Introduction to Scientific Programming

Prof. Dan Marzin

Introduction to IES

These slides are an extensively modified version of material from http://coweb.cc.gatech.edu/medialabs/ies/ies and Dr. Jennifer Kay

Very Important!!!!

• If you missed the first lecture, you MUST talk to me after class!!!

What the heck is a program?

• Instructions that tell the computer what to do
• Think: A Very Carefully Written Recipe

Key Concept in Programming: Computers are Not Very Smart!

• Computers are not very smart. They take everything you say very literally. They do what you tell them to do rather than what you want them to do
  • Even if what you tell them to do doesn’t make sense
  • Even if what you want them to do is obvious
• When something doesn’t work, remembering this key idea will help you to find the problem
Programming Languages

- Different programming languages are different ways (encodings) that turn into (same/similar) commands for the computer.

Python/Java

```python
def hello():
    print("Hello World")

Java

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

Geek Speak

- A **program** is a description in a programming language of a process that achieves some result.
- An **algorithm** is a description of a process in a step-by-step manner.
- The same algorithm could be written in many languages.

Computers: Fast but Stupid

- The only data they understand is 0’s and 1’s
- They can only do the most simple things with those 0’s and 1’s
- Move this value here
- Add, multiply, subtract, divide these values
- Compare these values, and if one is less than the other, go follow this step rather than that one.
- Done quickly enough, those simple things can be amazing.

Geek Speak: Digital Media

- **Digitizing media** is encoding media into numbers
- Converting an analogue (continuous) picture drawn with a crayon into a digital (discrete) picture
- Crayon -> Pixels
OK, So how does a Computer "See" a picture?

A picture drawn with crayons on paper can be thought of as continuous or analogue. There are an infinite number of colored points in between the location where you put the crayon down and start to draw, and the location where you pick it up again.

Hang on ... this picture is on a computer and you didn’t get it here with a crayon, what’s up with that??

I scanned the picture at 72 dpi

72 dpi = 72 Dots per inch

So the computer just thinks about pixels

Actual size of this picture: 5.107 in x 5 in = 25.535 square inches

At 72 dots per inch, that’s 383x375 pixels = 143,625 pixels!!

How can it work to digitize media?

• Why can we break it into pieces?
• Why don’t we perceive the breaks?
  • Human perception is limited.
  • We don’t see the dots in the pictures
  • We don’t hear the gaps in the sounds.
  • Flip books
Digitized Pictures and Sounds use a LOT of Data

• One byte = 8 digits (1s and 0s)
  • To represent all the dots on your screen:
    • More than 3,145,728 bytes

• One second of sound on a CD:
  • 44,100 bytes

Computers are dumb, but FAST
Get the computer to do the recipe!

• Make it hard / tedious / complex
• Crank through a million genomes? No problem!
• Find one person in a 30,000 campus? Yawn!
• Process a million dots on the screen or a bazillion sound samples?
  • That’s media computation

We Will Program in JES

• JES: Jython Environment for Students
• Question: Does everyone have a desktop or laptop they can install the software on?
• Installing JES and starting it up:
  • See class home page for links

JES Tip

• 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.
Switch to JES demo ...

DEMO: At Startup

DEMO: JES with help displayed

Use Window Layout to get the view you want

Demo: Using the Command Area

