5-day Hands-on Workshop on:

Python for Scientific Computing and TensorFlow for Artificial Intelligence

By Dr Stephen Lynch FIMA SFHEA

Holder of Two Patents

Author of PYTHON, MATLAB®, MAPLE™ AND MATHEMATICA® BOOKS

STEM Ambassador and Speaker for Schools

python
TensorFlow

s.lynch@mmu.ac.uk

https://www2.mmu.ac.uk/scmdt/staff/profile/index.php?id=2443

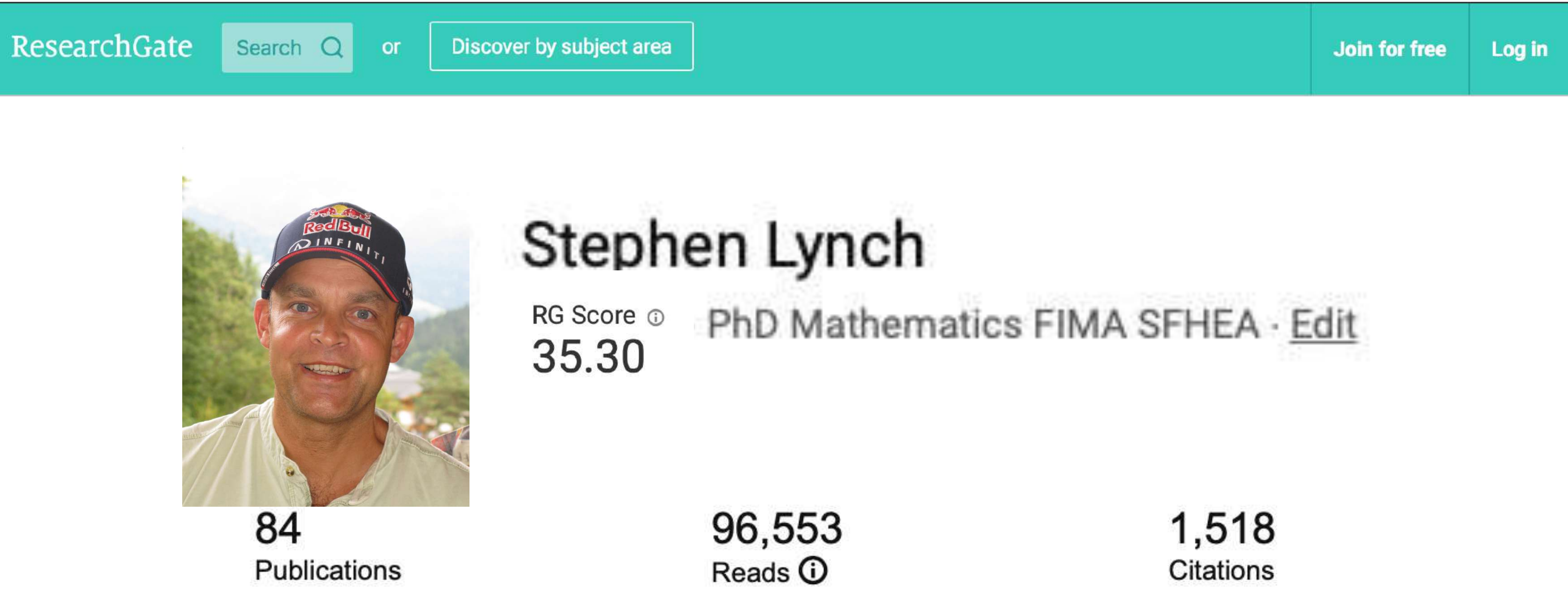
Instructor

Stephen Lynch is a world leader in the use of mathematics packages in teaching, learning, assessment, research and employability. He started using packages in the mid 1980’s whilst studying for his PhD in Pure Mathematics. Upon completion of his PhD, he started his lecturing career at Southampton University at the age of 24.

He has authored 2 international patents for inventions, 7 books, 4 book chapters, over 40 journal articles and a few conference proceedings.

Stephen is a Fellow of the Institute of Mathematics and Its Applications (FIMA), a Senior Fellow of the Higher Education Academy (SFHEA), a Reader with Manchester Metropolitan University and was concurrently an Associate Lecturer with the Open University (2008-2012). In 2010, Stephen volunteered as a STEM Ambassador, in 2012, MMU awarded him a Public Engagement Champion award and in 2014 he became a Speaker for Schools. In 2022, Stephen was nominated for a **National Teaching Fellowship** for his work in Widening Participation, programming in the Maths curriculum and his interdisciplinary research feeding in to teaching.

