Topics for Today

- Reading assignment (Zelle):
  - Chapter 1: Computers and Programs
  - Chapter 2: Writing Simple Programs
  - Chapter 3: Computing with Numbers (finish next week)
- Starting next week
  - Chapter 4: Computing with Strings

- Questions on downloading software?

Starting our whirlwind tour of programming and Python…

Continuing from last time …

- ```
  >>> def hello():
          print "Hello"
          print "Welcome to CS 190C"


  >>>
  ```
- The first line tells Python we are *defining* a new function called hello.
- The following lines are (automatically) indented to show that they are part of the hello function.
- The blank line (hit enter twice) lets Python know the definition is finished.
  ```

```
>>> def hello():
        print "Hello"
        print “Welcome to CS 190C"


>>>
```

- Notice that nothing has happened yet!
- We’ve defined the function, but we haven’t told Python to perform the function!

- A function needs to be *invoked:*

```
>>> hello()
Hello
Welcome to CS190C
```

```
>>> def hello():
        print "Hello"
        print "Welcome to CS 190C“
```

- Noted the empty parentheses. What are they for?
  - Functions can be “parameterized”: allows values (aka “arguments”) to be substituted for parameters in the function body
  - Side point: Most syntax is there for a reason

Function with Parameter(s)

```
>>> def greet(person):
        print "Hello", person
        print "How are you?"
```

Function with Parameters

```
 >>> greet("Terry")
Hello Terry
How are you?
>>> greet("Paula")
Hello Paula
How are you?
>>>
```

- Note the inserted space.
- When we use parameters, we can customize the behavior of our function.

Clicker Question

```
>>> print "Calculate:", "2 * 3", "is", 2 * 3
```

A. Calculate: 2*3 is 6

B. Calculate: 2 * 3 is 6

C. Calculate:6 is 6

D. Calculate: 2 * 3 is 2 * 3

Writing Programs with Python

- When we exit the Python prompt, the functions we've defined cease to exist
- Programs are usually composed of functions, *modules*, or *scripts* that are saved on disk so that they can be used again and again.
- A *module file* is a text file created in text editing software (saved as "plain text") that contains function definitions.
- A *programming environment* is designed to help programmers write programs and usually includes automatic indenting, highlighting, etc.

Python Program Levels

- Programs are composed of modules
  - Modules are imported or written by you
- Modules contain statements
  - Statements includes loops, conditions, assignments
- Statements contain expressions
  - Expressions include objects (e.g., numbers) and operators that compute a value
- Expressions create and process objects
  - Objects are built-in or created

Another Example

```
# File: chaos.py
# A simple program illustrating chaotic behavior

def main():
    print "This program illustrates a chaotic function"
    x = input("Enter a number between 0 and 1: ")
    for i in range(10):
        x = 3.9 * x * (1 - x)
        print x

main()
```

- We'll use the extension *.py* when we save our work to indicate it's a Python program.
- In this code we're defining a new function called **main**.
- The main() at the end tells Python to run the code.

Output from the Chaos Program

```
>>>
This program illustrates a chaotic function
Enter a number between 0 and 1: .5
0.975
0.0950625
0.335499922266
0.869464925259
0.442633109113
0.962165255337
0.141972779362
0.4750843862
0.972578927537
0.104009713267
>>>
```

Inside a Python Program

```
# File: chaos.py
# A simple program illustrating chaotic behavior
```

- Lines that start with # are called *comments*
- Intended for human readers and ignored by Python
- Python skips text from # to end of line

Inside a Python Program

```
def main():
```

- Beginning of the definition of a function called *main*
- Since our program has only this one module, it could have been written without the *main* function.
- The use of *main* is customary, however.

Inside a Python Program

```
print "This program illustrates a chaotic function"
```

- This line causes Python to print a message introducing the program.

Inside a Python Program

```
x = input("Enter a number between 0 and 1: ")
```

- x is an example of a *variable*
- A variable is used to assign a name to a value so that we can refer to it later.
- The quoted information is displayed, and whatever the user types in response is stored in x.

Inside a Python Program

```
for i in range(10):
```

- For is a *loop* construct
- A loop tells Python to repeat the same thing over and over.
- In this example, the following code will be repeated 10 times.
  - variable i will take on values 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Inside a Python Program