```python
>>> print 34+56
90
>>> print 10.0/3.0
3.3333333333333335
>>> print "Rowan"
Rowan
>>> print 10/3
3
```
PAUSE DEMO!!

• Why did you need quotation marks???
• Why didn’t the math work sometimes??

>>> print 34+56
90
>>> print 10.0/3.0
3.3333333333333335
>>> print ”Rowan”
Rowan
>>> print 10/3
3

The Problem – Computers only Understand 1s and 0s

• In memory, the computer stores some number, like 01001011
  • 01001011 could represent:
    • The whole number 75
    • The decimal number 4.11
    • The letter ‘K’
    • This color: 
    • A sound
    • A whopper with cheese
    • The model number of a stereo system

The Problem – Computers only Understand 1s and 0s

• We need to give the computer hints
  • How to translate things on the screen into 1s and 0s
  • How to translate the 1s and 0s in the computer’s “memory” into things we understand

• print 10.0 / 3.0
  • The decimal point is a “hint” that we want the computer to use decimal math

• print 10 / 3
  • No decimal point – the computer figures you just like to use whole numbers

Types: Naming our Encodings

• INTEGERS: Numbers without decimals
• FLOATING POINT (or FLOATS): Numbers with decimal points
• STRINGS: Collections of letters & numbers (things you can type on the keyboard)

• Tricky!!!
  • 15 is an integer
  • 15.0 is a float
  • “15” is a string
**Examples of Types – Human View**

Integers
- 31,364
- 12
- -12

Floats
- 12.998
- 34,654.01
- 1.01
- 0.01

Strings
- Glassboro, NJ
- Prof. Marzin
- Rowan

**Examples of Types – How to be clear for the computer**

Integers
- 31,364
- 12
- -12

Floats
- 34,654.01
- 1.01
- 0.01
- 12.998

Strings
- Glassboro, NJ
- Prof. Marzin
- Rowan
- "Glassboro, NJ"
- "Prof. Marzin"
- "Rowan"

**Demo: More Command area**

```python
>>> print 34 + 56
90
>>> print 34.1/46.5
0.7333333333333334
>>> print 22 * 33
726
>>> print 14 - 15
-1
>>> print "Hello"
Hello
>>> print "Hello" + "Mark"
HelloMark
```

Adding integers
Dividing floats
Multiplying integers
Subtracting integers
Printing a string
Adding (concatenating) two strings

**Demo: Using Names**

```python
>>> totalJellyBeans = 25
>>> numberEaten = 10
>>> jellyBeansRemaining = totalJellyBeans - numberEaten
>>> print jellyBeansRemaining
15
>>> print "jellyBeansRemaining"
jellyBeansRemaining
```
Tricky!!

```python
>>> print jellyBeansRemaining
15
>>> print "jellyBeansRemaining"
```

google

Some programming languages are **strongly typed**
- Strongly typed languages (NOT PYTHON!!)
  - A name has to be declared to have a type, before any data is associated with it
    ```python
    Integer numberEaten;
    numberEaten = 15
    ```
  - Python (not strongly typed)
    ```python
    >>> numberEaten = 15
    ```
    Python says "Oh, numberEaten must be an integer"

Geek Speak: Variables
- We call the names for data **variables**
  - Numbers:
    ```python
    >>> age = 19
    ```
  - Strings:
    ```python
    >>> school = "Rowan"
    >>> place = "school"
    ```
- **NOTES:**
  - Variable names: age, school, place
  - Integer: 19
  - Strings: "Rowan", "school"

Names
- So far: Names for Variables
- Coming soon:
  - Names for functions (just like math!)
    - Sine
    - Square Root
    - Is it a leap year
  - Names for files
    - myCat.jpg, mySong.wav, myProgram.py

In Memory

On your hard drive
DEMO: Values and names with same value are interchangeable

```python
>>> print 12 * 3
36
>>> value = 12
>>> print value
12
>>> print value * 3
36
```

DEMO: Values and names with same value are interchangeable

```python
>>> name = "Dan"
>>> print name
Dan
>>> print name * 3
DanDanDan
>>> print "Dan" * 3
DanDanDan
```

Keeping track of your variables

- Dr. Kay’s weird Scrap Paper Model:
  - Imagine that the computer has a big piece of scrap paper titled “command area” where it keeps all the variables it makes in the command area

```python
>>> age = 19
>>> school = "Rowan"
>>> place = "school"
>>> age = 23
>>> age = age + 1
```

Assignment Statements (Equals)

- Do each step one at a time
- Evaluate the right side and put answer in left side
- Don’t try to be smart!