Instructor: ResearchGate



Stephen Lynch

RG Score 35.30

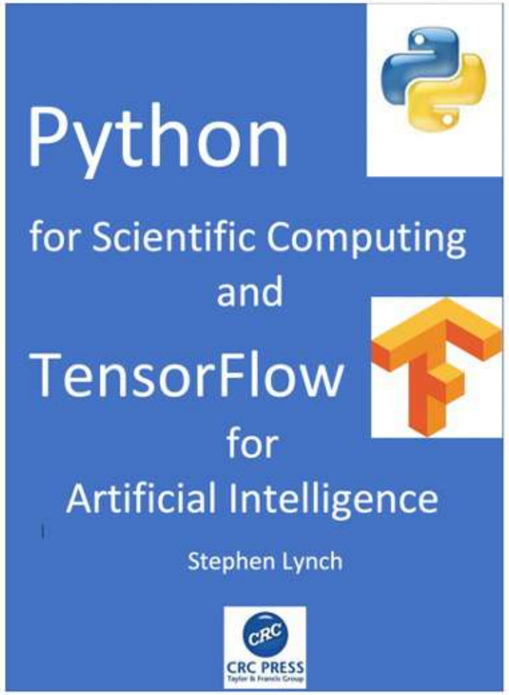
PhD Mathematics FIMA SFHEA · Edit

84 Publications

96,553 Reads

1,518 Citations

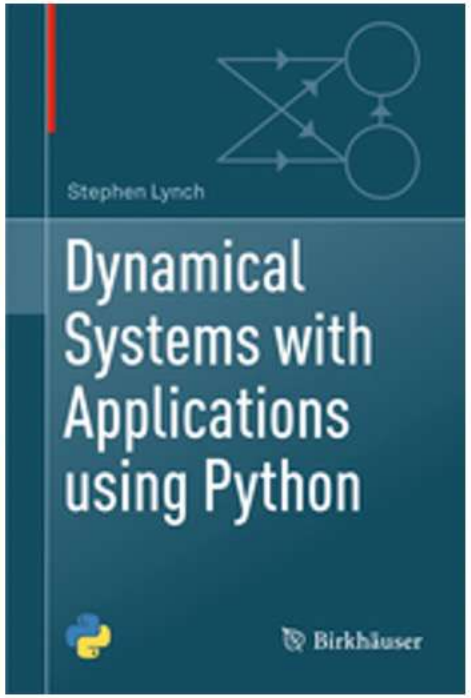
Instructor: Books



Python for Scientific Computing and TensorFlow for Artificial Intelligence

Stephen Lynch

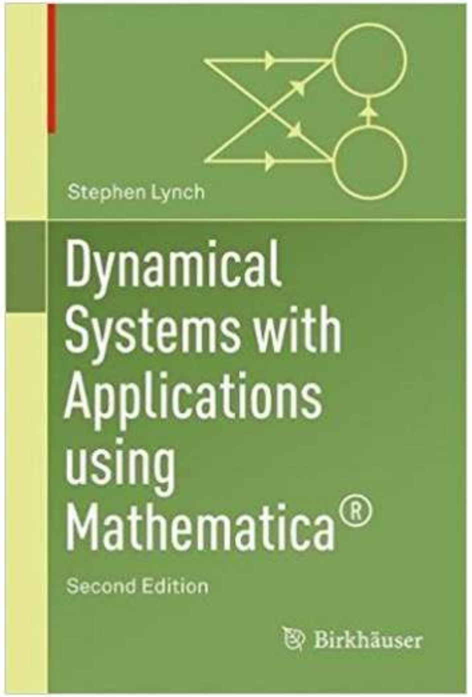
CRC PRESS



Stephen Lynch

Dynamical Systems with Applications using Python

Birkhäuser

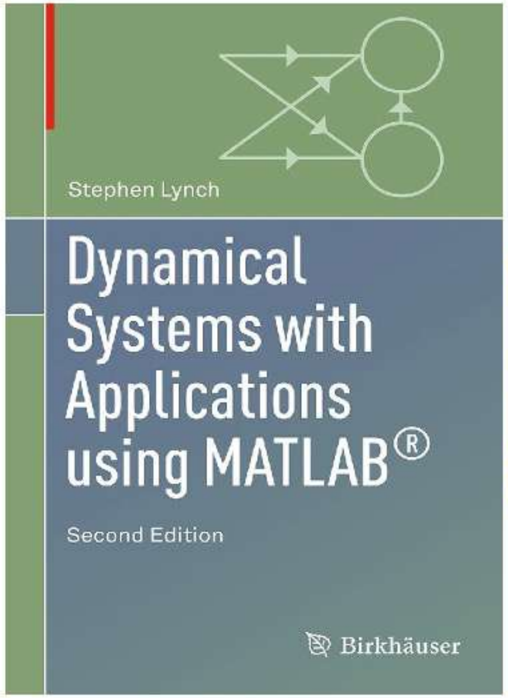


Stephen Lynch

Dynamical Systems with Applications using Mathematica®

Second Edition

Birkhäuser

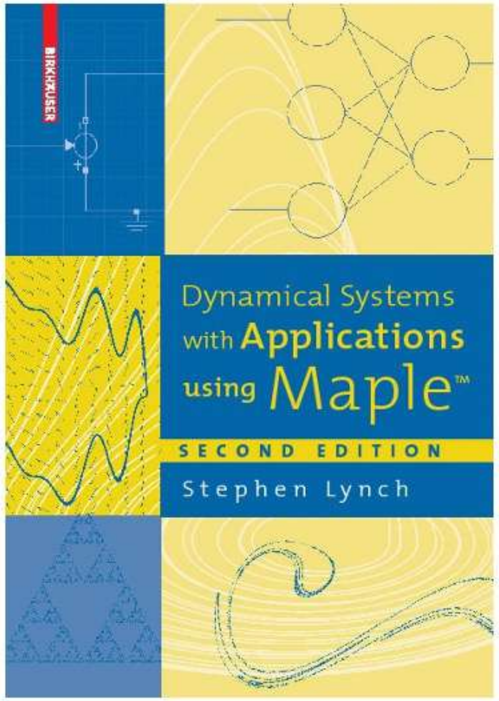


Stephen Lynch

Dynamical Systems with Applications using MATLAB®

Second Edition

Birkhäuser



