Introduction to Python Programming
CpSc 4810/6810
Clemson University
May 12, 2015
Syllabus

1 Course Overview

1. Programming environments
2. Using Python to write programs
3. Using a Graphical User Interface (GUI)
4. Scripting
5. Animations and video games

2 Specific Topics

In this course, I will assume that you:

1. Know how to read,
2. know how to turn a computer on, and
3. know how to use a web browser.

If you are unfamiliar with the above three skills than you are in the wrong course. I will also assume that you have never written a computer program and that you are unfamiliar with any computer language.

In the following sections, we list specific topics that we will cover in this course about developing skills for programming a computer. But to further illustrate the overview, we will be using the following technologies:

1. An editor
2. The Python programming language
3. Piazza
4. A media player
5. PDF reader
6. Web cam
7. Tkinter
8. handin

3 Programming in Python

1. Why Python?
2. Developing a Python "bag of tricks"
3. Using Python for scripting

3.1 Python Basics

1. Assignment
2. input (raw_input)
3. output (print),
4. data types: int and float
5. Some sample problems using assignment:
   (a) Read two numbers, print them backwards;
   (b) Read two numbers, print them backwards on the same line of output;
   (c) Read three numbers, print them backwards;
   (d) Read a number, print it's square;
   (e) Read the two legs of a right triangle, print the hypotenuse;
   (f) Read the miles and number of gallons consumed, print average miles per gallon;
   (g) The ratio of an object's weight on Mars and Jupitor is 0.38 and 2.64 respectively, read a person's weight and print their weight on each of the planets;
   (h) Read a fahrenheit temperature, print it's centigrade equivalent.
   (i) Read two numbers and swap them.
   (j) Read a 2 digit number and print each of its digits.
   (k) Read a 3 digit number and print each of its digits.

3.2 Interpreter/Files

1. Python can be used interactively from the command line; this modality empowers fast testing of concepts, constructs, behaviors, and help facilities. But Python can also accept programs as file input, which requires the use of an editor. I propose vim, or it's graphic counterpart gvim, which facilitates rapid search, update, and modification of text. For example:

```
$ python
Python 2.6.6 (r266:84292, Sep 15 2010, 15:52:39)
[GCC 4.4.5] on linux2
Type "help" for more information.
>>> import random
>>> help(random)
```

3.3 Decision: if, else, elif

1. Read an integer, print a message indicating if it's even or odd.
2. Read an integer, print a message if it's a perfect square.
3. Read an integer grade value and print out the corresponding letter value.
4. Read two floating point numbers representing miles traveled and gallons consumed. Print the average miles per gallon. However, if the gal-

lons is zero, print the miles as the average miles per gallon.
5. Read three numbers and print largest. Print the smallest.
6. Print the largest of 4 numbers. The smallest.
7. Read a number and print a message indicating it's evenly divisible by 3.

3.4 Loops: while

1. Print the perfect squares from 1 to 144; i.e., 1, 4, 9, 16, . . .
2. Read a sequence of numbers from a user until an end sentinel of -1 is reached. Print the average of the numbers.
3. Read a number n representing the number of numbers to follow. Find and print the largest of the numbers.
4. Read a list of numbers followed by an *end sentinel*, where the *end sentinel* is the letter z. Find and print the largest of the numbers.
5. For the previous problem, print the largest number and the number of numbers read.
6. An Arabian chieftain called upon a neighboring chieftain to help him in a feud. The neighboring chief agreed, provided that the feuding chieftain would place pennies on a checkerboard as follows: place one penny the first day, two pennies the second day, four pennies the third day, and so forth. Doubling the pennies each day. How much money did the feuding chief have to pay the neighboring chief?
7. Create two lists in two columns: the first column should list the numbers from 1 to 20, and the second column should list the corresponding factorial.
8. Create two lists in two columns: the first column should list the numbers from 1 to 20, and the second column should list the corresponding Fibonacci number.
9. Write a program that will pick a random number from 1 to 100 and then ask the user to guess the number. Count the number of trials required by the user to guess the number.
10. Write a program that will find and print the first 100 prime numbers.
11. Write a program that will find and print the prime numbers from 1 to 200.
12. Write a program that will read an integer from the user and then convert and print that integer as a binary number.
13. Write a program that will print a Christmas tree of asterisks.

3.5 Loops: for

3.6 Functions

1. What are they:
   (a) A named block of code.
   (b) Doesn't execute until you call it's name
   (c) Can accept parameters
   (d) Can return a value.
   (e) Doesn't execute if you don't call it
2. Examples:
   (a) square(n)
   (b) display()
   (c) isEven(n)
   (d) isPrime(n)

3.7 Strings

1. An array of characters
2. **zero** indexed
3. len
4. String functions:
   (a) string.find()
   (b) string.split()
   (c) string.join()
   (d) string.strip()
   (e) string.upper()
   (f) string.digits
   (g) string.letters

3.8 User Interfaces: Tkinter

3.9 Graphics: Turtle

3.10 Animations and Video Games: PyGame