Introduction to Python

Curtis Meadow
School of Computing and Information Technology
University of Maine
meadow@maine.edu

Outline

- Python Versions
- About Python
- Free Books
- Types of Errors
- Let's Start Learning Python
  - Hello World, Pleased to meet you
  - Python object types, variables and names
  - Linear programs and ASCII art
  - Multi-line Strings

About Python

About Python

- Python is a remarkably expressive dynamic programming language that is used in a wide variety of application domains. Python is often compared to Perl, Ruby, PHP, Tcl, Scheme or Java.
- Python runs everywhere – Windows, IOS, Linux, Android …
- Python is friendly... and easy to learn
  - But watch out – like many "easy to learn" languages it also makes it easy to "shoot yourself in the foot"
- Python is open source

Expressiveness

- Although languages are equal in power, some are more expressive and more efficient (both for humans and machines) for particular domains
- Example:
  - You would not write a Python program to control a space craft that is going to land on Mars. You would most likely use C and/or C++
  - You would not use C/C++ to create a website that would track the space craft in real time. Python would be a good choice however.

One Area where Python Excels…

- Certain language features make Python well suited for Bioinformatics
- "Bioinformatics is conceptualizing biology in terms of macromolecules (in the sense of physical-chemistry) and then applying "informatics" techniques (derived from disciplines such as applied maths, computer science, and statistics) to understand and organize the information associated with these molecules, on a large-scale."
- http://www.ncbi.nlm.nih.gov/pubmed/11552348

Where is Python used?

- Web and Internet development
- Scientific computing (especially bioinformatics)
- Education
- Desktop GUIs
- System administration scripts
- Games and 3D Graphics
- See https://www.python.org/about/apps/

Who Uses Python?

- http://www.python.org/about/quotes/
- Python is used successfully in thousands of real-world business applications around the world, including many large and mission critical systems.
- Well-known users include YouTube, Google, Industrial Light & Magic and many others

Introduction to Python

- Go to www.python.org for extensive documentation and downloads of Python
- Many tutorials and books are available
- Beginner to expert level

Python Versions

- The Python language now is generally available in versions 2.x or 3.x
- Until recently there was a great deal of software that would not run on 3.x
  - Apple as of OS X 10.13 still ships with v2.7
  - Many Linux distros include 2.7
  - Windows does not pre-install Python
- In COS 125 Intro to Programming and Problem Solving we used 2.7 until the course changed to Java in Fall 2017.

Why all of the fuss?

- Guido van Rossum, the original designer of Python, wanted to fix some design flaws in the language
- In a very unusual move, backwards compatibility was abandoned
  - The first Python 3 was released in 2008
- Almost all v2.x software would have to be rewritten

So which to use?

- From PYTHON 2 VS PYTHON 3: IT'S DIFFERENT THIS TIME

https://www.activestate.com/blog/2017/01/python-3-vs-python-2-its-different-time

- … reflecting on where we are in 2017, the landscape has changed in favour of Python 3. Examples of this shift can be seen in some of the larger projects, like TensorFlow and Thrift, both adding Python 3 support in the past 12 months.
- Of the top 360 packages listed on http://py3readiness.org/, 95% are Python 3 compatible.
- Python 2.7, the latest in the 2 series, will only have bug and security fixes going forwards. … Remember the 2.7 series first came out in 2010!

V2-3 Differences at this level

- We will encounter only four differences:
  1. Different **`print`** syntax

     V3: `print ("99 bottles of beer on the wall.")`
     V2: `print "99 bottles of beer on the wall."`
  2. Integer Division

     V3:
     ```
     >>> 9/7
     1.2857142857142858
     ```
     V2:
     ```
     >>> 9/7
     1
     ```

V2-3 Differences

3. raw_input is now input

   V3: `Name = input("What is your name? ")`
   V2: `Name = raw_input("What is your name? ")`

4. range() in v3 is not a list

```
V3:     >>> range(1,6)
        range(1, 6)
V2:     >>> range(1,6)
        [1, 2, 3, 4, 5]
```

Free Textbooks!

- *Think Python* by Allen Downey.
  - https://greenteapress.com/wp/think-python/
  - http://greenteapress.com/wp/think-python-2e/
- The subtitle of this book is "How to Think Like a Computer Scientist"
- This book was originally written for Java, but …

  "Jeff Elkner, a high school teacher in Virginia, adopted my book and translated it into Python. He sent me a copy of his translation, and I had the unusual experience of learning Python by reading my own book. "

Books by Al Sweigart

- *Invent Your Own Computer Games with Python* by Al Sweigart
  - https://inventwithpython.com/
- "Who is this book for?
  - Complete beginners who want to teach themselves programming, even if they have no previous experience.
  - Kids and teenagers who want to learn programming by creating games.
  - Adults and teachers who wish to teach others programming.
  - Anyone, young or old, who wants to learn how to program by learning a professional programming language."

