PYTHON and TEACHING MATHS in SECONDARY SCHOOLS

Dr Stephen Lynch NTF FIMA SFHEA

Why Python?

Python for A-Level Maths, Undergraduate Maths and Employability

IMA Workshop: Python for A-Level Maths and Beyond

Register here:

Hands-On Workshop: Friday September 22nd or Saturday September 23rd 2023

In [ ]:

```
# BODMAS
4 - 7 * (8 - 5 + 20) + (3 - 5) / 2
```

In [ ]:

```
# Exponents
2**8
```

In [ ]:

```
# The number of permutations of a pack of playing cards!
from math import *
factorial(52)
```

In [ ]:

```
# Ceiling function: the lowest integer greater than or equal to x, where
ceil(2.5)
```

In [ ]:

```
# Floor function: the highest integer less than or equal to x, where x is
floor(2.5)
```

In [ ]:

```
# Truncate towards zero on the real line.
trunc(2.456)
```

In [ ]:

```
# Round to n decimal places.
round(pi , 10)
```

In [ ]:

```
# Plot a graph and save the figure with a given resolution.
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-2 , 4 , 0.01)
y = (x - 1)**2 + 2
plt.title("$y=(x - 1)^2+2$")
plt.plot(x , y , color = "blue")
plt.xlabel("x")
plt.ylabel("y")
plt.ylim(0 , 10)
# Figure will be saved in the Files folder to the left of this window.
plt.savefig("parabola.png" , dpi = 400)
plt.show()
```

2. Very Simple Programs

In [ ]:

```
# Define a function. Think of adding a button on your calculator.
def sqr(x):
  return x * x

sqr(-30)
```

In [ ]:

```
# Using a while loop to sum natural numbers.
def sum_N(n):
  sum , i = 0 , 1
  while i <= n:
    sum += i       # sum = sum + i.
    i += 1         # i = i + 1.
  print("The sum is" ,  sum)

sum_N(100)
```

In [ ]:

```
# Using if, elif, else to test integers.
def testinteger(n):
  if n > 0:
    print("The integer", n , "is positive.")
  elif n <0:
    print("The integer" , n , "is negative.")
  else:
    print("The integer" , n , "is zero.")

testinteger(-946)
```

3. Plotting Fractals with Turtle

In [ ]:

```
# These programs are usually run in Python IDLE.
# You must run this cell before the other turtle programs.
# Install Turtle into Google Colab.

!pip install ColabTurtlePlus
from ColabTurtlePlus.Turtle import *
```

In [ ]:

```
# Plot a colour bifurcating fractal tree.
# Edit the program to plot a trifurcating tree.

initializeTurtle()
setheading(90)            # Turtle points up.
penup()
setpos(0 , -250)
pendown()
speed(0)                  # Fastest speed.

def fractal_tree_color(length , level):
    pensize(length / 10)
    if length < 20:
        pencolor("green") # The leaves.
    else:
        pencolor("brown") # The trunk and branches.
    if level > 0:
        fd(length)         # Forward length.
        rt(30)             # Right turn 30 degrees.
        fractal_tree_color(length * 0.7 , level - 1) # Right branches.
        lt(90)             # Left turn 90 degrees.
        fractal_tree_color(length * 0.5 , level - 1) # Left branches.
        rt(60)
        penup()
        bk(length)
        pendown()
fractal_tree_color(200 , 8)
```

4. A-Level Mathematics

Jupyter Notebook: Python for A-Level Mathematics and Beyond

In [ ]:

```
# Solve the quadratic equation: x**2 - 4 * x - 3 = 0.
from sympy import *
x = symbols("x")
solve(x**2 - 4 * x - 3 , x)
```

In [ ]:

```
# Differentiate.
diff(x * cos(x) / exp(x))
```

In [ ]:

```
# Integrate.
integrate(x / exp(x) , (x , 0 , oo))
```

In [ ]:

```
# Infinite series.
summation(1 / x , (x , 1 , oo))
```

In [ ]:

```
# Taylor series expansion.
(exp(x) * sin(x)).series(x , 0 , 10)
```

In [ ]:

```
# Simplfy trigonometric expressions.
trigsimp(cos(x) - cos(x)**3)
```

Numerical Methods

Use the Newton-Raphson method to find the root of $x^3 - 0.9x^2 + 2 = 0$, starting with the point $x_0 = 2$. Give your answer to four decimal places. Recall that:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

In [ ]:

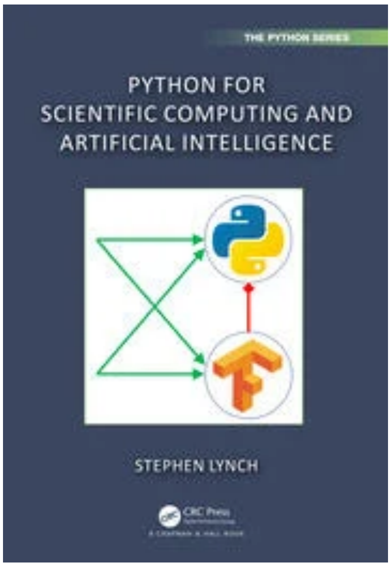
```
# The Newton-Raphson Method
def fn(x):
    return x**3 - 0.9 * x**2 + 2
def dfn(x):
    return 3 * x**2 - 1.8 * x
def NewtonRaphson(x):
    i = 0
    h = fn(x) / dfn(x)
    while abs(h) >= 0.0001:
        h = fn(x) / dfn(x)
        x = x - h
        i += 1              # i = i + 1
        print("x(", i ,")", "is {:8.4f}.".format(x))
# Start at x( 0 ) = 2.
NewtonRaphson(2)
```

5. Animations

In [ ]:

```
# Animation of a Sine Wave - Change the frequency.
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import animation
fig = plt.figure()
ax = plt.axes(xlim=(0, 2) , ylim=(-2 , 2))
line, = ax.plot([] , [] , lw=2)
plt.xlabel("t")
plt.ylabel("$\sin(\omega t)$")
plt.close()
def init():
    line.set_data([],[])
    return line,
# The function to animate. Now frames = 101, and 0 <= i <= 100.
def animate(i):
    t = np.linspace(0 , 2 , 1000)
    y = np.sin(0.1 * i * t)
    line.set_data(t , y)
    return line,
# Change interval to change speed of animation. There are 100 frames in t
anim = animation.FuncAnimation(fig, animate, init_func = init, frames = 1
# The code to produce an animation in html.
from IPython.display import HTML
HTML(anim.to_jshtml())
```

6. New Book: Python for Scientific Computing and Artificial Intelligence



CRC Press 2023: Book URL

Features:

1. No prior experience of programming is required.
2. Online GitHub repository available with codes for readers to practice.
3. Covers applications and examples from biology, chemistry, computer science, data science, electrical and mechanical engineering, economics, mathematics, physics, statistics and binary oscillator computing.
4. Full solutions to exercises are available as Jupyter notebooks on the Web.

Table of Contents:

Section I: AN INTRODUCTION TO PYTHON

Section II: PYTHON FOR SCIENTIFIC COMPUTING

Section III. ARTIFICIAL INTELLIGENCE