120th ASEE Annual Conference & Exposition
FRANKLY, WE DO GIVE A D*MN
June 23-26, 2013

Paper ID #6601

# A CASE FOR PYTHON SCRIPTING IN UNDERGRADUATE ENGINEERING TECHNOLOGY

**Dr. jai p agrawal, Purdue University, Calumet (Tech)**
**Prof. Omer Farook, Purdue University, Calumet (Tech)**

# A CASE FOR PYTHON SCRIPTING IN UNDERGRADUATE ENGINEERING TECHNOLOGY

Abstract

This paper presents a new course in Python Programming in the undergraduate program of study in Engineering/Technology/Science. Motive behind using Python is that it is a programming language that lets interactive and quick design and effective integration with modern systems. Python usage leads to immediate gains in productivity and lower maintenance costs. Python is becoming the work-horse in all new computer science activity in the modern industry. It supports multi programming paradigms, including object-oriented and structured programming. Python elegantly combines the number crunching capability like that of MATLAB with the programming ease of C based languages with a difference of better readability and interactivity .

The Python Programming is a 400-level, 3-credit course that contains all five components:
a) the basic elements like the statements, comments, data types, data manipulation input/output and control flow, b) data structures like dictionaries, lists, tuples, and classes c) structured and object oriented programming methods,  d) interactive graphic programming and e) the html, xml and http processing.

The paper elaborates the pedagogy of classroom delivery and impact on student comprehension, conceptual understanding, learning and mastering of Python philosophy. Both methods of vertical and horizontal learning methods are used in this class. All programs that students write are added to a class repertoire which the current and future students will have access to for enhanced horizontal learning. Students are required to a design a project at the end of the class in which student teams of twos work on a project using python and share with the whole class.

The paper presents the student feedback and its analysis. The authors intend that this paper serves as a pointer to fellow academicians in bringing the technological currency in the undergraduate Engineering/Technology/Science programs.

## I. Introduction

Currently most of the Curriculum programs in Electrical, Electronic, Computer and similar tracks use one or two programming courses. Most of these programs use either Basic or C++. Of the more recent languages; Java, PHP, Python, Ruby use of Python is gaining ground among modern computer programmers. Learning Python is easier, less grammatical and uses more natural syntax. These two reasons are enough to make the case for teaching it as the first programming language.

Python is easy to learn and simple to program. Python is highly readable. It is a high-level programming language that can be used for a wide variety of programming tasks that an electrical/computer graduate will encounter in their professional carriers for at least five years.

In modern computer industry, Python is used extensively for system administration and maintenance tasks. "As of October 2012, Python ranks at position 8 in the TIOBE Programming Community Index.[1] Large organizations that make use of Python include Google, Yahoo, CERN, NASA, ILM and ITA."[2]. Google has developed a language GO for its website which is very similar to the Python itself.

Python is used in web applications, such as the Apache web server and in the web application frameworks like Django, Pylons, Pyramid, web2py, Flask and Zope. Python is standard component in several operating systems, such as several Linux distributions, OS X, Ubuntu and Fedora. The highly popular Raspberry Pi single-board computer project has adopted Python as its principal user programming language.

Python is an interpreted programming language that is automatically compiled into machine code (executable code) and executed immediately. The ensuing machine code is saved automatically, so that any more compilation is not needed so long as the source code is unchanged. It is also a dynamically typed language and highly interactive like the command-line MATLAB[3] instructions.

Most important reason for using Python, to some educators, is that it is open source. This enables easy access for students. It may "fire up" more students than otherwise. It has extensive open source library and it is continuously growing. As all educators know so well that learning programming requires more off-class engagement by students. It enables more interactive exercises than the in-class education. This is certainly helpful in learning among students at the horizontal level.

Being open source, Python programmers have access to extensive libraries like NumPy, SciPy and Matplotlib for scientific computing. PiCloud provides supercomputing processing capacity on the cloud, using a Python interface.