Books by Al Sweigart

- *Invent Your Own Computer Games with Python* can be read online for free in the 4th edition
  - Earlier editions can be downloaded in PDF
  - All can be purchased in print at amazon.com
- A note about Al Sweigart's code:
  - It is quite different in approach and style from the approach and style we taught in COS 125
  - In particular, it is nearly all "procedural" code while we endeavored to teach Object-Oriented Programming (OOP)
  - OOP is almost universal in modern code
- But he explains things well for absolute beginners!

Other Books by Al Sweigart

- *Automate the Boring Stuff with Python*
  - "For anyone who uses a computer how to write small, practical programs to automate tasks on their computer."
- *Cracking Codes with Python*
- *Making Games with Python & Pygame*

John Zelle

- Python Programming by John Zelle
  - Another traditional computer science textbook
  - Useful for getting a different perspective / explanation. For Python 3:
  - https://www.codewithc.com/python-programming-an-introduction-to-computer-science-pdf/
- For Python 2:
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.6062&rep=rep1&type=pdf

Mark Pilgrim

- *Dive Into Python* by Mark Pilgrim
  - This book is for experienced programmers who want to learn Python
  - It covers advanced topics as well as the basics
- For Python 3
  http://www.diveintopython3.net/
- For Python2
  http://www.oswego.edu/~kanbur/IRES2008/python.pdf

What we will do here…

- We will focus on the language itself, with the goal that you can understand a program that uses selection (if-then-else), iteration (loops) as well as sequential execution
- We will NOT cover installation
- We will discuss only a few details of using IDLE
  - Idle is a simple integrated development environment (IDE) that is included with Python and is written in Python

Errors

- Regardless of language, there are three broad types of errors that can occur

1. Syntax Errors
2. Runtime Errors
3. Semantic or Logic Errors

- Writing code is fairly easy. Writing code that works all of the time is not.

Types of Errors

- Syntax Errors
  - You wrote something incorrectly – usually caught when statement is executed, but sometimes caught before then by the text editor
- Runtime Errors
  - These occur when the program is running. Examples: division by 0; some data not being present when expected; data in an unexpected form …

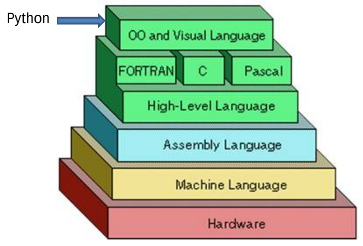
Types of Errors

- Semantic or Logic Errors
  - Your solution is just not correct – you left something out or had an incorrect procedure for figuring it out
  - This type of error is often resolved by debugging
- "I hate this computer. It never does what I want it do. It only does what I tell it to do."

Let's Start Learning Python

- Python is a high level programming language
  - What does this mean?
- It means that we "abstract away" the hardware

Language Levels



From http://cset.sp.utoledo.edu/sample/engt1050/engt1050_languages.html

The First Program – "Hello World"

- **`print ("Hello, World!")`**
- print means display on the screen – does not print on a printer
- You can also write
- **`print ('Hello, World')`**
- What's the difference?
- You can't write: **`print ('Hello, World!")`**

Getting Started

- All programming projects start with a goal
- Usually, it is a problem that needs to be solved
- In this course, we focus on games because they are relatively easy to explain and at the same time provide many challenging problems
- We will start solving the problem in English

Types of Objects in Python

- It is important to distinguish between different types of objects
- Python has strings, numbers, lists and a variety of other objects
- For example, "3" and 3, 3.0 and 3+0j are different objects in Python
  - You can convert between them, but they behave differently

What Does Python Display?

- **`print (2 + 2)`**
- **`print ('2'+'2')`**
- **`print ('2'+"2")`**
- **`print ('2+2')`**
- **`print ('2'*3)`**
- **`print ("2*3")`**
- **`print (2*3)`**
- **`print (2*3.0)`**
- **`print (2*3+0j)`**
- **`print ('2'*'3')`**

Variables

- Python permits you to store values in variables
- You can write expressions such as **`X = 2`**
- **`Pi = 3.14159`**
- **`pi = 3.14159`**
- Python is case sensitive so **`Pi,pi, pI,`** and **`PI`** are 4 different names
- Special names begin with _ or even _ _

Variable Names

- Variable names can be any length and made from the characters 'A'..'Z', 'a'..'z', 0'..'9' and '_'
- They may not begin with a digit ('0'…'9')
- Usually, Python variable names begin with a lower case letter

  `newValue`

  `oldValue`
- This naming convention is called camel case

Which Names are Legal?