```python
>>> year = 2013
>>> age = year - 1984
>>> year = year + 3
>>> print year
>>> print age
```

Trace This!!!!
DEMO: 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.

DEMO: Take care with strings

```python
>>> school = "Rowan"
>>> print school
Rowan
>>> print Rowan
The error was:Rowan
Name not found globally.
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.
```

Geekspeak: Commands

- An instruction in Python
- `=` (equals) is a Python command ("assignment statement")
  ```python
  >>> age = 19
  >>> print age
  19
  >>> print "Rowan"
  Rowan
  >>> print age * 3
  57
  ```

Variable Names can be (nearly) whatever we want

- Must start with a letter*
- Then any sequence of letters, numbers, or underscores
- Be careful not to use command names as your own names
  - OK:
    ```python
    >>> printy = 1
    1
    ```
  - Not OK
    ```python
    >>> print = 1
    Trace This!!!!
    ```

*Strictly speaking you can start with an underscore, but don’t!
Variable Names (cont)

- Case matters
  - Print is not the same as print
  - myPicture is not the same as mypicture
- Names must be good!!
  - What’s good? I know it when I see it!
  - Others can understand it easily

*Strictly speaking you can start with an underscore, but don’t!*

Don't forget that Jython uses "stupid math" whenever it can

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

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

Writing Recipes

- So far we’ve only executed one command at a time
- Most of the time we want to do more than that

Don't Panic – We’ll get to pictures soon!!

- But let’s start with the basics – stuff you know how to do with a pencil and paper
- A recipe to find the cube of a number:

```python
def computeCube (num):
  answer = num * num * num
  print answer
```
Geek Speak - Terms
- Recipe
- Program
- Function

Running Compute Cube

Once we've got a function loaded, we can use it as often as we want

```python
def computeCube (num):
    answer = num * num * num
    print answer
```

This line is called our "function definition" (a.k.a. "heads up computer, this is a recipe!")

We name our functions (recipes)
This one is called computeCube

Arguments (ingredients) to the function go in the parentheses
The indented stuff is the "function body" - the instructions for the actual recipe itself

Arguments (inputs)

def means: "Get ready, I’m going to give you a recipe (but don’t do it yet)"

Running Compute Cube

Once we've got a function loaded, we can use it as often as we want

```python
== Loading Program ==
>>> computeCube(3)
27
>>> z = 5
>>> computeCube(z)
125
>>> foo = 2
>>> computeCube(foo)
8
>>> computeCube(z+foo)
343
```
Lots more recipes also find the cube of a number

```python
def computeCube(num):
    answer = num * num * num
    print answer
```

```python
def myLittlePony(num):
    answer = num * num * num
    print answer
```

```python
def iceCream(yum):
    dessert = yum * yum * yum
    print dessert
```

```python
def a(b):
    c = b * b * b
    print c
```

```python
def this_is_super_long(b):
    c = b * b * b
    print c
```

```python
def this_is_super_long(boy_is_this_long):
    xyz = boy_is_this_long * boy_is_this_long * boy_is_this_long
    print xyz
```

```python
def sTaRwArS(LUKE):
    yodA = LUKE * LUKE * LUKE
    print yodA
```

```python
def this_is_super_long(boy_is_this_long):
    xyz = boy_is_this_long * boy_is_this_long * boy_is_this_long
    print xyz
```

```
Let's do more examples ...

1. Write a function:
   - That takes your age as an argument and tells you what year you were born.
   - That takes your name as an argument and prints it out 3 times in a row
   - That takes the length of one side of an equilateral triangle as an argument and prints out its area
   - That takes your favorite flavor of ice cream as an argument and prints a message about how yummy it is

```
Functions can have more than one argument

def computeCube(num):
    answer = num * num * num
    print answer

def howOldAreYou(yearBorn, currentYear):
    age = currentYear - yearBorn
    print "At the end of this year, you will be" "years old!"
    print age

def flavors(favorite, prettyGood, yuck):
    print "Wow, so you really love"
    print favorite
    print "You kind of like"
    print prettyGood
    print "and you hate"
    print yuck

Running functions with multiple arguments – order matters!!!
Weird?!? Or not so weird?!?

