CS 100: Practice on Python Drawing

Chris Kauffman

Week 4-1

Logistics

Homework 3

- Due next Thursday
- Can work with partner
- Submit both Word Doc/PDF AND Python code

Reading

- Pattern Ch 4
- Zyante Ch 4
- Think Ch 3-7

Mini-Exam

Will return and discuss on Thursday

Goals Today

- Python basics
- Drawing Exercises

Staying Organized

- HW and python files
- Single `Desktop/cs100/hw3` directory
  - `Homework 3.doc` (written HW)
  - `hw3.py` which contains code for the HW
- When HW 4 rolls around, make `Desktop/cs100/hw4`
  - `Homework 4.doc` (written HW)
  - `hw4.py` for code
- When working in class, create a file for the days work
  - `classwork_9_16.py` (spaces screw things up)

Quick Review

- How do you "start" python for CS 100?
- Where is the best place to write your programs for the class?
- How do you get the code you write to run?

Exercise: Draw a plain house

Basic commands

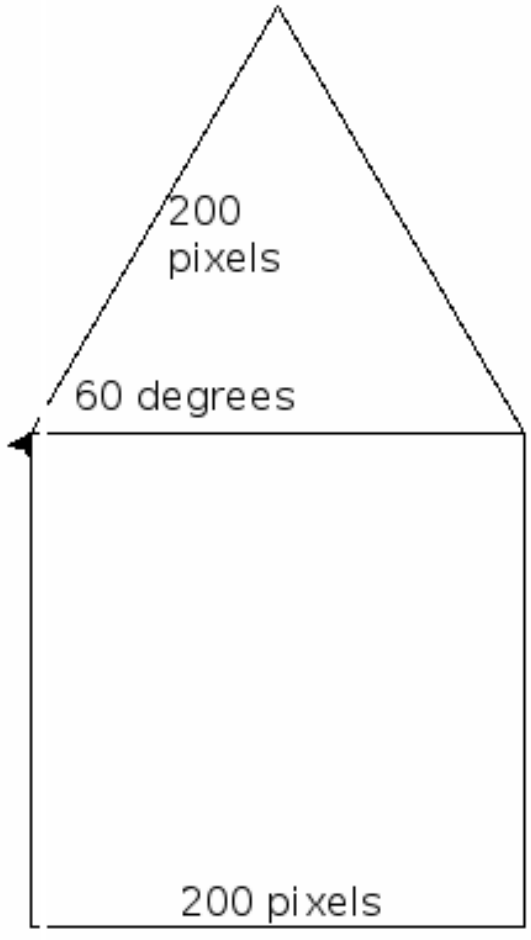
```
forward(length)
right(angle)
left(angle)
```

Repetition

```
for i in range(4):
  forward(100)
  right(90)

backward(200)
```

Spaces to indent loops



Spaces in Python

Spaces between things doen't matter too much

```
x = 1                                   # Assign x to be 1
x=2                                     # Assign x to be 2
x    =   3                              # Assign x to be 3

for i in range(4):                      # Repeat 4 times
    print(i)

for   i  in range(  4):                 # Repeat 4 times
    print(i)
```

Spaces in Python

Spaces in front of things matter a lot

```
x = 1                                   # Assign x to be 1
  y=2                                   # Error!

if (x > 2):                             # Indent things that should
    print("x > 2")                      # be done if x > 2
    print(x)
else:                                   # Indent things to do
    print("x <= 2")                     # when x <= 2
    if(x == 2):                         # Check if x is 2
        print("x is 2")                 # Print if it is
print("All done")                       # ALWAYS do this

for i in range(4):
    print(i)                            # Do this 4 times
print("hi")                             # Do this once
```

Color Names as Strings

```
from turtle import *
color(x,x)                          # what is x?
color(blue,blue)                    # what is blue?
color("blue","blue")                # I know the "word" blue!
```

