RECIPE FOR ILLUSTRATIVE WEBSITE DESIGN IN SCIENCE AND ENGINEERING†

Godfrey E. Akpojotor

Theoretical and Computational Condensed Matter Physics, Physics Department, Delta State University, Abraka, Nigeria

Abstract

The mission of the Python African Computational Science and Engineering Tour (PACSET) Project is to take Python to the nook and cranny of Africa to kick start a generation of Python programmers in science, technology, engineering, mathematics and innovation (STEMI) in the continent. One of the projects to achieve this is to use illustrative websites. In this current work, the recipe for designing illustrative websites in science and engineering (S &E) using Python codes and Django is presented.

Keywords: Python, Django, Dreamweaver, Illustrative Website

E-mail for corresponding author: akpogea@delsu.edu.ng: info@pacset.net

1.0 INTRODUCTION

As the relationship between teaching/learning/research and computation continually evolves, new tools are required not only to treat numerical problems, but also to solve various diverse problems from algorithm implementations, computer modeling, simulation and visualization to website design and presentation. The implication is that computational approaches have become an integral part of modern science, technology, engineering, mathematics and innovation (STEMI). In deciding the computational approaches for STEMI, two important choices have to be made and they are the choice of the generalized numerical methods and the choice of the programming language. Numerical methods which is the study of algorithms for the problems of continuous mathematics finds applications in all fields of STEMI [1,2]. It is therefore broad and diverse hence it is taught as full courses in Mathematics Departments [3].

†***African Journal of Physics Vol.6, pp. 131-145, (2013)***

ISSN: PRINT: 1948-0229 CD ROM:1948-0245 ONLINE: 1948-0237

The means of conveying the task of algorithm to a computer is by a computer programme which is the instruction given to the computer on how to implement the specific algorithm. Such instructions which are usually collections of statements are designed (or written in codes) to follow a particular structure and syntax commonly known as the programming language. Therefore the choice of the programming language is an important factor to consider in computational physics. An important consideration in making the choice of programming environment to teach new beginners, is the desire to ease the learners into programming and to give them the opportunity to develop a conceptual model of what a programme is and what it does [4]. Python which has a clean design and object-oriented nature as well as a comprehensive library of modules not only for scientific computation but also for tasks as diverse as manipulating images and running web server, is gaining the reputation as the popular choice of a programming environment for a first exposure to computation in STEMI [5-8]. Even more compelling is that Python is an open source, multi-platform and free software so that any user connected to the internet can download the entire package into any platform, install it and immediately begin to use it [6].

It is this observation that actuated the formation of the Python African Tour (PAT) (See www.pythonafricantour.com) Project led by a group of developers, advocates and activists, to help programming advance in Africa through the use of agile technologies such as the Python language [8]. Starting with Morocco in 2008, PAT has been visiting several countries in Africa with Python (General introduction, Django, SciPy) trainings being organized in each country. PAT had its only stop in Nigeria till date in June 2010 and was hosted by the African University of Science and Technology, Abuja. This event was organized with the help of Datasphir and sponsored by the National Information Technology Development Agency (NITDA), Abuja while Google sent some of its resource persons. The participants were more of web designers. Some of us had a meeting during this event where it was suggested that I should be the coordinator of scientific computation of the PAT. This led to my initiation of the Python African Computational Science and Engineering Tour (PACSET) Project and its mission is to ease the learners in Africa into programming in S & E with Python and to use it for modeling, simulation and visualization to aid the teaching/learning process and research in Africa (See www.pacset.net). Currently, the three major goals of PACSET are: (1) teaching Python to students, new beginners as well as expert programmers in S & E (2) using Python to model, simulate and visualize concepts, laws and phenomena in S & E to compliment the teaching of theory and experiment and (3) using Python to model, simulate and visualize laboratory experiments to aid the teaching and learning of experimental S & E.

