Introduction to Programming with Python – Basics

Swapnil Sayan Saha and Noor Nakhaei
PhD Students, ECE and CS
swapnilsayan@g.ucla.edu,
noornk@ucla.edu

Note: modified from original LACC slides by Dr. Lucas Wanner, Prof. Mani Srivastava, Dr. Mark Gottscho, Dr. Bharathan Balaji, Wojciech Romaszkan, Aishwarya Sivaraman, Lev Tauz, Ankur Sarker, Sandeep Singh Sandha and Shuyang Liu

Slides taken from UCLA-LACC Intro module

Why do we choose Python?

- Relatively ***simple syntax***.
- Relatively easy to ***debug***.
- ***Availability of open-source tools*** and modules/libraries for eclectic applications with ***large development community.***
- ***Platform agnostic*** (runs anywhere).
- ***Broad application spectrum***, e.g. web applications / frameworks, standalone applications, video games, data science, artificial intelligence, machine learning, embedded systems etc.

How simple is Python?

- To print “Hello world!” on screen, C, Java and Python need to do:

- **C:**

```c
#include <stdio.h>

int main() {
    printf("Hello World!");
    return 0;
}
```

- **Java:**

```java
public class HelloWorld {

  public static void main (String[] args) {
     System.out.println("Hello, world!\n");
  }

}
```

```
print "Hello, World!"
```

Find Python around you

- Python is used everywhere!
  - **Web applications (server side):**
    - YouTube
    - Facebook
    - Dropbox
    - Pinterest
    - MoinMoin, a popular wiki engine




Find Python around you

- Python is used everywhere!

  - **Applications (client side):**
    - Original BitTorrent client, along with several derivatives
    - Ubuntu Software Center, a graphical package manager
  - Dropbox (desktop application)
  - Mercurial

Find Python around you

- Python is used everywhere!
  - **Web frameworks:**
    - Google App Engine
    - Django

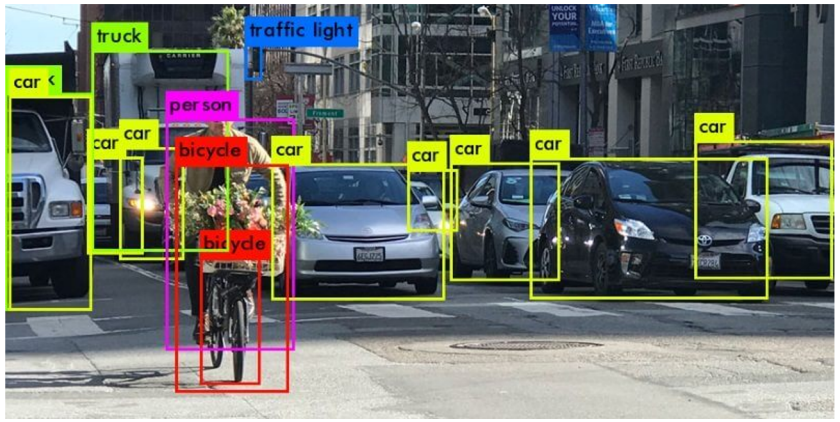
Find Python around you

- Python is used everywhere!
  - **Video games:**
    - Civilization IV
    - Battlefield 2

Find Python around you

- Python is used everywhere!

  - **Machine Learning:**
    - Pytorch
    - Tensorflow



Getting Python

- **Python Official Website:** http://www.python.org

- Lots of “packages” that add capabilities to the basic language and “tools” that make writing, running, and debugging easier

- Free and commercial “distributions” that bundle core Python with useful packages and tools and simplify installation, e.g.
  - **Anaconda:** https://store.continuum.io/cshop/anaconda/
  - Enthought Canopy: https://www.enthought.com/products/canopy/

- Browser-based Python (limited functionality)
  - http://www.tutorialspoint.com/ipython_terminal_online.php
  - http://www.tutorialspoint.com/execute_python_online.php
  - http://repl.it/languages/Python
  - http://pythonfiddle.com

Free Python Learning Material on the Web

- Official Tutorial: https://docs.python.org/2/tutorial/
- TutorialsPoint: http://www.tutorialspoint.com/python/index.htm
- Google: https://developers.google.com/edu/python/
- LearnPython: http://www.learnpython.org
- Code Academy: http://www.codecademy.com/en/tracks/python

- And many many more…

Jupyter Notebooks and Google Collab

- **Jupyter Notebook App** is an application running inside the browser. It allows you to edit and run code via a web browser.