Bare names like

```
blue red
```

are treated as variables, often undefined
Things in quotes like

```
"blue" "red" "Several colors at once"
```

are string literals: "wordy" data

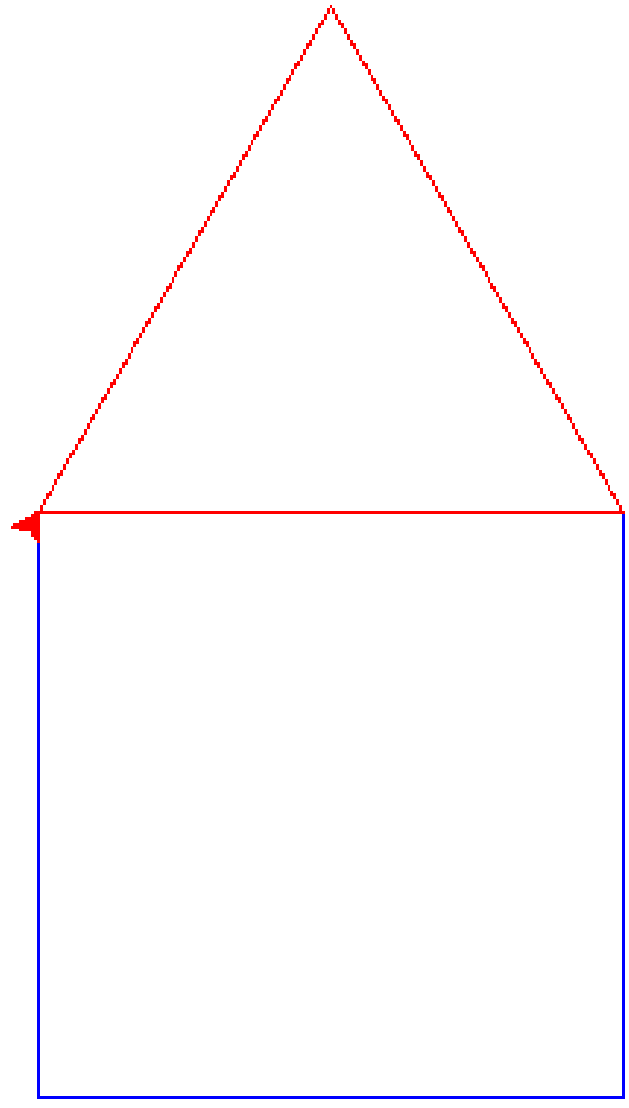
Exercise: Colored House

Add `color("something")` commands

```
from turtle import *

for i in range(4):
  forward(200)
  right(90)

left(60)
for i in range(3):
  forward(200)
  right(120)
```



Filling Areas with Color

New Commands

`begin_fill()` and `end_fill()` can create shapes filled with color.

- Call `begin_fill()` to start coloring
- Looks like nothing happens
- When `end_fill()` is called, will fill in an area

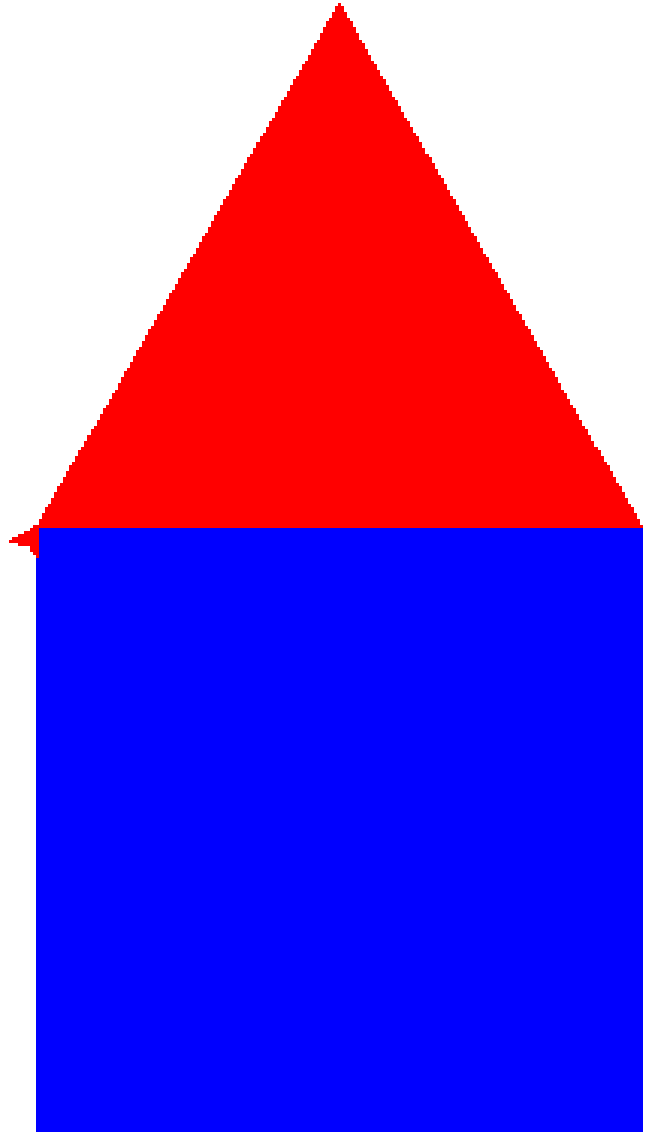
Try the Following Code

```
color("green")
begin_fill()
for i in range(5):
  forward(100)
  right(72)
end_fill()
```