Dynamical Systems with Applications using Maple™

SECOND EDITION

Stephen Lynch

BIRKHÄUSER

PUBLISHED IN 2023

Chapter downloads
1st Edition (2018): 140,000
Web: Jupyter Notebook

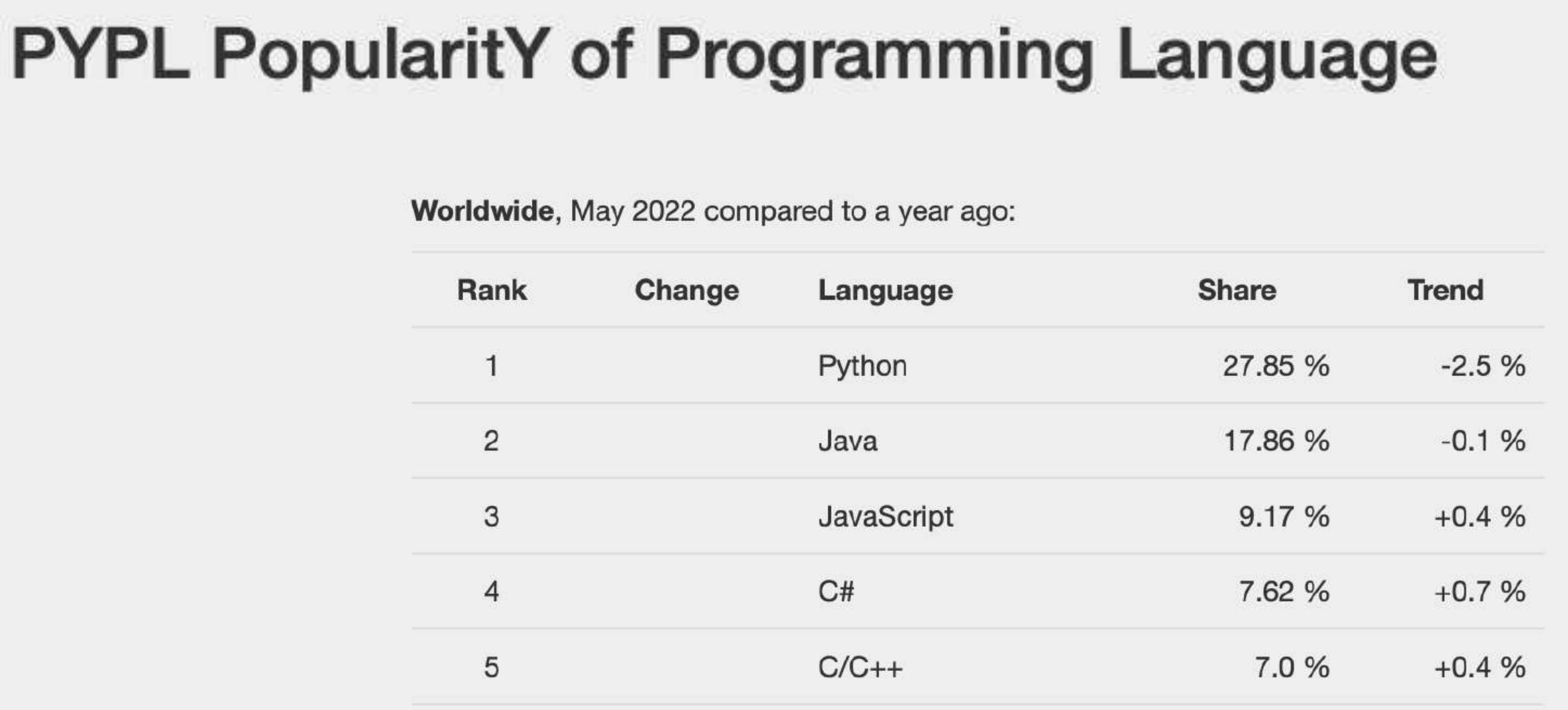
Chapter downloads
2nd Edition (2010): 65,000
1st Edition (2000): 9,000

Chapter downloads
2nd Edition (2014): 240,000
1st Edition (2004): 27,000

Chapter downloads
2nd Edition (2017): 60,000
1st Edition (2004): 72,000

Manchester Metropolitan University

There are over 600 Programming Languages



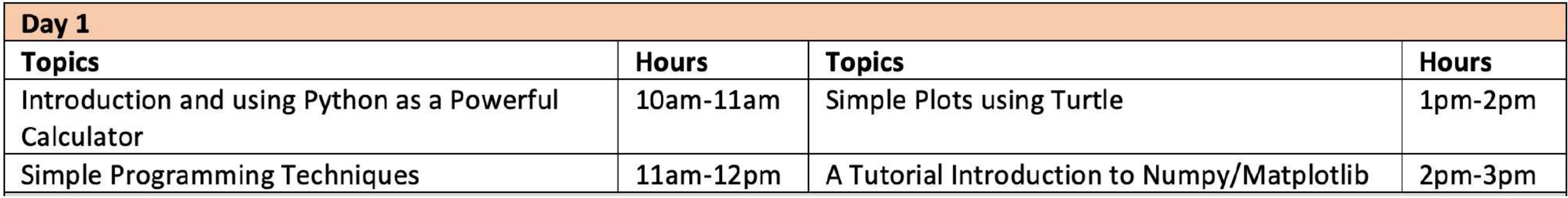
PYPL PopularitY of Programming Language

Worldwide, May 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.85 %	-2.5 %
2		Java	17.86 %	-0.1 %
3		JavaScript	9.17 %	+0.4 %
4		C#	7.62 %	+0.7 %
5		C/C++	7.0 %	+0.4 %