`HelloWorld`
- Legal

`Hello-World`
- Not legal – because – is not allowed
- **`Hello>World`**
- Not legal – > not allowed
- **`Hello3World`**
- Legal

`3HelloWorld`
- Not legal – can't start with digit
- **`_HelloWorld`**
- Legal
- **`helloWorld`**
- Legal
- **`h4ll0W0rld`**
- Legal

Another Program

```
print ("Hello, World!")
Name = input("What is your name? ")
print ("It is good to meet you, " + Name)
```

```
>>>
Hello, World!
What is your name? CM
It is good to meet you, CM
>>>
```

Note space between ? and " in second line.

Linear Programs

- We will start out with programs that look like:
  - Statement 1
  - Statement 2
  - Statement 3
  - …
- The computer just executes one instruction after another
- Easy place to start, but of limited usefulness
- We call them **linear** because execution proceeds directly through the statements

What Else is There?

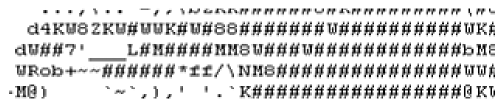
- Computer languages provide only three basic control structures:
  1. Linear execution
  2. Selection (If-Then-Else)
  3. Iteration (Looping)
- Any language that has these three abilities is called Turing-complete and can compute anything that is computable
- No language is more “powerful” than any other language

ASCII Art

- A class of interesting linear programs are drawing programs of a type called ASCII Art
  - ASCII (pronounced As-Key) is the American Standard Code for Information Interchange
  - Quite a mouthful and a long story
- We start with an exercise

Question

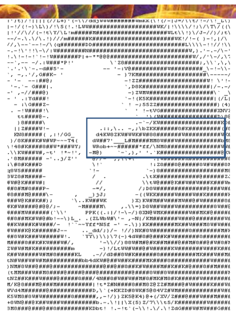
- What is the following?



```
 d4KW8ZKW#WWK#W#88#######W###########WK
dW##7'____L#M#####MM8W####W###############bM8
WRob+~~#######*ff/\NM8####################WW
-M@)      `~`,),'  '.`K####################@KW
```

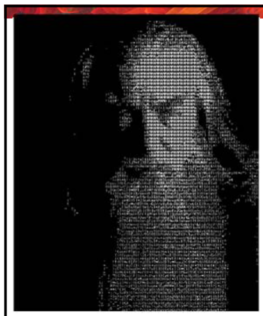
Answer

- It's Mona Lisa's eye, of course!



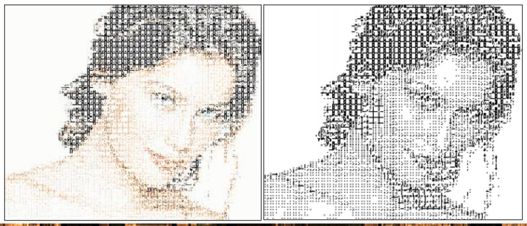
ASCII Art

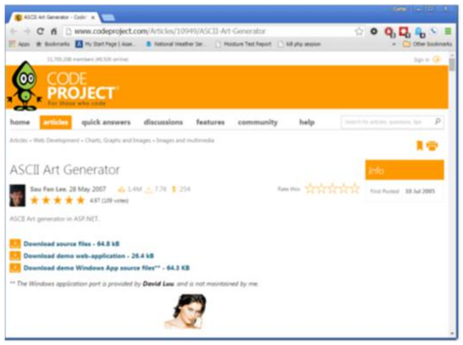
- These are drawings made from characters (you will learn later why we call them ASCII characters)
- In fact, most graphics produced on a computer or on a TV screen are made up of dots of different shades or colors
- Here are some more examples



For an interesting application (called steganography) see

http://www.labnol.org/internet/favorites/hide-your-secret-document-inside-ascii-art/6500/




www.codeproject.com/Articles/10949/ASCII-Art-Generator

ASCII Art Generator

Sau Fan Lee, 28 May 2007

ASCII Art generator in ASP.NET.