Currently we have a growing compendium of Python programmes on modeling, simulation and visualization of concepts, laws and phenomena as well as laboratory experiments in S & E using Python to meet the three aforementioned goals [6]. Further, we have organized several local and international workshops to teach both beginners and even expert programmers the Python programming language. It was observed in some of these workshops that some of the participants would also want to learn how to design websites using Python. This led to including how to design illustrative websites for S & E using Python in the PACSET project. The remaining part of this paper will be on the recipe to design this illustrative website.

2.0 THE DESIGN TOOLS AND DEVELOPMENT OF AN ILLUSTRATIVE WEBSITE

Web applications are programs that run within a web browser which is a software application for presenting, retrieving and traversing information resources on the World Wide Web [9,10]. It does this with the aid of a Uniform Resource Locator (URL), that identifies an information resource which can either be a web page, an image, an audio or video file or other piece of contents. The presence of Hyperlinks in resources enable users to easily navigate their browser. The major web browsers include Torch, Google chrome, Mozilla Firefox, Internet Explorer, Opera and Safari.

An illutratative website here is a web application in form of a set of related webpages served from a single web domain to demonstrate the desired concept, law or phenomenon as well as experiment. In general, a webpage is a document, typically written in plain text interspersed with formatting instructions of Hypertext Markup Language (HTML) [9]. Webpages are accessed and transported with the hypertext transfer protocol (HTTP). This is possible because the user's application, often a web browser, render the page content according to its HTML markup instruction onto a display terminal. The URL of the page organizes them into a hierarchy. Web pages can be viewed or otherwise accessed from a range of computer based and internet enabled devices of various size including desktop computers and laptops. A website is hosted on a computer system known as a web server, also called HTTP server. The ability to maintain and update web applications without disturbing and re-installing software on potentially thousands of client computers, is a key reason for their popularity which is the intrinsic support also for their cross platform compatibility [10]. This web application software update may occur each time the webpage is visited.

The web application software can be created using the following major steps:

i. Installation of the needed softwares to build it
ii. Setting up Django
iii. Creating dynamic web content and
iv. Mapping the URL to the view

2.1 Installation of the needed softwares to build it

Basically, to develop a Python website, we use Python codes, Django and Dreamweaver. Therefore the first step for a new beginner is to install these softwares. We start with the installation of the appropriate version of the Python and then the desired modules depending on the intended usage. Here we plan to demonstrate with two illustrative websites: (i) illustrative websites for simple differentiation and (ii) illustrative websites for plotting simple 2D graphs. Therefore the scientific modules to be installed are NumPy, SciPy, SymPy and MatPlotlib.

The Django is next installed. The Django is a free and open source web application framework, written in python which follow the model-view-controller architectural pattern [11]. The Django primary goal is to ease the creation of complex, database-driven websites. Django emphasizes rapid development and the principle of don’t repeat yourself [12]. Python is used throughout, even for setting files data model. One of Django’s most captivating features is its ability to make a database backed web site. This implies that there is need to have a database server installed. It thus enables the system to run as a mini – server, because all web pages are server powered; either remotely (through the internet) or locally (with the aid of a mini – server).

Finally, the Dreamweaver is installed. The Dreamweaver is a web design and development application that provides a visual editor and a code editor with standard features such as syntax highlighting, code completion and code collapsing as well as more sophisticated features such as real-time syntax checking and code introspection for generating code hints to assist the user in writtin code [13]. The design view facilitates rapid layout design and code generation as it allows users to quickly create and manipulate the layout of HTML elements.

2.2 Setting up Django

Starting a project implies the initialization of a collection of setting for a specific Django project, including the database configuration [11]. This is done by starting the Windows Command Prompt and typing the following; '*django-admin.py startproject filename*', in the project folder that must have been created: here the filename to be used is '*physicweb_Demo*'. This is done to create a directory called 'physicweb_Demo' in the current directory.

Next, the 'physicweb_Demo' directory is then located on the drive and opened to confirm what the 'startproject' command has created in it. This is the built – in lightweight mini web server (also called a development server) in Djangothat one can rapidly use while developing one's site, without having to deal with configuring the production web server, until one is ready for production. The following files will be ready for the production:

init.py: A file required for Python to treat the directory as a package (that is, a group of modules)
manage.py: A command line utility that lets you interact with Django project in various ways.
setting.py: A file containing configurations for this Django project.
*urls.py:*A file containing the table of content for your Django project.

