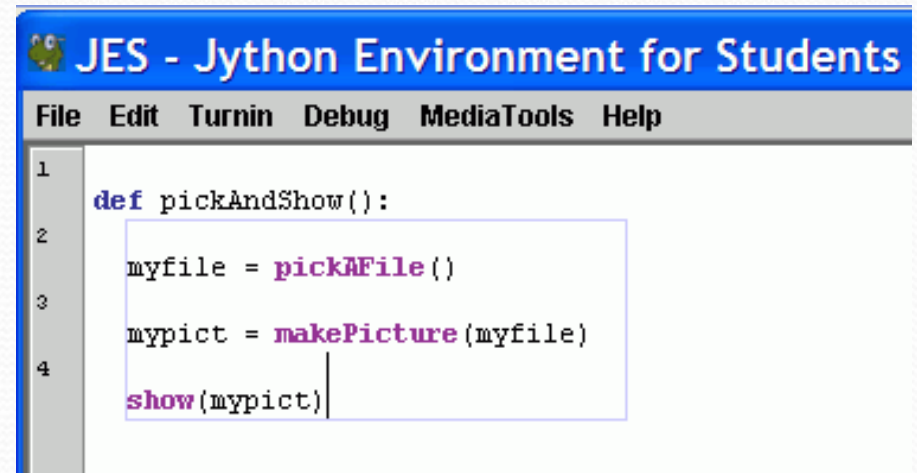
## Making our own functions

- To make a function, use the command **def**
- Then, the name of the function, and the names of the input values between parentheses (“(input1)”)
- End the line with a colon (“:”)
- The *body* of the recipe is indented (Hint: Use two spaces)
  - That’s called a *block*



## Blocking is indicated for you in JES

- Statements that are indented the same, are in the same block.
- Statements that are in the same block as where the line where the cursor is are enclosed in a blue box.



The screenshot shows the JES application window with a menu bar (File, Edit, Turnin, Debug, MediaTools, Help) and a code editor. The code editor contains the following Python code:

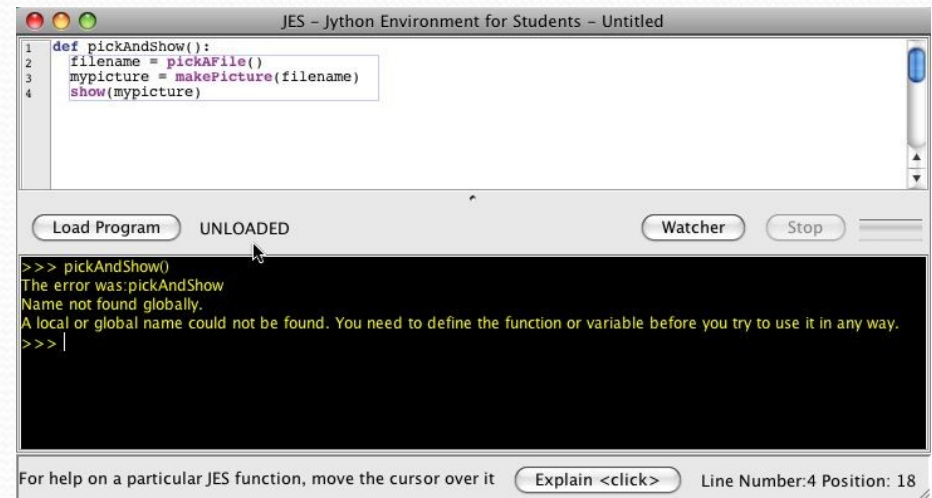
```
1 def pickAndShow():
2     myfile = pickAFile()
3     mypict = makePicture(myfile)
4     show(mypict)
```

A blue rectangular box highlights the code block starting from line 2, indicating that these statements are in the same block as the function definition on line 1. The cursor is positioned at the end of line 4.



# The Most Common JES Bug: Forgetting to Load

- Your function does **NOT** exist for JES until you *load* it
  - Before you load it, the program is just a bunch of characters.
  - Loading *encodes* it as an executable function
- Save and Save As
  - You must Save before Loading
  - You must Load before you can use your function



**An “Unloaded”  
function doesn’t exist  
yet.**



# Making functions the easy way

- Get something working by typing commands
- Enter the **def** command.
- Copy-paste the right commands up into the recipe



# A recipe for playing picked sound files

```
def pickAndPlay():  
    myfile = pickAFile()  
    mysound = makeSound(myfile)  
    play(mysound)
```

**Note:** **myfile** and **mysound**, inside **pickAndPlay()**, are *completely different* from the same names in the command area.



# A function for playing picked picture files

```
def pickAndShow():  
    myfile = pickAFile()  
    mypict = makePicture(myfile)  
    show(mypict)
```



# What if you forget your variable names? showVars()





# A function for a specific sound or picture

```
def playSound():  
    myfile = "FILENAME"  
    mysound = makeSound(myfile)  
    play(mysound)
```

```
def showPicture():  
    myfile = "FILENAME"  
    mypict = makePicture(myfile)  
    show(mypict)
```

**You can always replace data (a *string* of characters, a number, whatever) with a name (*variable*) that holds that data—or vice versa.**

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



# What to do about Windows filenames?

- Python doesn't like you to use “\” in filenames, like “C:\mediasources\barbara.jpg”
- What to do?
  - Option #1: Put r in front of Windows filenames:  
r“C:\mediasources\pic.jpg”
  - Option #2: Use forward slashes. Python will translate it for you:  
“C:/mediasources/pic.jpg”



# A function that takes input

```
def playNamed(myfile):  
    mysound = makeSound(myfile)  
    play(mysound)
```

**What functions do you need? What should be their input?**

```
def showNamed(myfile):  
    mypict = makePicture(myfile)  
    show(mypict)
```

**In general, have enough to do what you want, easily, understandably, and in the fewest commands.**

**We'll talk more about what that means later.**



# What can go wrong?

- Did you use the *exact* same names (case, spelling)?
- All the lines in the block must be *indented*, and *indented the **same** amount*.
- Variables in the command area don't exist in your functions, and variables in your functions don't exist in the command area.
- The computer can't read your mind.
  - It will only do exactly what you tell it to do.



# MOST IMPORTANT THING TO DO TO PASS THIS CLASS!

- *DO THE EXAMPLES!*
- Try them out for yourself. Try to replicate them.  
*Understand them*
  - EVERY WEEK, TYPE IN AT LEAST TWO OF THE EXAMPLES FROM CLASS
- To understand a program means that you know why each line is there.
- You will encounter all the simple-but-confusing errors *early*—**BEFORE** you are rushing to get homework done!!