Python, Part 2

CS 8: Introduction to Computer Science
Lecture #4

Ziad Matni
Dept. of Computer Science, UCSB

A Word About Registration for CS8

- This class is currently **FULL**
- The waitlist is **CLOSED**

Lecture Outline

- Numbers and Arithmetic in Python
- Variables in Python
- The Python Interpreter
  - Using Python IDLE tool for demos/labs
- Modules
- Functions

Yellow Band = Class Demonstration! ☺

A Function To Draw A Square

- Part of listing 1.2 from the text (p. 30)

```
def drawSquare(myTurtle, sideLength):
    myTurtle.forward(sideLength)
    myTurtle.right(90)   # side 1
    …
```

- Then to invoke it for drawing a square that has 20 pixels on each side using a turtle named `t`:

```
>>> drawSquare(t, 20)
```

- What might happen if we invoked `drawSquare(20, t)`?

Let's try it out!

Importing From A Module

- Imagine the **`drawSquare`** function is in a file called **`ds.py`**
- We have two basic choices to use this function:

1. Import whole module, and specify module to use

```
>>> import ds
>>> ds.drawSquare(t, 20)
```

2. Import part(s) of module, then just use the part(s)

```
>>> from ds import drawSquare
>>> drawSquare(t, 20)
```

"sys" is a standard module and "path" is one of its objects that stores the directory paths where your Python files will be

- Of course, Python must know where **`ds.py`** is on the computer!
  - Easy solution: store it in current directory or along **`sys.path`**

- Or in Python IDLE: *File* → *Open* – no need to import

Controlling the Flow of a Program

- Programs will often need to make decisions on what to continue doing
  - Like coming to a fork in the road…

- We present the algorithm/program with a *conditional statement* (if-then-else)
  - e.g. **if** (calories > 1800) **then** stop_eating( )
    else keep_eating( )

If-Else in Python

- The syntax in Python is:

```
if conditional_statement :
   statement 1
   statement 2
   …
else:
   else-statement 1
   else-statement 2
   …
```

Let's try it out!

More on Conditional Statements

- Conditional statements follow Boolean logic
  - That is, they are either TRUE or FALSE

- Often we use comparisons, like "equal to" or "greater than or equal to"
  - Like in math…

Meaning	Math Symbol	Python Symbols
Less than	<	<
Greater than	>	>
Less than or equal	≤	<=
Greater than or equal	≥	>=
Equals	=	==
Not equal	≠	!=

Boolean Logic Operators

- Other than comparison operations, we can perform Boolean logic operations

- Logic AND (and)
  - True if *all* of the conditions are True
- Logic OR (or)
  - True if *any* of the conditions is True
- Logic NOT (not)
  - True if the condition is False
  - False if the condition is True

More on If-Else in Python

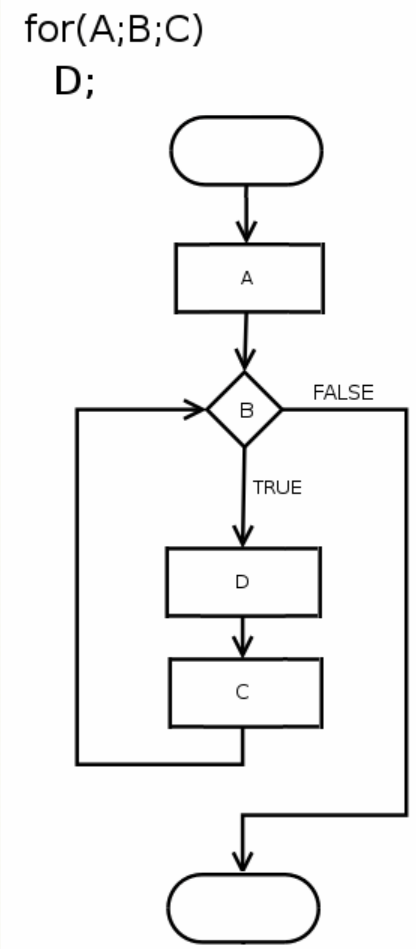
- You can create multiple "else" blocks, like this:

```
if conditional_statement A :
    statement A1
    statement A2
    …
elif conditional_statement B :
    statement B1
    statement B2
    …
else:
    else-statement 1
    else-statement 2
    …
```

Loops

- Sometimes we want to be able to repeat a part of the program a certain number of times
  - Called a "loop"

- A popular way to do this is with the **`for`** command.



Repetition with a `for` loop

- `for` *ref* `in` *a list*`:`
  `#` *block – ref refers to current object in list*
  - `for`, `in`, `:` – mandatory parts
  - *ref* – a name for referring to objects in the list

- Example:

```
for numbers in (1, 2, 3, 4, 5):
    print (numbers)
```

  This will print out the numbers 1 thru 5 in sequence

Using `range` with `for` loops

- The `range` function provides a handy list
- Simplest use: **`range(n)`** – a list with **`n`** items **`[0,1,…n-1]`**

- Example:

```
for numbers in range(5):
    print (numbers)
```

  This will print out the numbers 0 thru 4 in sequence

More `range` with `for` loops!

- You can also do a range with start & stop parameters.
- Example:

```
for numbers in range(5, 8):
    print (numbers)
```

This will print out the numbers 5 thru 7 (excludes 8) in sequence

- Or you can have start, stop and step parameters.
- Example:

```
for i in range(1, 11, 4):
    print(i)
```

This will print out the numbers 1, then 5, then 9

Let's try these out!

Simpler drawing by repetition

- Listing 1.3 from the text (p. 34)

```
def drawSquare2(myTurtle, sideLength):
    for i in range(4):
        myTurtle.forward(sideLength)
        myTurtle.right(90)
```

- Small variation draws a spiral (Listing 1.4)

```
def drawSpiral(myTurtle, maxSide):
    for sideLength in range(1, maxSide+1, 5):
        myTurtle.forward(sideLength)
        myTurtle.right(90)
```

More drawing abstraction

- Contrast – a triangle vs. a square (Listing 1.5)

```
def drawTriangle(myTurtle, sideLength):
    for i in range(3): # draw 3 sides, not 4
        myTurtle.forward(sideLength)
        myTurtle.right(120) # 120°× 3
```

- Hmm…any regular polygon? (Listing 1.6, p. 38)

```
def drawPolygon(myTurtle,sideLength,numSides):
    turnAngle = 360 / numSides
    for i in range(numSides):
        myTurtle.forward(sideLength)
        myTurtle.right(turnAngle)
```

Let's try these out!

Problem solving: Draw a circle with a given radius

- *Notice: a polygon with many sides looks like a circle*
  - But how many sides to draw?
  - And how long should each side be?
- *Start simple*: decide to draw 360 sides every time
- *Think*: length of 1 side = circumference / 360
- And remember from math that circumference equals 2πr
- Put it all together: Listing 1.7 from the text (p. 40)

```
def drawCircle(myTurtle, radius):
    circumference = 2 * 3.1415 * radius
    sideLength = circumference / 360
    drawPolygon(myTurtle, sideLength, 360)
```

Let's try it!

YOUR TO-DOs

- ❑ Read **Chapter 2**
- ❑ Finish **Homework2** (due **Thursday 4/20**)
- ❑ Prepare for **Lab2**

- ❑ Brush your teeth after every meal

</LECTURE>