After the setting up the Django, we return to the 'physicweb_Demo' directory and run the following on the command prompt; 'manage.py runserver' and then allow the server to validate model. By default, the 'runserver' command starts the development server on port 8000, only for local connections. Now that the server is running, we may then visit the following address;

‘http://localhost:8000/physicweb_Demo’, with the system’s web browser and we’ll see a ‘Welcome to Django’ page shaded with pastel blue, showing; ‘It Worked!’

2.3 Creating dynamic web content

Having set up the Python module for web design, the Python codes of the law, concept, phenomenon or experiment to be illustrated is then developed. Thereafter we have to link up the code to the website. This is achieved by writing a ‘view function’ which is simply a python function that takes a web request and returns a web response. The response can be the HTML content of a web page or a 404 error, or an Extensible Markup Language (XML) document, or an image, or anything really prepared with a Dreamweaver. The view itself contains whatever arbitrary logic that is necessary and required to return that response. Put simply, it contains all the Python codes required to display a result and it must be on the Python path in the ‘physicweb_Demo’ directory, created earlier.

The except of the view function for the illustrative website for differentailtion is the following:

```
From Sympy import*
def differentiate it(request):
   c1= request. GET ['a']
   c2= request. GET ['b']
   c3= request. GET ['c']
   c4= request. GET ['n']
   c5= request. GET ['n2']
   c6= request. GET ['n3']
   c7= request. GET ['d']
   a1= int (c1)
   b1= int (c2)
   c1= int (c3)
   n1= int (c4)
   n2= int (c5)
   n3= int (c6)
   d1= int (c7)
   x=Symbol ('x')
```

```
g=a1*x**n1+b1*x**n2+c1*x**n3+d1
d=diff (g, x,1)
Return render_to_response ('result.html', {'d': d})
```

The except of the Python codes for the illustrative website for plotting graphs is the following:

```
frompylab import*
ax = axes([0.073,0.073,0.86,0.86])
ax.set_xlim(-4,4)
ax.set_ylim(-3,3)
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))
ax.xaxis.set_major_locator(MultipleLocator(1.0)) # On the x-axis, 1cm
represents 1 unit
ax.xaxis.set_minor_locator(MultipleLocator(0.2))
ax.yaxis.set_major_locator(MultipleLocator(1.0)) # On the y-axis, 1cm
represents 1 unit
ax.yaxis.set_minor_locator(MultipleLocator(0.2))
grid(which='major', axis='x', linewidth=1.00, linestyle='-',
color='0.55')
grid(which='major', axis='y', linewidth=1.00, linestyle='-',
color='0.55')
grid(which='minor', axis='x', linewidth=0.45, linestyle='-',
color='0.55')
grid(which='minor', axis='y', linewidth=0.45, linestyle='-',
color='0.55')
x = linspace(-3,3)
y1 = x + 2
y2 = 3 - 4*x
plot(x,y1,'-c')
plot(x,y2,'-r')
text(0.4,1.4,r'$y = 3 - 4x$',fontsize=15)
text(-1.5,0.4,r'$y=x+2$',fontsize=15)
title('Linear Graph I')
show()
```