IMA Maths Careers (2021): Python for A-Level Maths, Undergraduate Maths and Employability
https://www.mathscareers.org.uk/python-for-a-level-maths-undergraduate-maths-and-employability/

Schedule (Day 1)



Day 1			
Topics	**Hours**	**Topics**	**Hours**
Introduction and using Python as a Powerful Calculator	10am-11am	Simple Plots using Turtle	1pm-2pm
Simple Programming Techniques	11am-12pm	A Tutorial Introduction to Numpy/Matplotlib	2pm-3pm

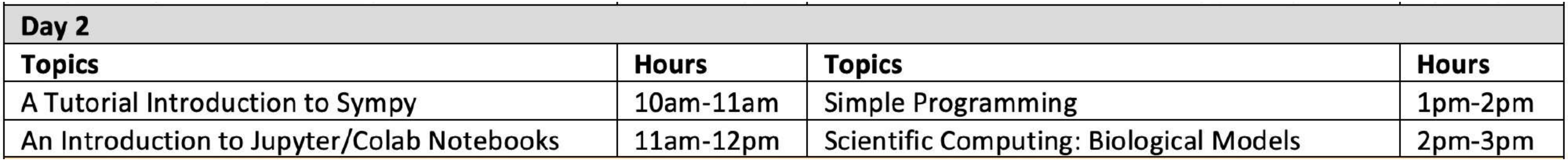
Download all files from GitHub:

https://github.com/DrStephenLynch/Tekbac

See Jupyter Notebook here:

http://www.doc.mmu.ac.uk/STAFF/S.Lynch/DSAP_Jupyter_Notebook.html

Schedule (Day 2)



Day 2			
Topics	**Hours**	**Topics**	**Hours**
A Tutorial Introduction to Sympy	10am-11am	Simple Programming	1pm-2pm
An Introduction to Jupyter/Colab Notebooks	11am-12pm	Scientific Computing: Biological Models	2pm-3pm

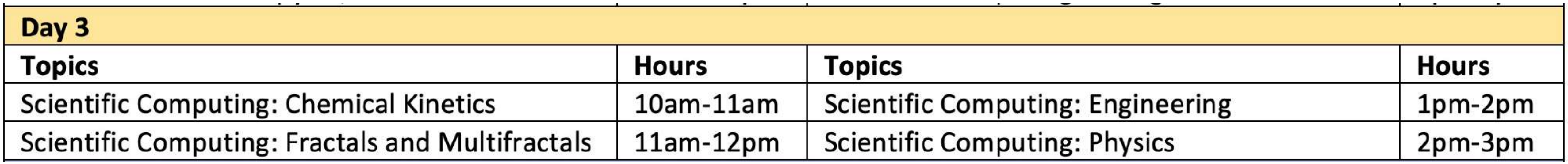
Download files from GitHub:

https://github.com/DrStephenLynch/Tekbac

See Jupyter Notebook here:

http://www.doc.mmu.ac.uk/STAFF/S.Lynch/DSAP_Jupyter_Notebook.html

Schedule (Day 3)



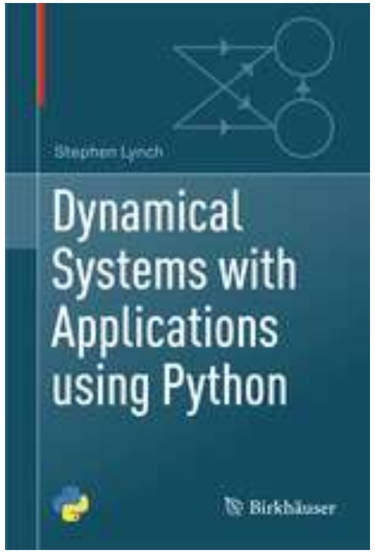
Day 3			
Topics	**Hours**	**Topics**	**Hours**
Scientific Computing: Chemical Kinetics	10am-11am	Scientific Computing: Engineering	1pm-2pm
Scientific Computing: Fractals and Multifractals	11am-12pm	Scientific Computing: Physics	2pm-3pm

Download files from GitHub:

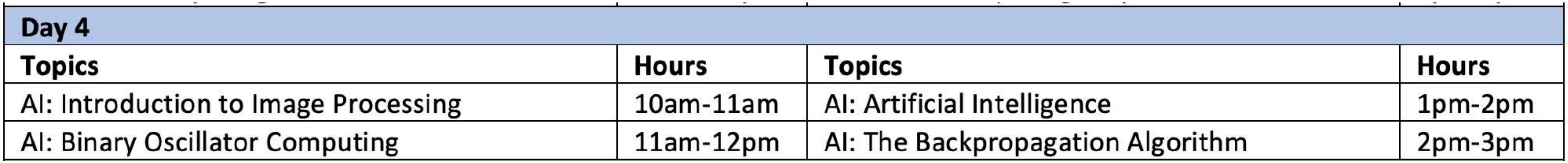
https://github.com/DrStephenLynch/Tekbac