## II. Features of Python

### 2.1. Indenting

The most unusual aspect of Python is that whitespace is significant; instead of block delimiters (braces → "{}" in the C family of languages), indentation is used to indicate where blocks begin and end. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. For example, the following Python code can be interactively typed at an interpreter prompt. "Hello World!" appears on the user screen immediately by pressing the *Enter* key after the statement:

```
>>> print "Hello World!"
Hello World!
```

### 2.2. Multi-Platform

Python can run on Microsoft Windows, Macintosh and all Linux distributions with ease. This renders the program coded in Python very portable, as any program written for one Platform can easily be used at another.

### 2.3. Declaration-free Data

Unlike C/C++ Python does not require declaration of data types. The interpreter automatically recognizes the type of data when entered. Example:

```
>>> x=3.4   #a float
>>> y=5     # an integer
>>> x*y
```

```
17.0
```

## 2.4. Rich Element Types

Python provides a powerful assortment of built-in types of data (e.g., lists, dictionaries and strings), a number of built-in functions, and constructs. For example, loop constructs that can iterate over items in a collection instead of being limited to a simple range of integer values.

## 2.5. Programming/Modes: Interactive /Scripting

Interactive mode is a command line shell which gives immediate feedback after entering a statement. The Python continues to run previously fed statements in active memory for use in subsequent statements.

The prompt >>> on the screen shows that it is in the interactive mode. In interactive mode what you type is immediately run. A sample interactive session:

```
>>> 5*4
20
>>> "hello"*4
'hellohellohellohello'
```

If one ever feels the need to play with new Python statements, just go into interactive mode and try them out.

The script mode is the mode where the python statements are written in a file, saved as *script_mode.py*,

```
#script_mode.py      script mode programming
print 5*4
print "hello"*4
```

The script is run from the script window itself (one of many ways it can be executed). The result is shown in the interpreter window:

```
>>> ============================ RESTART
==============================
>>>
20
hellohellohellohello
```

Alternatively import the script as a module in the interpreter and run by pressing *Enter*.

```
>>> import script_mode
20
hellohellohellohello
```

To make the experience interactive, we write the following script in *area_of_circle.py* file:

```
#area_of_circle.py

radius=input("Specify the radius in meters and then enter: ")

area = 3.1415*radius**2

print "the area of circle of radius = ", radius, " meters is = ",  area, "square meters."
```

**Input** is a built in function to get the input from the keyboard. The input from the keyboard is evaluated and then assigned to the variable radius. The **input** is basically a delayed input.

Save and Run either in the script window or import in the interpreter window as a module. The result in the interpreter window is

```
Specify the radius in meters and then enter: 5
the area of circle of radius =  5  meters is =  78.5375 square meters.
```

## 2.6. Multiple Programming Paradigms

Python supports multiple programming paradigms: a) sequential programming b) structured programming and c) object-oriented programming. The later being the preferred paradigm, is also recommended by the in the first class in programming.

### 2.6.1. Example of a structured programming (function based) is shown below. Save, and Run and then invoke from the interpreter window as shown:

```
#Greetings.py    module name
def greetings(person):             #the function header
    print "Hello", person
    print "How are you today!!"
```

Person is passed as the argument within the ( ) in the function header.

```
>>> greetings ("John")
```

```
Hello John
How are you today!!
```

### 2.6.2. Object-Oriented Programming in Python

Object-oriented paradigm is to view a complex system as the interaction of simpler *objects*. Objects  (entities, items or things) have **methods** (functions, actions or events) and **properties** (values, attributes or characteristics). Objects interact with other objects. In the Object-Oriented_Programming approach, the **objects** are defined with **methods** and **properties**, resulting in more readable, more reusable code.

Example of a projectile object:

The projectile is specified by angle of launch, initial height and the initial velocity of launch. The program calculates the horizontal position, vertical position and the corresponding velocities at every second.