```python
def howOldAreYou(yearBorn, currentYear):
    age = currentYear - yearBorn
    print "At the end of this year, you will be"
    print age
    print "years old!"

>>> now = 2013
>>> then = 1992
>>> howOldAreYou(then, now)
At the end of this year, you will be
21
years old!
```

Computers aren’t smart, but they are good at keeping track of things

```python
def howOldAreYou(yearBorn, currentYear):
    age = currentYear - yearBorn
    print "At the end of this year, you will be"
    print age
    print "years old!"

>>> now = 2013
>>> then = 1992
>>> howOldAreYou(then, now)
At the end of this year, you will be
21
years old!
```

How to Trace a Function (computeCube)

1. Get a new piece of scrap paper – write the name of the function at the top
2. Add boxes for the arguments (if there are any)
3. Fill in the values for the arguments
4. Run the function
5. (Return as necessary)
6. Throw out the scrap paper

```python
def computeCube (num):
    answer = num * num * num
    print answer
```

How to Trace a Function (a)

1. Get a new piece of scrap paper – write the name of the function at the top
2. Add boxes for the arguments (if there are any)
3. Fill in the values for the arguments
4. Run the function
5. (Return as necessary)
6. Throw out the scrap paper

```python
def a (b):
    c = b * b * b
    print c
```
How to Trace a Function (prettyFlower)

1. Get a new piece of scrap paper – write the name of the function at the top
2. Add boxes for the arguments (if there are any)
3. Fill in the values for the arguments
4. Run the function
5. (Return as necessary)
6. Throw out the scrap paper

```python
def prettyFlower():
    print "   _   ")
    print " _(_)_ ")
    print " (_)@(_)")
    print "  (_)  ")
```

How to Trace a Function (instructions)

1. Get a new piece of scrap paper – write the name of the function at the top
2. Add boxes for the arguments (if there are any)
3. Fill in the values for the arguments
4. Run the function
5. (Return as necessary)
6. Throw out the scrap paper

```python
def instructions():
    print "Welcome to tic tac toe!"
    print "You will be x and I will be o."
    print "Click on the square where you want to start"
```

How to Trace a Function (howOld)

1. Get a new piece of scrap paper – write the name of the function at the top
2. Add boxes for the arguments (if there are any)
3. Fill in the values for the arguments
4. Run the function
5. (Return as necessary)
6. Throw out the scrap paper

```python
def howOldAreYou(yearBorn, currentYear):
    age = currentYear - yearBorn
    print "At the end of this year, you will be" + str(age)
    print "years old!"
```

How to Trace a Function (flavors)

1. Get a new piece of scrap paper – write the name of the function at the top
2. Add boxes for the arguments (if there are any)
3. Fill in the values for the arguments
4. Run the function
5. (Return as necessary)
6. Throw out the scrap paper

```python
def flavors(favorite, pretty_good, yuck):
    print "Wow, so you really love"
    print favorite
    print "You kind of like"
    print pretty_good
    print "and you hate"
    print yuck
```
What does this function do??

```python
def mystery (h, w):
a = h * w
p = 2 * (h+w)
```

Try running this in JES!!

Key Concept in Programming: Computers are Not Very Smart!

- Computers are not very smart. They take everything you say very literally. They do what you tell them to do rather than what you want them to do
- Even if what you tell them to do doesn’t make sense
- Even if what you want them to do is obvious
- When something doesn’t work, remembering this key idea will help you to find the problem

How to Trace a Function (mystery)

1. Get a new piece of scrap paper – write the name of the function at the top
2. Add boxes for the arguments (if there are any)
3. Fill in the values for the arguments
4. Run the function
5. (Return as necessary)
6. Throw out the scrap paper