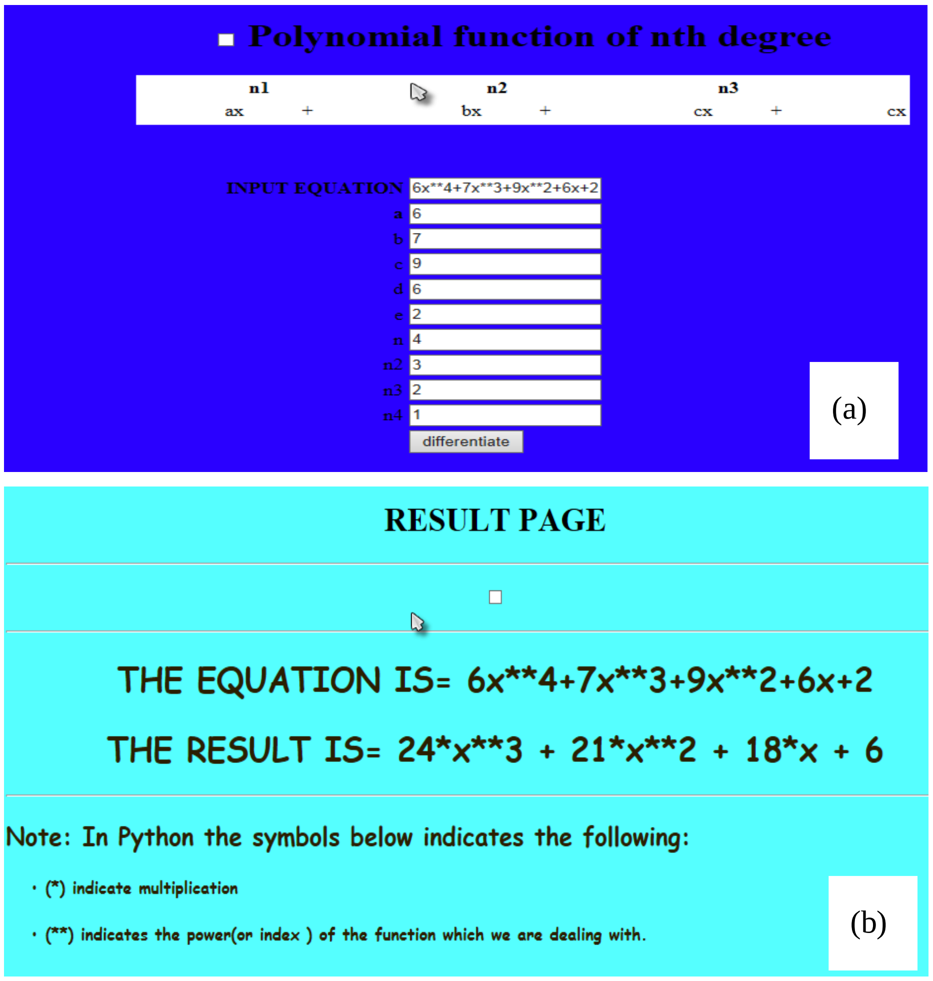
2.4 Mapping the URL to the view

The view function actually returns a response page that includes the result of the request. The view function is linked to the Django using the URL configuration (URLconf).The URLconf is like a table of content for the Django powered web site. It is a mapping or function between URL patterns and the view functions that should be called for those URL patterns. The URLconf is actually how one tells Django to call a specific code for a particular.

3.0 WEB APPLICATION OUTPUT

The output which is the requested web page for the illustration is interactive as one is able to communicate with the server database via the web browser by inputing variables. Fig. 1a shows the snapshot of the illustrative input webpage for the differentiation of the polynomial function of n^{th} degree, $16x^4 + 7x^3 + 9x^2 + 6x + 2$ say and Fig. 1b shows the result webpage. The website designer decides how to display the result; here the output webpage shows both the input function and the result.

Observe that for simplicity, different input webpages are created in this current study for the differentiation of the various functions. It is also possible to create a single input webpage applicable to all types of functions. This may require a little more complex programming than the one adopted for the demonstration here. So as depicted in Fig. 2a, a different input webpage is created for the differentiation of the product of functions, $4x^7 \sin x$ say.