Download source files - 64.8 kB
Download demo web-application - 26.4 kB
Download demo Windows App source files** - 64.3 KB

** The Windows application port is provided by David Luu and is not maintained by me.

ASCII Art Generators

- Above from the Code Project

http://www.codeproject.com/Articles/10949/ASCII-Art-Generator

- Also see (for Python)
- https://www.hackerearth.com/notes/beautiful-python-a-simple-ascii-art-generator-from-images/
- Only about 50 lines of Python code - plus PIL (Python Imaging Library)

Simple Drawings

- As you can see there are some impressive drawings that can be done with ASCII art
- Let's see how to do some simple drawings

A Problem

- How can we get the computer to draw the picture that is shown to the right?
- What commands do you know?
- How can you string them together

```
     ******
    ********
   **********
  ************
 **************
****************
****************
**  **    **  **
**  **    **  **
******    ******
******    ******
******    ******
******    ******
******    ******
```

First Step in Modeling a Solution

```
print line 1
print line 2
print line 3
print line 4
print line 5
print line 6
print line 7
print line 8
print line 9
print line 10
print line 11
print line 12
print line 13
print line 14
```

- We will use incremental development
- Model a solution in English, then develop it in steps

Version 2

```
##print line 1
##print line 2
##print line 3
##print line 4
##print line 5
##print line 6
##print line 7
##print line 8
##print line 9
##print line 10
##print line 11
##print line 12
##print line 13
##print line 14
print
print "DONE!"
```

Comment character

Analyzing the Problem

```
1.   5b + 6* + 5b                          ******
2.   4b + 8* + 4b                         ********
3.   3b +10*+3b                          **********
4.   2b +12*+2b                         ************
5.   b + 14* + b                       **************
6.   16*                              ****************
7.   16*                              ****************
8.   2* + 2b + 2* + 4b + 2* + 2b + 2* **  **    **  **
9.   2* + 2b + 2* + 4b + 2* + 2b + 2* **  **    **  **
10.  6*+4b+6*                         ******    ******
11.  6*+4b+6*                         ******    ******
12.  6*+4b+6*                         ******    ******
13.  6*+4b+6*                         ******    ******
14.  6*+4b+6*                         ******    ******
```

Version 3

```
print 5*" "+6*"*"+5*" "                                          ******
print 4*" "+8*"*"+4*" "                                         ********
print 3*" "+10*"*"+3*" "                                       **********
print 2*" "+12*"*"+2*" "                                      ************
print 1*" "+14*"*"+1*" "                                     **************
print 16*"*"                                                ****************
print 16*"*"                                                ****************
print 2*"*" + 2*" "+2*"*"+4*" "+2*"*"+2*" "+2*"*"           **  **    **  **
print 2*"*" + 2*" "+2*"*"+4*" "+2*"*"+2*" "+2*"*"           **  **    **  **
print 6*"*"+4*" "+6*"*"                                     ******    ******
print 6*"*"+4*" "+6*"*"                                     ******    ******
print 6*"*"+4*" "+6*"*"                                     ******    ******
print 6*"*"+4*" "+6*"*"                                     ******    ******
print 6*"*"+4*" "+6*"*"                                     ******    ******
print
print "DONE!"
```

Version 4

```
for i in range(5,-1,-1):                                 ******
    print (i*" "+(16-2*i)*"*" + i*" ")                  ********
    print (16*"*")                                     **********
    win = 2*"*" + 2*" " + 2*"*"                       ************
    for i in range(2):                               **************
        print (win+ 4*" "+ win)                     ****************
    for i in range(5):                              ****************
        print (6*"*"+4*" "+6*"*")                   **  **    **  **
    print ("")                                      **  **    **  **
    print ("DONE!")                                 ******    ******
                                                    ******    ******
                                                    ******    ******
                                                    ******    ******
                                                    ******    ******
```

```
i = 5, 4, 3, 2, 1, 0

5b+6*+5b
4b+8*+4b
3b+10*+3b
```

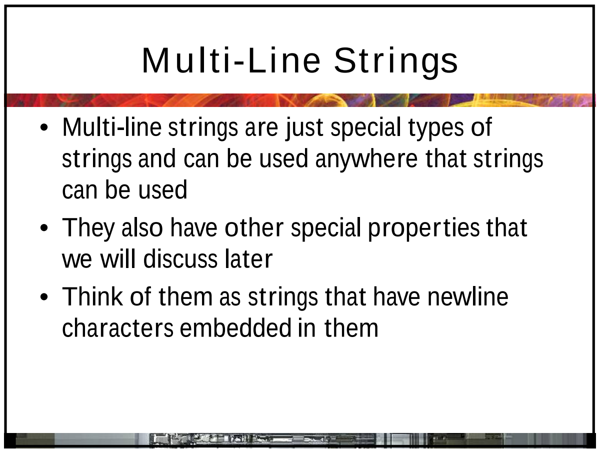
Version 5

```
def drawHouse():
    for i in range(5,-1,-1):
        print i*" "+(16-2*i)*"*" + i*" "
    print 16*"*"
    win = 2*"*" + 2*" " + 2*"*"
    for i in range(2):
        print win+ 4*" "+ win
    for i in range(5):
        print 6*"*"+4*" "+6*"*"
    print
    print "DONE!"

for i in range(3):
    drawHouse()
```