See Jupyter Notebook here:

http://www.doc.mmu.ac.uk/STAFF/S.Lynch/DSAP_Jupyter_Notebook.html



Schedule (Day 4)



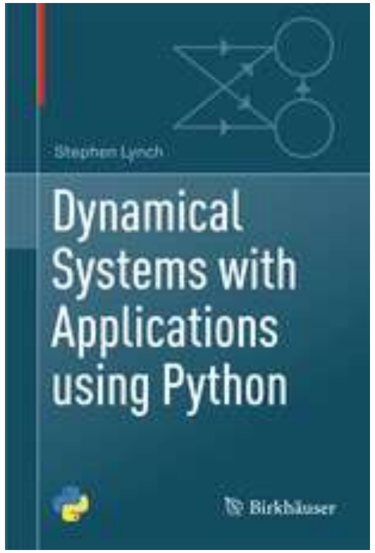
Day 4			
Topics	**Hours**	**Topics**	**Hours**
AI: Introduction to Image Processing	10am-11am	AI: Artificial Intelligence	1pm-2pm
AI: Binary Oscillator Computing	11am-12pm	AI: The Backpropagation Algorithm	2pm-3pm

Download files from GitHub:

https://github.com/DrStephenLynch/Tekbac

See Jupyter Notebook here:

http://www.doc.mmu.ac.uk/STAFF/S.Lynch/DSAP_Jupyter_Notebook.html

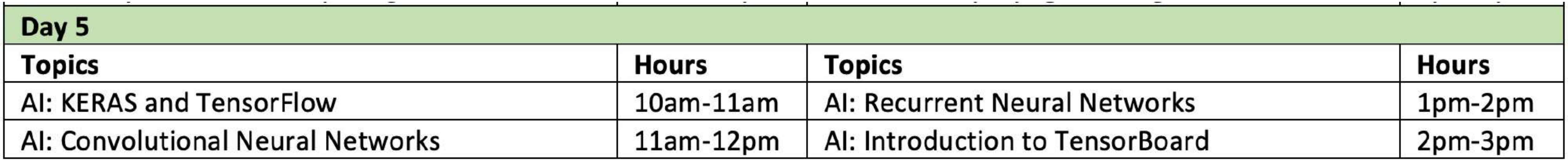


Stephen Lynch

Dynamical Systems with Applications using Python

Birkhäuser

Schedule (Day 5)



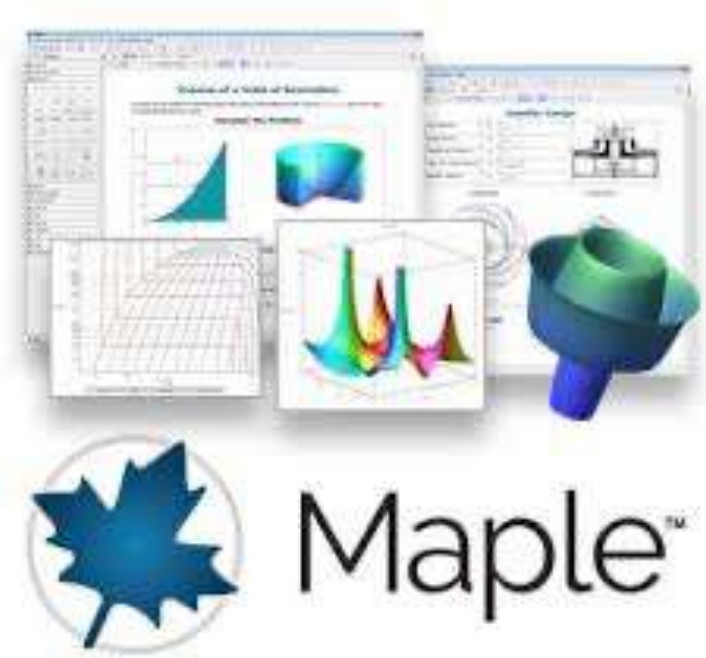
Day 5			
Topics	**Hours**	**Topics**	**Hours**
AI: KERAS and TensorFlow	10am-11am	AI: Recurrent Neural Networks	1pm-2pm
AI: Convolutional Neural Networks	11am-12pm	AI: Introduction to TensorBoard	2pm-3pm

Download files from GitHub:

https://github.com/DrStephenLynch/Tekbac

Application Programming Interface (API)