Polynomial function of nth degree

n1		n2		n3		
ax	+	bx	+	cx	+	cx

INPUT EQUATION	6x**4+7x**3+9x**2+6x+2
a	6
b	7
c	9
d	6
e	2
n	4
n2	3
n3	2
n4	1

RESULT PAGE

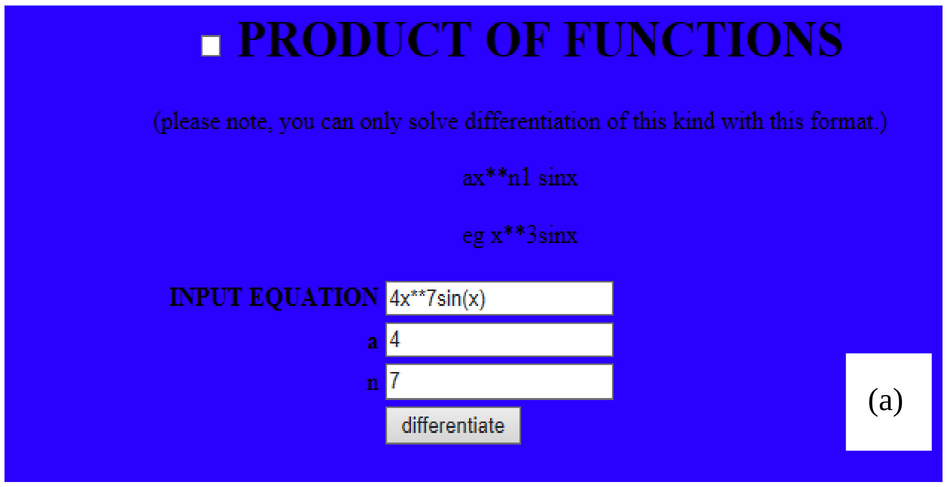
THE EQUATION IS= 6x**4+7x**3+9x**2+6x+2

THE RESULT IS= 24*x**3 + 21*x**2 + 18*x + 6

Note: In Python the symbols below indicates the following:

- (*) indicate multiplication
- (**) indicates the power(or index) of the function which we are dealing with.

Fig. 1.(Colour online) Snapshots of the illustrative webpages for the differentiation of the polynomial function of n^{th} degree, $16x^4 7x^3 + 9x^2 + 6x + 2$ say: (a) the input webpage and (b) the output webpage.



PRODUCT OF FUNCTIONS

(please note, you can only solve differentiation of this kind with this format.)

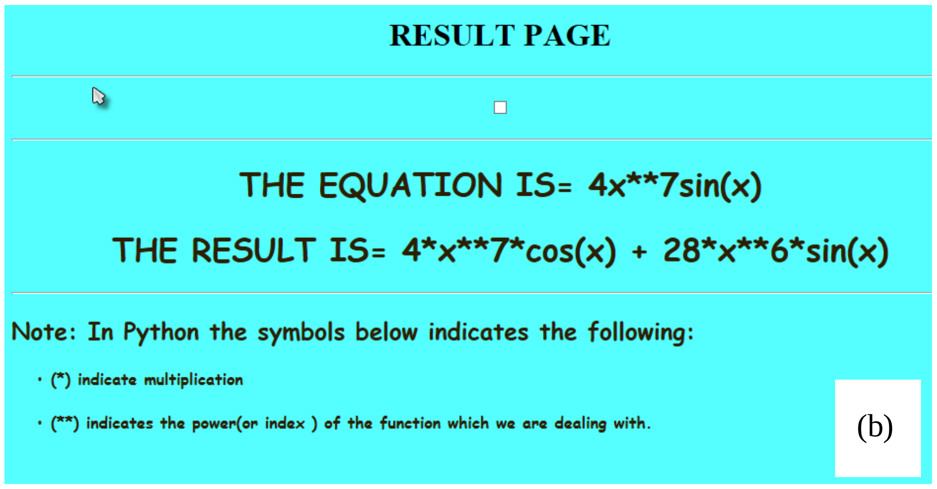
ax**n1 sinx

eg x**3sinx

INPUT EQUATION 4x**7sin(x)

a 4

n 7



RESULT PAGE

THE EQUATION IS= 4x**7sin(x)