Can do this directly in interactive loop or in a file

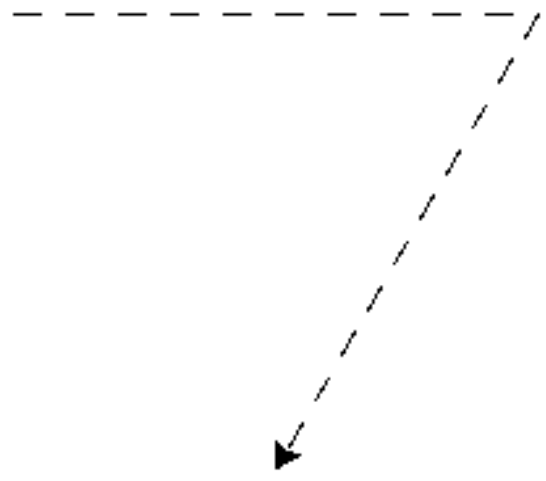
Exercise: The Pretty House

Add `begin_fill()` and `end_fill()` to your code to produce the pretty house at the right



Pen goes up, Pen goes down

- `penup()` stops drawing lines, allows turtle to move without drawing
- `pendown()` starts drawing lines again
- Useful for dashes and for `face.py`



```
for i in range(10):
  forward(10)
  penup()
  forward(10)
  pendown()

right(120)
for i in range(10):
  forward(10)
  penup()
  forward(10)
  pendown()
```

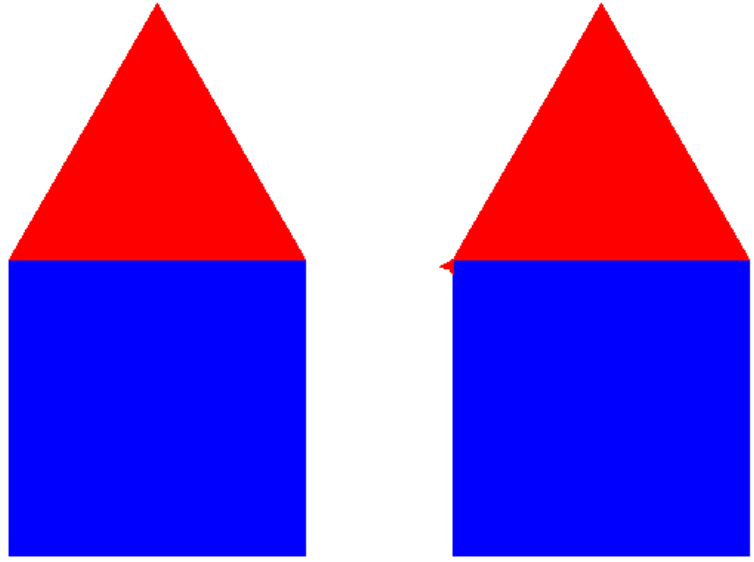
Exercise: Two Houses

Single House

```
# Draw the body of the house
color("blue")
begin_fill()
for i in range(4):
  forward(200)
  right(90)
end_fill()

# Draw the roof of the house
color("red")
begin_fill()
right(300)
for i in range(3):
  forward(200)
  right(120)
end_fill()
```

Now `penup()`, change angle, move, `pendown()` and do it again



Variables

- A name like `size` associated with a value
- Can change the value associated with the name with assignment

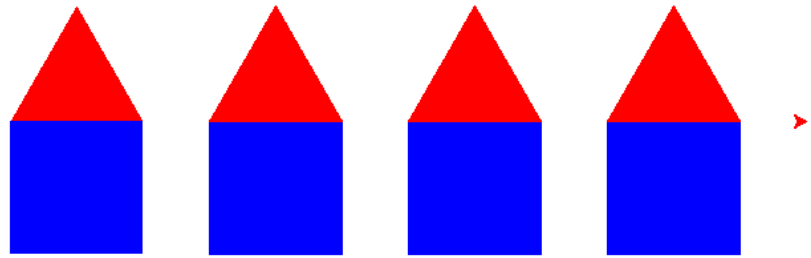
```
# size is 100
size = 100
# change size to 200
size = 200
# value of i is 3
i = 3
# change size to 300
size = i * 100
```

```
# Little square
size = 100
for i in range(4):
  forward(size)
  right(90)

# Big square
size = 200
for i in range(4):
  forward(size)
  right(90)
```