Maple, MATLAB and Mathematica: Symbolic Computation and Modelling



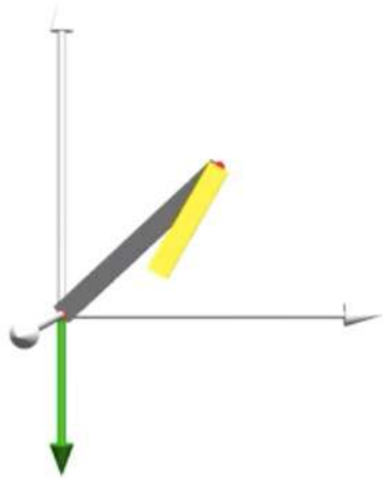
https://www.maplesoft.com

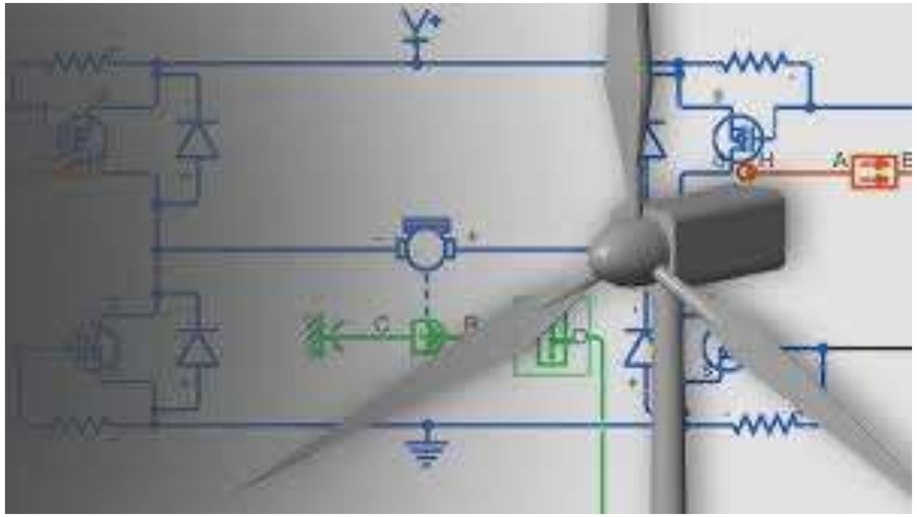
https://www.mathworks.com

https://www.wolfram.com/mathematica/

Maple, MATLAB and Mathematica for SIMULATION

MATLAB®
Simulink®
Simscape®





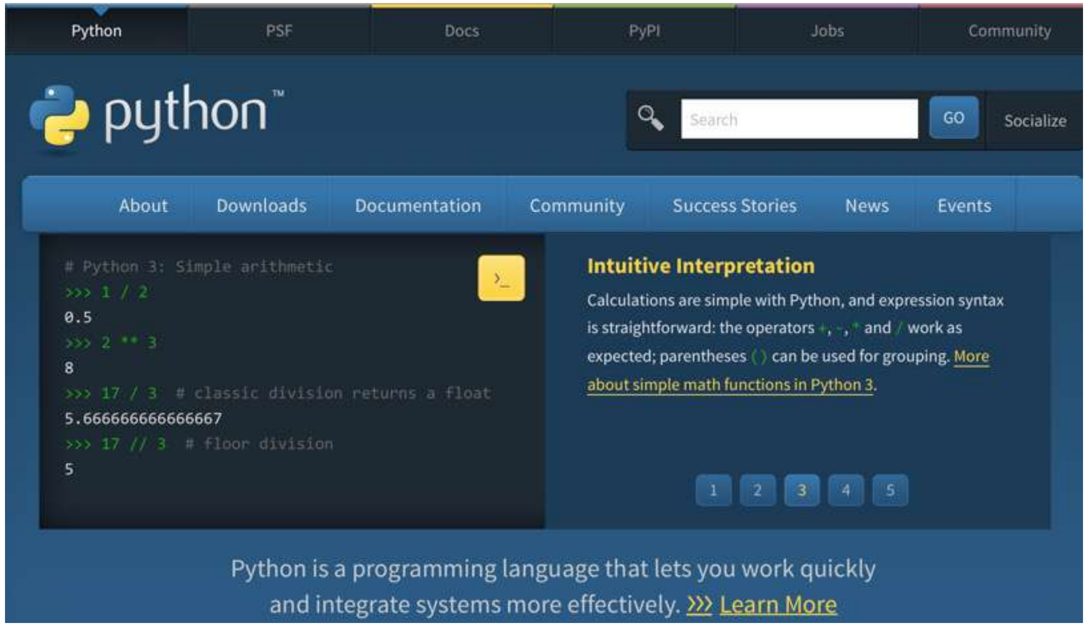
MapleSim™
Advanced System-Level Modeling

Wolfram
SystemModeler™

Download URLs for Python

1. To download Python (IDLE):

https://www.python.org/



```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>> 17 // 3  # floor division
5
```