We will define a class **Projectile** in a module **projectile.py**.

```
# projectile.py
from math import pi, sin, cos

class Projectile:
    def __init__(self, angle, velocity, height):
        self.xpos = 0.0
        self.ypos = height
        theta = pi * angle / 180.0
        self.xvel = velocity * cos(theta)
        self.yvel = velocity * sin(theta)
    def update(self, time):
        self.xpos = self.xpos + time * self.xvel
        yvel1 = self.yvel - 9.8 * time
        self.ypos = self.ypos + time * (self.yvel + yvel1) / 2.0
        self.yvel = yvel1
    def getY(self):
        return self.ypos
    def getX(self):
        return self.xpos
```

Now we write a Python script **cball2.py** where we use the just created class,

```
#cball2.py
from  projectile  import  Projectile
# note the first projectile is the module and second Projectile is the class there in

def getInputs():
    a = input("Enter the launch angle (in degrees): ")
    v = input("Enter the initial velocity (in meters/sec): ")
    h = input("Enter the initial height (in meters): ")
    t = input("Enter the time interval between position calculations: ")
    return a,v,h,t

def main():
    angle, vel, h0, time = getInputs()
    cball = Projectile(angle, vel, h0)    #instantiation of the object cball in the class Projectile
    while cball.getY() >= 0:
        cball.update(time)

    print "\nDistance traveled: %0.1f meters." % (cball.getX())
```

Now run the Python script in the Interpreter.

```
>>> import cball2
>>> main()
Enter the launch angle (in degrees): 30
Enter the initial velocity (in meters/sec): 50
Enter the initial height (in meters): 2
Enter the time interval between position calculations: .1
```

The python returns the result in the interpreter window.

```
Distance traveled: 225.2 meters.
```

## III. Course Syllabus and Description

ECET 49900 - Dynamic Programming with Python
class 3, lab 0, credit 3

Python is a programming language that lets interactive and quick design and effective integration with modern systems. Use of Python leads to immediate gains in productivity and lower maintenance costs. Python is becoming the work-horse in all new computer science activity in the modern industry. It supports multi programming paradigms, including object-oriented and structured programming. Python elegantly combines the number crunching capability like that of MATLAB with the programming ease of C based languages with a difference of better readability and interactivity. Topics cover a) the basic elements like the statements, comments, data types, data manipulation input/output and control flow, b) data structures like dictionaries, lists, tuples, and classes c) structured and object oriented programming methods, d) interactive graphic programming and e) the html, xml and http processing. All programs that student will be collected in the form of a class repertoire which the future students will have access to for enhanced horizontal learning.

## IV. Course Pedagogy

The pedagogy of the course is based on Outcome Base d Education[5] , and utilizes the interactive model of learning[6]. All the students maintain an online portfolio of their work. The system designed in the laboratory to perform a specific task is the core measurement as the learning outcome of the course. The laboratory performance of the course is performed in teams of two/three students. This mode provides a platform for horizontal learning through active and engaged discourse and discussion. Students are empowered to charter their learning and feed their curiosity. These classroom practices and laboratory environment provides a challenging and invigorating environment that prepares them for a lifelong learning process and career path.

Part 1 – Basic

| | |
|---|---|
| 1. Intro to Python | 2 hours |
| 2. Basics | 2 hours |
| 3. Sequences (Strings, Lists, Tuples, Dictionaries, Sets) | 2 hours |
| 4. Math (use NumPy and Matplotlib modules) | 2 hours |
| 5. Functions | 2 hours |
| 6. String manipulations | 2 hours |
| 7. Control structures | 2 hours |
| 8. Python Project I | 1 hour |
| 9. Test 1 | 1 hour |