- It is a document that contains code (e.g. python) and text elements.

- For this course, we will be using Jupyter notebooks (.ipynb files) with Google Collab.

- How to use Google collab?
  - https://www.youtube.com/watch?v=RLYoEyIHL6A
  - Importing Jupyter notebooks in Collab from Github: https://www.youtube.com/watch?v=jjdBbqV7q8s

Python Basics: Variables, Types & Operators

- What is a Variable?

  A variable is a place in memory, which has a name, and where you can store a value (a number, phrase, etc.).

```
In [1]: temperatureF = 78
```

- Assigning Values to Variables:
  - Python variables are declared automatically when you assign a value to a variable.
  - The equal sign (=) is used to assign values to variables.

- Values may be of different "types"
  - Common Data Types: int, str, float, bool, list, dictionary

- Operators perform arithmetic and logical operations over variables and values.

Python Operators

- ***What is an operator?***
  - Python operators are very similar to the mathematical operators we use everyday. For example, +, -, *…

Python Basic Operators

Python Operator Types:

- Arithmetic Operator
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operator

Python Arithmetic Operators

Assume variable **a** holds 10 and variable **b** holds 20 then:

Addition	a **+** b = 30
Subtraction	a - b = -10
Multiplication	a * b = 200
Division	b **/** a = 2
Reminder	b **%** a = 0
Exponent	a ** b = 10^{20}

```
In [3]: temperatureC = (temperatureF - 32)*0.5556
        print(temperatureC)

        25.5576
```

Python Arithmetic Assignment Operators

Assume variable **a** holds 10 and variable **b** holds 20 then:

Addition	a **+=** b	a=30
Subtraction	a -= b	a = -10
Multiplication	a *= b	a = 200
Division	b **/**= a	b = 2
Reminder	b %= a	b= 0
Exponent	a **= b	a = 10^{20}

Python Comparison Operators

Assume variable ***a*** holds 10 and variable ***b*** holds 20 then:

Equality	a == b = False
Inequality	a != b = True
Less than	a < b = True
Greater than	a > b = False
Less than or equal	a **<=** b = True
Greater than or equal	a **>=** b = False

```
In [4]: temperatureF > 80

Out[4]: False
```

Python Logical Operators

Assume variable ***a*** holds True and variable ***b*** holds False then:

Logical AND	a **and** b = False
Logical OR	a **or** b = True
Logical NOT	**not** a = False

```
In [9]: humidity = 50
        temperatureF > 80 or humidity > 30

Out[9]: True
```

Python Data Types

- In computer programming, a data type is a classification identifying one of various types of data, just like in real life we separate words “Hello!” and numbers “12345”.

- Standard Data Types in Python:
  - **Numbers**, e.g., 1000
  - **String**, e.g., “Hello world!”
  - **Boolean** e.g. True or False
  - **List**, a container e.g., [365, 12, 30] or [“hi”, “every”, “body”]
  - **Dictionary**, a key-indexed container.

Python Types: Numbers

- **Numbers**
  - Store numeric values,
  - Example:

```
In [1]: width = 20
        height = 15
        width * height

Out[1]: 300
```

  - In this example, “width = 20” defines variable width to be 20, and “height = 15” defines height to be 15. This sort of number is called **integer** or **int**.
  - After initializing width and height, we can calculate width * height as shown in the example.
  - To use real numbers (also called **float / double**),, add a dot:

```
In [10]: width = 20.5
         height = 15.25
         width * height

Out[10]: 312.625
```

Warm-up

- Start up the Jupyter notebook in Google Collab
- Do the “Warm-up” part (instructions included)

Python Types: Lists

- **Lists:**
  - Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets.
  - Example:

```
In [2]: a = ['blue', 'red', 100, 365]
        a

Out[2]: ['blue', 'red', 100, 365]
```

  - In this example, we assign four items to list a. The items are string 'blue', string 'red', number 100, and number 365.
  - We print a's items in the second command, and get its four items printed.
  - Note that items in the list have different types (string and int) – this is not common among programming languages.