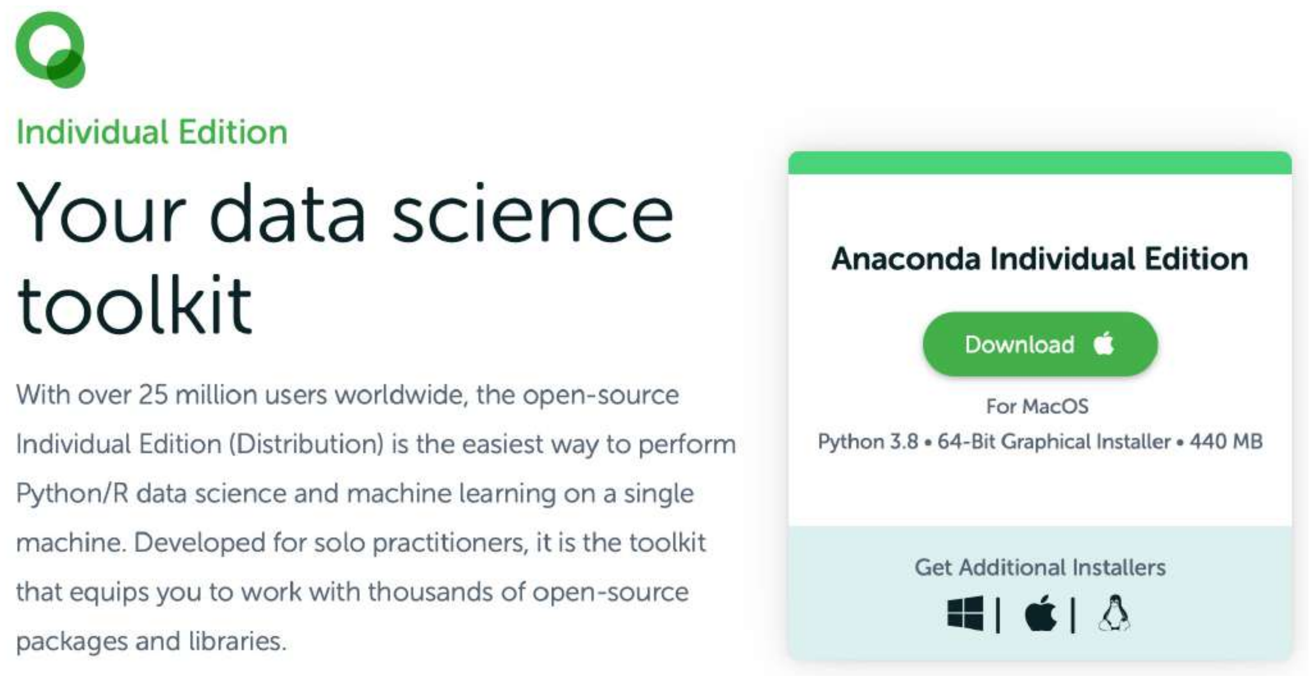
Intuitive Interpretation

Calculations are simple with Python, and expression syntax is straightforward: the operators +, -, * and / work as expected; parentheses ( ) can be used for grouping. More about simple math functions in Python 3.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