Exercise: The Suburbs

- Smaller houses - size 100 sides
- Use a variable `size = 100`
- Change `forward(200)` to `forward(size)`
- Use a `for` loop to repeatedly draw houses and move turtle

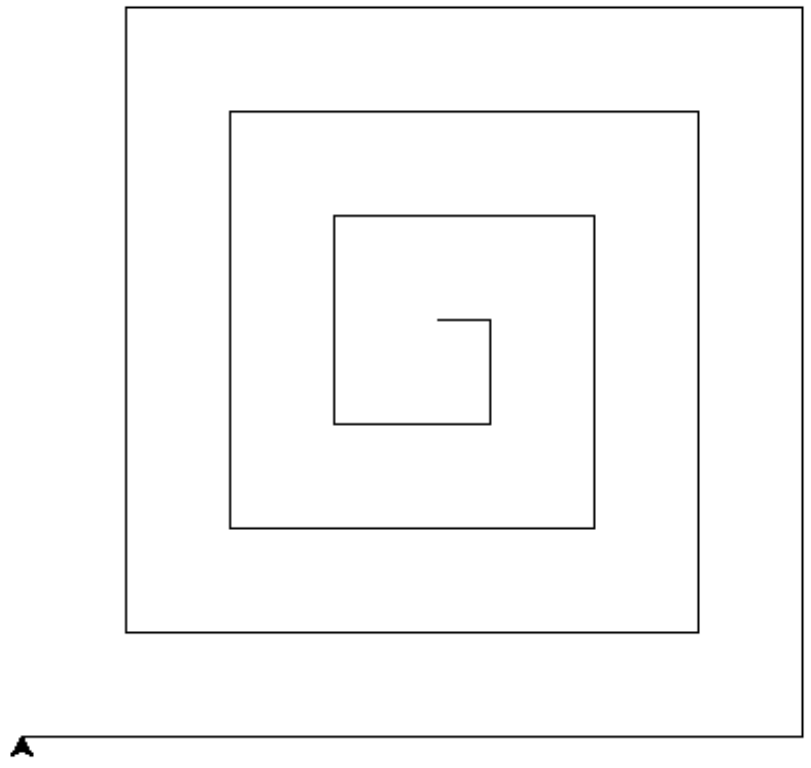


Loop Variables Change Each iteration

The `range(N)` statement produces a sequence of numbers from `0` to `N`; good for loops

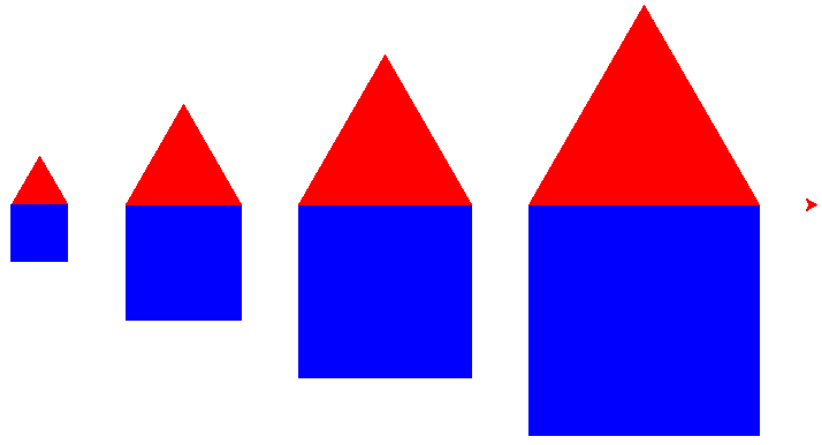
```
# prints 0, 1, 2, 3
for i in range(4):
    print(i)

# Square spiral
size = 0
for i in range(15):
  size = (i+1) * 25
  forward(size)
  right(90)
```



Exercise: Suburbs part 2

- Change `size` each loop iteration
- Remember that loop variables start at 0



```
for i in range(4):
    print(i)
# prints 0, 1, 2, 3
```

Next Time

- Read "Think": Functions, Selection, Iteration
- Finish "Pattern" Ch 4
- Start on HW 3