```
x = 3.9 * x * (1 - x)
print x
```

- These lines are the *body* of the loop.
- The body of the loop is what gets repeated each time through the loop.
- The body of the loop is identified through indentation.
- The effect of the loop is the same as repeating these two lines 10 times

Inside a Python Program

```
for i in range(10):
        x = 3.9 * x * (1 - x)
        print x
```

- These are equivalent

```
x = 3.9 * x * (1 - x)
print x
x = 3.9 * x * (1 - x)
print x
x = 3.9 * x * (1 - x)
print x
 x = 3.9 * x * (1 - x)
print x
 x = 3.9 * x * (1 - x)
 print x
 x = 3.9 * x * (1 - x)
 print x
 x = 3.9 * x * (1 - x)
 print x
 x = 3.9 * x * (1 - x)
 print x
 x = 3.9 * x * (1 - x)
 print x
 x = 3.9 * x * (1 - x)
 print x
```

Inside a Python Program

```
x = 3.9 * x * (1 - x)
```

- This is called an *assignment* statement
- The part on the right-hand side (RHS) of the "=" is a mathematical expression.
- * is used to indicate multiplication
- Once the value on the RHS is computed, it is stored back into (*assigned*) into x

Inside a Python Program

```
main()
```

- This last line tells Python to *execute* the code in the function *main*

Chaos and Computers

- chaos.py:

```
def main():
    print "This program illustrates a chaotic function"
    x = input("Enter a number between 0 and 1: ")
    for i in range(10):
        x = 3.9 * x * (1 - x)
        print x

main()
```

- For any given input, it returns 10 seemingly random numbers between 0 and 1
- It appears that the value of x is *chaotic*

Chaos and Computers

Input: 0.25	**Input: 0.26**
0.73125	0.75036
0.76644140625	0.73054749456
0.698135010439	0.767706625733
0.82189581879	0.6954993339
0.570894019197	0.825942040734
0.955398748364	0.560670965721
0.166186721954	0.960644232282
0.540417912062	0.147446875935
0.9686289303	0.490254549376
0.118509010176	0.974629602149

Clicker Question

What is an "algorithm"?

A. A mathematical formula
B. Program statements
C. Precise steps to solve a problem
D. A module
E. Problem description

Clicker Question (participation)

for i in range(10): iterates over 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

range(0,10,1) generates the same result

What does **for i in range(1,10,2**): iterate over

A. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

B. 0, 2, 4, 6, 8,10

C. 1, 3, 5, 7, 9

D. Don't know

The Programming Process

- Start with an idea of a problem to solve
  - The **specification**
- Determine a precise way to solve it
  - The **algorithm**
- Write sequence of statements that implement algorithm
  - The **code**

- Complex problems require built-up solutions
  - Decompose problem into sub-problems
  - Introduce abstractions
  - Create functions

What Can a Program Do? (1)

- Compute things—"expressions"
- Compute with numbers
  - 1 + 1
  - 2 * (3 - 5)
- Compute with strings
  - "Go" + " " + "Boilers"

What Can a Program Do? (2a)

- Name things—called “variables” and “functions” (and more)
- Name values created by expressions… variable assignment

```
x = 5*7
y = 3*(2-5)
z = "Alice and Bob"
a, b = 12, 27
```

What Can a Program Do? (2b)

- Name things—called "variables" and "functions" (and more)
- Name sequences of statements… function definition

```
def doit():
    a = 5
    b = 7
    print "hello there", a*b
```

What Can a Program Do? (2c)

- Name more things
- Functions can take parameters and return values

```
def docalc(a, b):
    t1 = a – b
    t2 = a + b
    return t1 * t2
docalc(4, 5)
```

What Can a Program Do? (2d)

- Name still more things…
- Modules (files) and classes (objects) also have names

```
import myfile
myfile.docalc(3, 5)

from visual import *
ball = sphere()
ball.color = color.red
```

What Can a Program Do? (3)

- Input

  ```
  x = input("Enter a number: ")
  ```

- Output

  ```
  print "x and y are", x, y
  ```

What Can a Program Do? (4)

- Repeat things

```
x = 0.35
for i in range(10):
    x = 3.9 * x * (1 – x)
    print x
```

- A loop is often used to accumulate a sum

```
n = 0
for i in range(10):
    n = n + input("next value: ")
print n
```