Python – The Swiss Army Knife of Programming

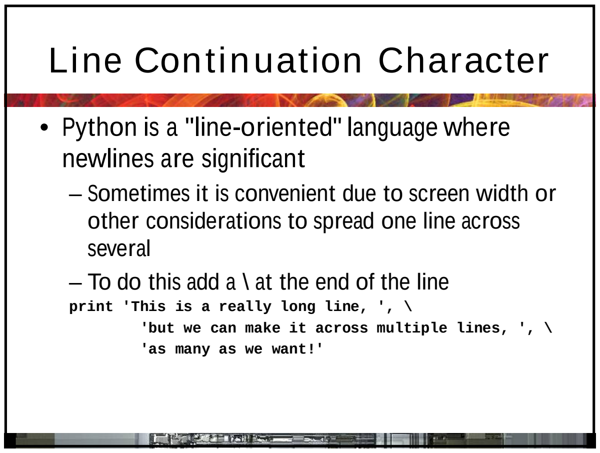
- There are often many ways to do something in Python
- Python has a very handy feature called ***multi-line strings***
- These are strings that run over multiple lines
- To create a multi-line string just use three quotes (single or double) consecutively
- The string ends with a sequence of three quotes

```
J = '''     ******
    ********
   **********
  ************
 **************
****************
****************
**  **    **  **
**  **    **  **
******    ******
******    ******
******    ******
******    ******
******    ******'''

print (J)
```



Multi-Line Strings

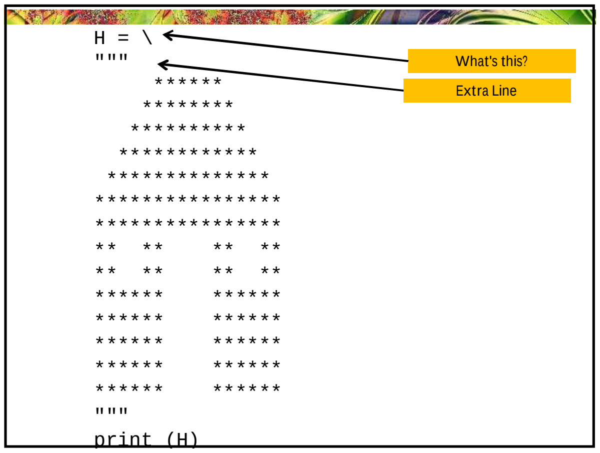
- Multi-line strings are just special types of strings and can be used anywhere that strings can be used
- They also have other special properties that we will discuss later
- Think of them as strings that have newline characters embedded in them



Line Continuation Character

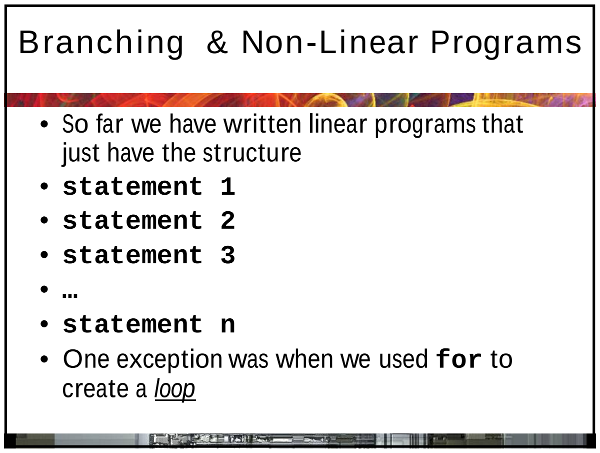
- Python is a "line-oriented" language where newlines are significant
  - Sometimes it is convenient due to screen width or other considerations to spread one line across several
  - To do this add a \ at the end of the line

```
print 'This is a really long line, ', \
        'but we can make it across multiple lines, ', \
        'as many as we want!'
```



```
H = \
"""
     ******
    ********
   **********
  ************
 **************
****************
****************
**  **    **  **
**  **    **  **
******    ******
******    ******
******    ******
******    ******
******    ******
"""
print (H)
```

What's this?
Extra Line



Branching & Non-Linear Programs

- So far we have written linear programs that just have the structure
- **`statement 1`**
- **`statement 2`**
- **`statement 3`**
- …
- **`statement n`**
- One exception was when we used **`for`** to create a *loop*