Part 2 – Intermediate

| | |
|---|---|
| 1. Classes | 4 hours |
| 2. Gui – an introduction | 2 hours |
| 3. Object Oriented Design | 6 hours |
| 4. File operations | 2 hours |
| 5. Python Project II | 1 hour |
| 6. Test 2 | 1 hour |

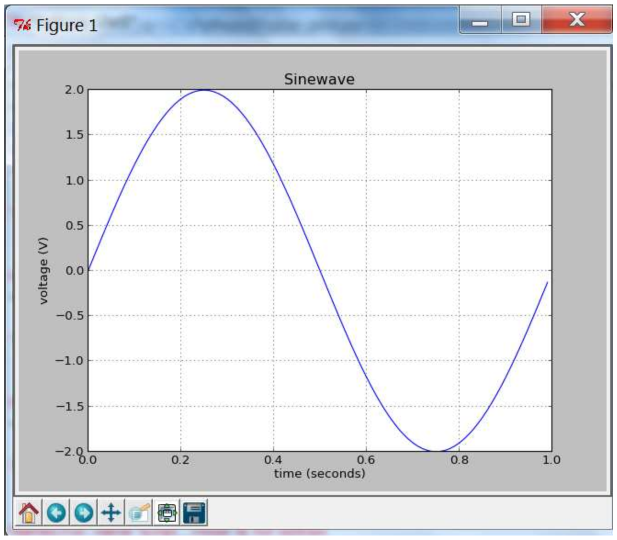
Part 3 – Advanced

| | | |
|---|---|---|
| 1. | Database | 6 hours |
| 2. | Web programming | 4 hours |
| 3. | Interfacing with C and MATLAB | 4 hours |
| 4. | Final Test | 2 hours |

An example of using open-source plotting example is shown below:

```
#sine_plot.py
from pylab import *

f=1.0          #frequency 1 Hz
T=1/f           #period
A= 2.0          #Peak amplitude
#time array from 0 to T in 100 steps
t = arange(0.0, T, T/100)
sine = A*sin(2*pi*f*t)
plot(t, sine, linewidth=1.0)
xlabel('time (seconds)')
ylabel('voltage (V)')
title('Sinewave')
grid(True)
show()
```



Course Assignments
The course requires a weekly position paper that ex
pounds the conceptual understanding of the
subject matter content and inferences drawn from th
e laboratory performance. The course
assignments are submitted on line. Each student mai
ntains an online portfolio of the work.

V. Expected Outcomes: Tangible and Intangible

Learning objects in Spring 2013: Students should be able to program exercises in

1. basic elements of programming
2. data structures
3. structured and OOP methods
4. Web programming
5. interfacing with C and MATLAB

## VI. Student and Faculty Feedback Regarding Developed Course.

This class is offered as an independent study course in the Spring 2013 semester because of cost cutting measures there is not much room for experimental courses being offered.. The authors will report the experiences in the final presentation at the ASEE conference.

## VII. Summary

This paper presents the motivation for a new course in Python Programming in the undergraduate program of study in Engineering/Technology/Science. Python is easy to learn, simple and very readable. Python interpreter and associated software is open source, supported by a wide community of users and libraries, therefore, it is easily accessible for off-class learning. Python empowers the learners to explore, interact dynamically. Python can be easily embedded in system design leads to immediate gains in productivity and lower maintenance costs. The paper presents the details of a 3-credit semester level course for beginners in the undergraduate programs in Engineering/Technology/Science.

## Bibliography

[1] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
[2] http://en.wikipedia.org/wiki/TIOBE_Programming_Community_Index
[3] http://www.mathworks.com/
[4] http://en.wikibooks.org/wiki/Python_Programming
[5] Omer Farook, Jai P. Agrawal, Chandra R. Sekhar, Essaid Bouktache, Ashfaq Ahmed "Outcome Based Education and Assessment" , Proceedings of the 2006 American Society for Engineering Education Annual Conference & Exposition June 20 -23, 2006. Chicago, IL
[6] http://www.learning-theories.com/