Introduction to Python

L435/L555

Dept. of Linguistics, Indiana University

Fall 2016

A tour of Python

Today we're going to take a quick tour of Python.

- You'll see many different types of things, but nothing in-depth
- By the end, you should be able to:
  - Have some sense of what Python has to offer
  - Run some basic Python commands interactively
  - Write short Python programs and run them

For most of what we're doing: type `python3` in a terminal, to open the interactive shell

Hello World

```
print('Hello world.')
```

```
print('Hello world.')
print(4 + 5)
```

(http://en.wikibooks.org/wiki/Non-Programmer27s_Tutorial_for_Python_3/Hello,_World)

Data types

Data **types** are the building blocks from which everything else is built

- Simple Types: numbers, strings (**later:** booleans, bytes)
  - numbers: 3, 12.443, 89, ...
  - strings: ”hello”, ’manny’, ”34”, ...
- Complex Types: lists, dictionaries (**later:** sets, tuples)
  - lists: [1,2,3], [1,2,”a”], [”john”, ”mama”, ”denny”, ”michelle”], ...
  - dictionaries: {”a”:1, ”b”:16}, ...

Python is **dynamically typed**: you do not have to declare what type each variable is
(http://www.diveintopython3.net/native-datatypes.html)

Numbers

Some quick examples:

```
>>> 2+2
4
>>> 3/2
1.5
>>> 3//2
1
```

Python has integers and floating point numbers (& complex numbers), and operations to convert between them:

```
>>> float(3)
3.0
>>> int(4.123)
4
```

Variables

What is a variable?

Definition

A variable is a name that refers to some value (could be a number, a string, a list etc.)

Example

1. Store the value 42 in a variable named *foo*
   ```
   foo = 42
   ```
2. Store the value of foo+10 in a variable named bar
   ```
   bar = foo + 10
   ```

User input

Example

1. Ask the user to input a name, and store it in the variable *name*

   ```
   name = input('enter a name:  ')
   ```

2. Create a new string with a greeting

   ```
   greet = 'hello ' + name
   ```

3. Print the greeting

   ```
   print(greet)
   print('hello', name)
   ```

http://en.wikibooks.org/wiki/Non-Programmer's_Tutorial_for_Python_3/Who_Goes_There?

⇒ Let's look at `area.py`

User input

Example

1. Ask the user to input a number, and store it in the variable *foo*

```
foo = int(input('enter an integer:  '))
bar = float(input('enter any number:  '))
```

2. Add foo and bar together

```
foo + bar
```

3. Calculate the average of *foo* and *bar*, and save it in a variable named *avg*

```
avg = (foo + bar)/2
```

String basics

- Strings must be enclosed in quotes (double or single)
- Strings can be concatenated using the + operator
  - Note that this is the same as numerical addition ...
  - But you cannot combine a string and a number (common error)

(http://www.diveintopython3.net/strings.html)

Strings

- Many ways to write a string:
  - single quotes: `'string'`
  - double quotes: `"string"`
  - can also use `"""` to write strings over multiple lines:

```
>>> s="""<html>
... example
... </html>
... """
>>> s
'<html>\nexample\n</html>\n'
```

- There are string characters with special meaning: e.g., `\n` (newline) and `\t` (tab)
- Get the length of a string by the `len` function

String indices & slices

You can use slices to get a part of a string (or other sequence)

```
>>> s = "happy"
>>> len(s)  # use the len function
5
>>> s[3]    # indexed from 0, so 4th character
'p'
>>> s[1:3]  # characters 1 and 2
'ap'
>>> s[:3]   # first 3 characters
'hap'
>>> s[3:]   # everything except first 3 characters
'py'
>>> s[-4]   # 4th character from the back
'a'
```

Functions

What is a function?

Definition

A function is like a mini-program. It can take several *arguments*, and *returns* a value.

We won't look at the syntax of these yet.

Modules

What is a module?

Definition

Python is easily *extensible*. Users can write programs that extend the basic functionality, and these programs can be used by other programs, by loading them as a *module*

Example

1. Load the math module
   `import math`
2. Round quotient of foo and bar down to nearest integer
   `math.floor(foo/bar)`
3. Get the value of pi
   `math.pi`

(https://docs.python.org/3.2/library/math.html)

Sidebar: Anaconda Python

There are built-in modules (e.g., `math`, `re`)

- ... and then there are third-party packages which extend Python in numerous ways

Some people like to install Anaconda Python

- This is Python + 720 other packages (including NLTK, NumPy, SciPy, etc.)
- https://www.continuum.io/downloads

Note: the package is quite large

- There is the `miniconda` option: http://conda.pydata.org/miniconda.html

Saving and executing programs

Example

- Script File: `hello.py`

```
# this prints 'hello world' to stdout
print("hello world")

# note how # denotes comments!
# (i.e., ignored by python interpreter)
```

- Run the program:

```
python3 hello.py
```

Creating & Editing Python files

Python files are simply text files, so we just need a text editor. Some options:

- Windows: Notepad(++)/Wordpad → Save as plain text
  - Sometimes Windows is set up s.t. it forces you to add a `.txt` extension to your file.
  - This isn't a problem, but to get rid of it, (I think) you need to save as "All files" and also change your desktop settings so that they show file extensions
- Mac/Unix: nano, Emacs (or Aquamacs [which I use]), VIM, TextWrangler, and probably others
  - I use Aquamacs, but will likely use TextWrangler & IDLE (next slide) in class, but use what you like ...
  - Note: Emacs & VIM provide a lot of Python support

IDLE

Some text editors offer **syntax highlighting**, which shows variable names, indentation, etc.

Integrated Development Environments (IDEs) offer syntax highlighting, debugging features, streamlined code-running, etc.

- One IDE which comes with Python is IDLE (https://docs.python.org/3/library/idle.html)
  - Windows: Once you've installed Python, this should be available from something like: Start → Applications → Python34 → ...
  - Mac: Check the Applications folder (or use spotlight to find it)

IDEs

Other than IDLE, there are a number of IDEs that are popular:

- PyDev: http://www.pydev.org
  - runs on top of Eclipse (http://www.eclipse.org)
- PyCharm: http://www.jetbrains.com/pycharm/
- PTVS (Windows): http://microsoft.github.io/PTVS/
- IDEs for Anaconda: https://docs.continuum.io/anaconda/ide_integration
- ...

Also consider

- ipython (jupyter): https://ipython.org
  - Graphical, interactive interface
  - Integrated with *notebook* facilities