Python Types: Lists

- **Lists:**
  - It is possible to change individual elements of a list.
  - Example:

```
In [7]: a = ['blue', 'red', 100, 365]
        a[2] = a[2] + 50
        a[2]

Out[7]: 150
```

  - In this example we modify the **third** element in the list by adding 50 to its value
  - We then print the third element by using *name*[*index*]

Python Types: Lists

- **Lists:**
  - Many “operations” and “methods” available for lists:

Is the value in the list?	100 in a = True
Is the value not in the list?	‘yellow’ not in a = True
Length of the list	len(a) = 4
Index of an element	a.index(‘red’) = 1
Append element	a.append(10)
Reverse the list	a.reverse()

+ sort, concatenate, insert, sum and many more at: https://docs.python.org/2/tutorial/datastructures.html#more-on-lists

Python Types: Range

- **Creating number sequences:**
  - **range(n)** for integer n > 0 returns the list [0, 1, … n-1]

```
In [26]: a = range(10)
         print(*a)

0 1 2 3 4 5 6 7 8 9
```

  - **range(m,n)** returns the list [m, m+1, … n-1]

```
In [31]: a = range(5,10)
         print(*a)

5 6 7 8 9
```

  - **range(m,n,o)** returns the list [m, m+o, … n-1]

```
In [32]: a = range(0,10,2)
         print(*a)

0 2 4 6 8
```

Exercise # 1

- Go to Exercise 1 section in your Jupyter Notebook
- Given a list L = [2, 3, 4, 5, 6]

  - Print the second item in this list.
  - Get the sum of all the values in this list.
  - Append value 7 after value 6 into the list.
  - Using range, print all the even numbers from 4 to 12.

  - Look at https://docs.python.org/2/tutorial/datastructures.html#more-on-lists for various methods available for list data type

Python Types: Strings

- **Strings:**
  - Strings in Python are identified as a contiguous set of characters in between quotation marks.
  - A String is a list of characters or **char** data types where each char specifies the symbol.
  - Example:

```
In [34]: string = "ABCDEFG."
         string
Out[34]: 'ABCDEFG.'
```

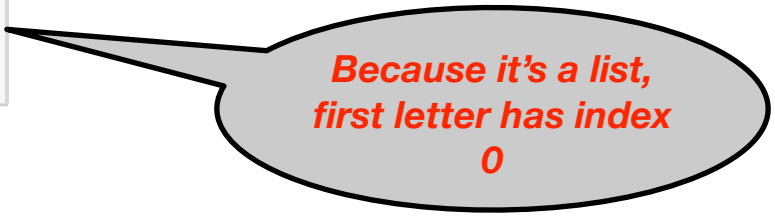
  - In this example, we assign “ABCDEFG.” to variable string. And we print string. The result we get is still “ABCDEFG.”

Python Types: Strings

- **Strings:**
  - Strings can be subscripted (indexed). This is because they are essentially lists of characters.
  - Example: In the string defined as str = “ABCDEFG.”, we query the **fifth** letter in this string.

```
In [35]: string = "ABCDEFG."
         string[4]

Out[35]: 'E'
```



Python Types: Strings

- **Strings:**
  - Many “operations” and “methods” available for strings:

Is the letter in the string?	‘A’ in string = True
Is the substring in the string?	‘CDE’ in string = True
Length of the string	len(string) = 8
Index of a letter (first)	string.index(‘A’) = 0
Append element	string + “Z” = “ABCDEFG.Z”
Make lowercase	string.lower() = “abcdefg.”

and many more at https://docs.python.org/2/library/stdtypes.html#string-methods

Exercise # 2

- Go to Exercise 2 section in your Jupyter Notebook
- Given a string S: “This is an example string”

  - Print the fifth letter in the string.
  - Capitalize all letters.
  - Replace spaces with hyphens (a bit more advance).

  - Look at https://docs.python.org/2/library/stdtypes.html#string-methods for various methods available for list data type

What will we cover on Monday?

- Review of basics if necessary
- Computer fundamentals
- Python concepts:
  - Splicing
  - Dictionaries
  - Control Flow and Decision Making
  - Loops
  - Functions
  - Python Modules and Namespaces
  - Recursion

- Assignments