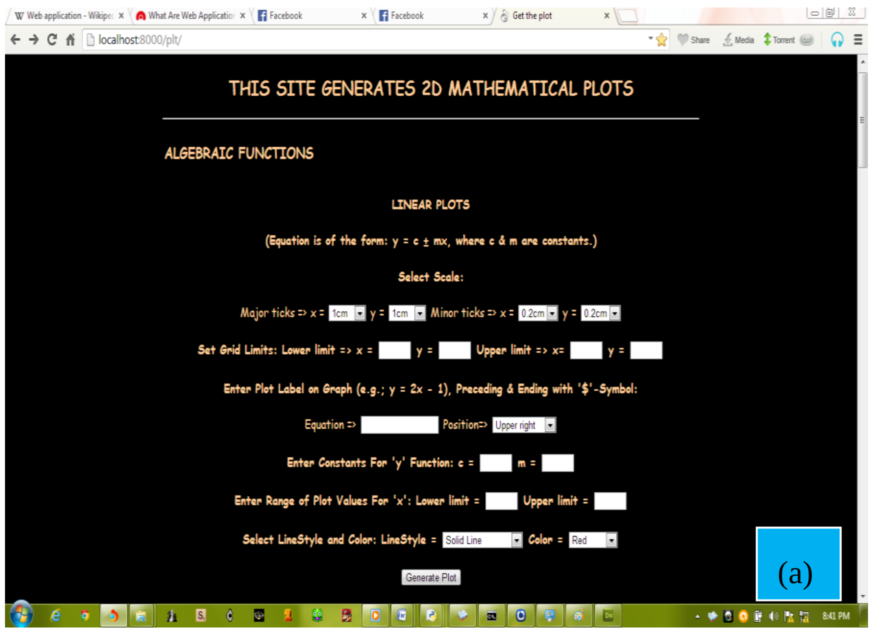
THE RESULT IS= 4*x**7*cos(x) + 28*x**6*sin(x)

Note: In Python the symbols below indicates the following:

- (*) indicate multiplication
- (**) indicates the power(or index) of the function which we are dealing with.

Fig. 2.(Colour online) Snapshots of theillustrative webpages for the differentiation of the product of functions, $4x^7 \sin x$ say: (a) the input webpage and (b) the output webpage.

Similarly, for simplicity, different input webpages are created in this current study for the different type of graphs. So the webpage for functions is different from the one in which the input date are from experiments. So as depicted in Fig. 3a, a simplified input webpage is developed to plot the simultaneous equation of linear and quadratic equations, $y = -2x + 11$ and $2x^2 + 2x + 5$ say respectively and the result shown in Fig.3b.



THIS SITE GENERATES 2D MATHEMATICAL PLOTS

ALGEBRAIC FUNCTIONS

LINEAR PLOTS

(Equation is of the form: y = c ± mx, where c & m are constants.)

Select Scale:

Major ticks => x = 1cm y = 1cm Minor ticks => x = 0.2cm y = 0.2cm

Set Grid Limits: Lower limit => x = y = Upper limit => x= y =

Enter Plot Label on Graph (e.g.; y = 2x - 1), Preceding & Ending with '$'-Symbol:

Equation => Position=> Upper right

Enter Constants For 'y' Function: c = m =

Enter Range of Plot Values For 'x': Lower limit = Upper limit =

Select LineStyle and Color: LineStyle = Solid Line Color = Red

Generate Plot

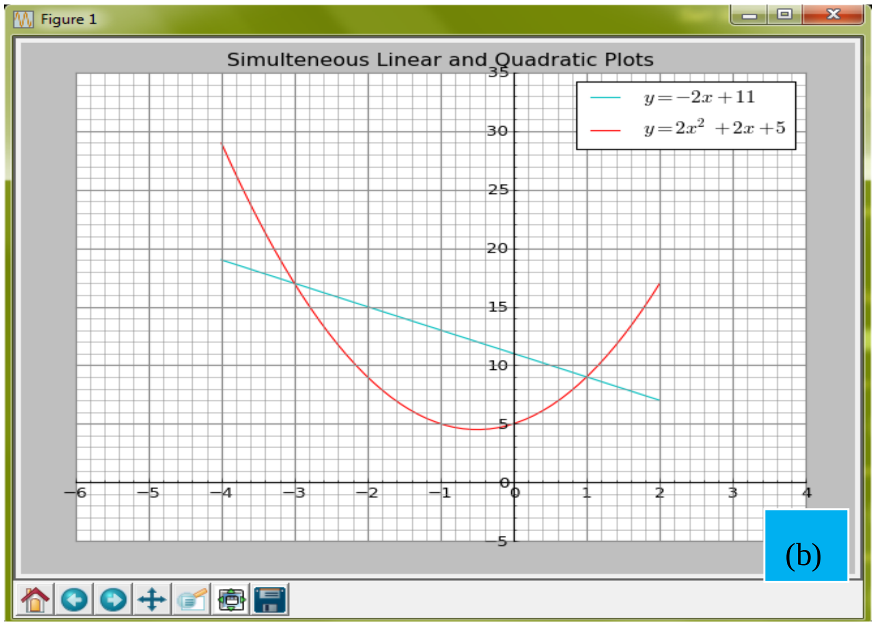


Fig. 3.(Colour online) Snapshots of theillustrative webpage for the plotting of the simultaneous equation of linear and quadratic equations, $y=-2x+11$ and $2x^2+2x+5$ say respectively: (a) the input webpage and (b) the output webpage.

4.0 CONCLUSION

The mission of the Python African Computational Science and Engineering Tour Project is to compliment the teaching/learning and research in science, technology, engineering, mathematics and innovation in Africa using Python programming language which is a free and open source software with an increasing global community. One of the unique features of Python is that it is multi-purpose. Therefore we have included illustrative website designs in our PACSET project. The simple recipe for illustrative website designs discussed in this paper can be extended in principle, to other more complex website designs.

Acknowledgement

I appreciate the very useful discussion with Pius Ufuoma Ifodo, Louis OchukoEnaemo and Davidson Unusotame. I also appreciate the Director of the National Mathematical Centre (NMC) Abuja for sponsoring the inclusion of the Python foundation postgraduate course at the International Seminar on Theoretical Physics and National Development (ISOTPAND) series at the Centre.

REFERENCES

[1] Hetland M. L. (2010) Python Algorithms: Mastering Basic Algorithm in the Python Language, Apress, New York. Read it on Safari Books online (http://bit.ly/qZ5HjY)

[2] Pang T. (2006), An introduction to Computational Physics. Cambridge University Press, Cambridge. See also Kiusalaas J. (2005) Numerical methods in engineering with python. Cambridge University Press, New York

[3] Quarteroni A., Sacco R. and Saleri F. (2006) Numerical Mathematics Texts in Applied Mathematics Vol 37, Springer-Verlag, New York.

[4] Chabby R. and Sherwood B. (2008). Computational Physics in the introductory calculus based course, Am. J. Phys. 76, 307 – 313.

[5] Borcherds P.H. (2007) Python: a language for computational physics, Proceedings of the Conference on Computational Physics 2006, Comput. Phys. Com. **177**, 199-201.

[6] Akpojotor G., Ehwerhemuepha L., Echenim M. and Akpojotor F. (2010) Modeling and visualization of some physics phenomena with python simulations, African Journal of Physics 3, 94-118.

[7] Donaldson T. (2008) Python: Visual QucikStart Guide (2nd Ed), Peachpit Press, Berkeley. Read it on Safari Books online (http://bit.ly/onljj4)

[8] Miller B. and Ranum D. (2008) Python programming in Context, Jones and Bartlett Learning. Read it on Safari Books online (http://bit.ly/oVsAot) See also Knowlton J. (2008) Python: Create-Modify-Reuse, Wiley Publishing, Inc., Indianapolis, Indiana. Read it on Safari Books online (http://bit.ly/pmqKW1)

[9] Schafer S. M. (2005) Web Standards Programmer's Reference: HTML, CSS, JavaScript, Perl, Python and PHP, Wiley Publishing, Inc., Indianapolis, Indiana. Read it on Safari Books online (http://bit.ly/qRz3ks)

[10] Sikos L. (2011) Web Standards: Mastering HTML5, CSS3, and XML, Apress, New York.

[11] Forcier J., Bissex P. and Chun W. (2008) Python Web Development with Django, Addison_Wesley Professional. Read it on Safari Books online (http://bit.ly/qLY8U3)

[12] Greenfeld D. and Roy A. (2014) Two Scoops of Django: Best Practices For Django 1.6 (2nd Ed), Two Scoops Press.

[13] McFarlandD. S. (2008) Dreamweaver CS4 - The Missing Manual, O'Reilly Media, Inc.

Appendix Step by Step development

1. Create the python project folder in the system's drive (call it any name)
2. Start the command prompt
3. Change the directory to the Python project folder by typing the following; 'cd\name_of_python_project_folder' then press the 'enter key'.
4. Create a *new*Django project by typing the following; 'django-admin.py startprojectnew_name_of_project', then press the 'enter key'.
5. Subsequently, type the following; 'cd new_name_of _project', then press the 'enter key'.
6. Again, type; 'manage.py runserver' and press the 'enter key'.
   Note that the number of spaces between each type should be observable (not more than one spacing). Note also that the Steps 5 and 6 above are majorly to *initialize or set the mini – server running* and if does not occur, one cannot preview *the database web page* with the web browser.

7. Use a standard web development tool such as Dreamweaver to design a HTML file for the web interface.
8. Create another folder in the Python project folder, to hold the HTML files and name it ‘*template*’.
9. Next, locate the Python file called ‘setting’ in the new python project folder and set the path to the template directory, in order for Django to know exactly where to find the page. This is done by;

- Locating the template file in the new Python project folder and copying the address.
- Next, right click the ‘setting file’ and edit it with the IDLE, then locate the line ‘*TEMPLATE_DIRS*’ and paste the template address after the last statement of the block of statements.
- Next, change all the back slashes to forward slashes and then put quotes at the beginning and the end of that string.

10. Create a view.py file and save this file on the same path as that containing the setting.py file. The ‘view’ file is only a Python module which contains the functions and instructions necessary to display and process a web application or page. Normally, it starts with the import statements like;

    ‘*fromdjango.http import HttpResponse*’

    This simply means importing the class *HttpResponse,* which lives in the django.http module. It is important, because it is used later on in the coding process. Next, we must define a function called the view function like this;

    defquicktest (request):

    for instance and then proceed with the rest of the codes (Adrian, et al.) Note that the name of the view function does not matter, that is, it does not have to be named in a certain way for Django to recognize it.
11. Create the HTML result page, using the web development tool and save it with ‘result.html’ in the template folder. The content of the page looks like this;

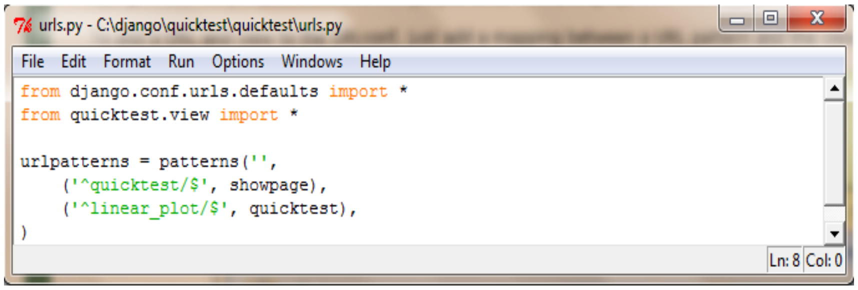
    {{name_of_new_project}}

    That is, the name of the new project created earlier.
12. Configure the URL by using aURLconf. This is done in order to tell Django explicitly, that a particular URL is being activated. The mapping between the URL pattern and the view function, is shown in the Fig. 4

13. Launch the web application using any browser of choice by entering the following address in the address bar;

'localhost:8000/name_of new_project/'

Note that the 'name_of new_project' depends on the actual name used during the design and development, else it will returns an *error response*, indicating that the page was not found


urls.py - C:\django\quicktest\quicktest\urls.py
File Edit Format Run Options Windows Help
Ln: 8 Col: 0

```
from django.conf.urls.defaults import *
from quicktest.view import *

urlpatterns = patterns('',
    ('^quicktest/$', showpage),
    ('^linear_plot/$', quicktest),
)
```

Fig. 4: (Colour online) Snapshot of the mapping between URL pattern and view function