2. To download Anaconda:

https://www.anaconda.com/products/individual



Individual Edition

Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

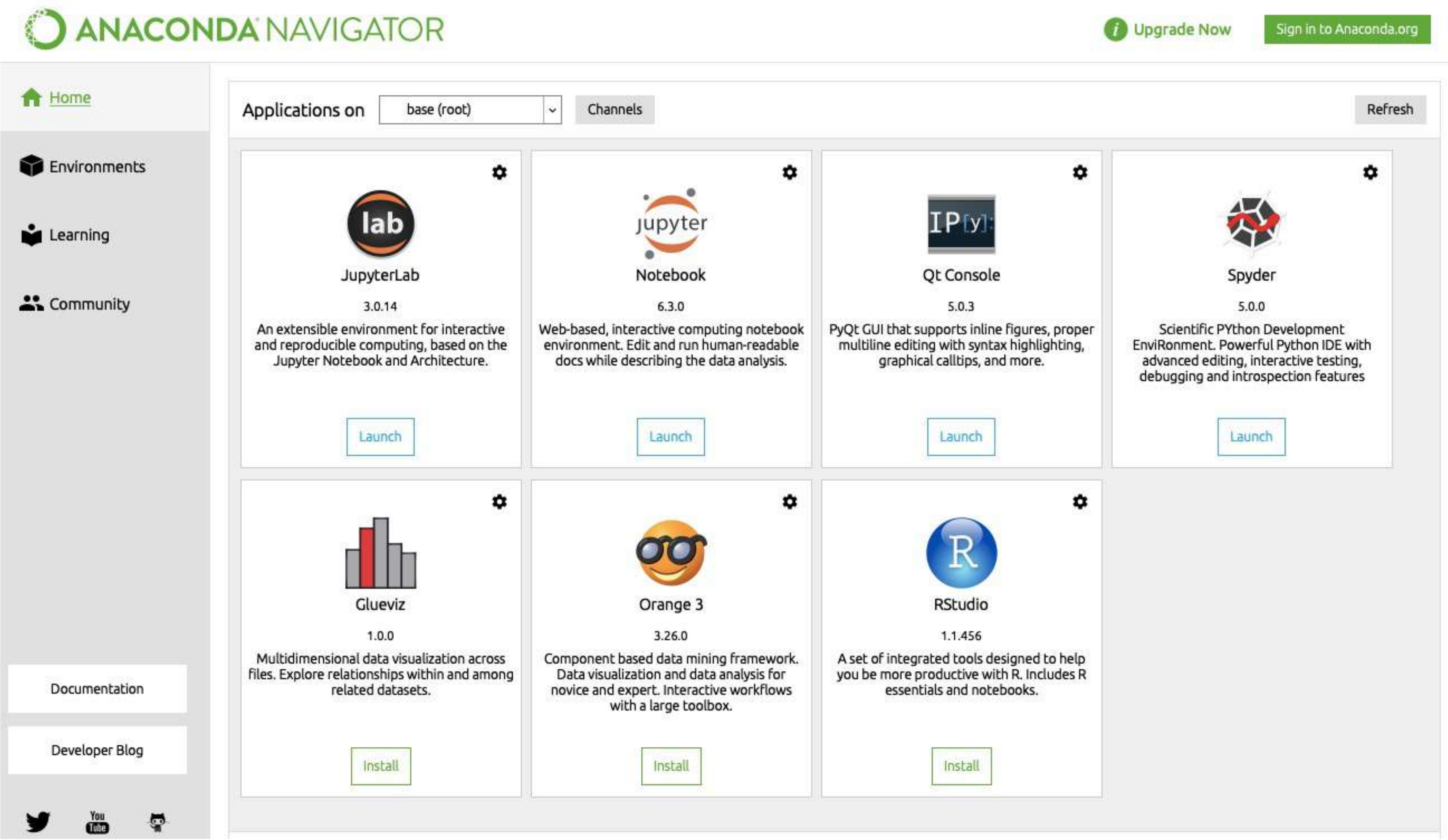
Anaconda Individual Edition

For MacOS

Python 3.8 • 64-Bit Graphical Installer • 440 MB

Manchester Metropolitan University

Anaconda Free Package Manager



Applications on base (root)

JupyterLab 3.0.14
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook 6.3.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Qt Console 5.0.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

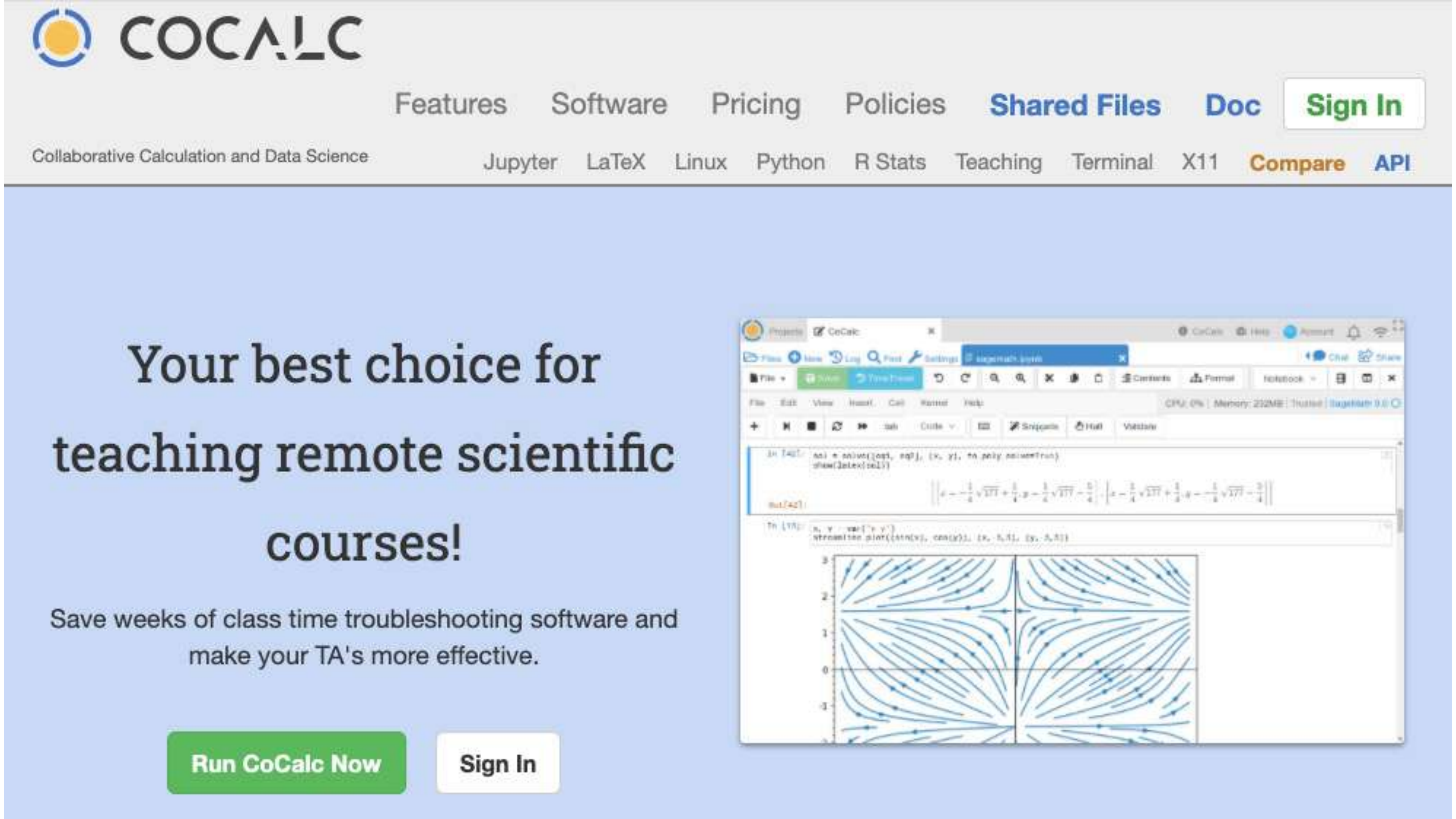
Spyder 5.0.0
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

Glueviz 1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install

Orange 3 3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install

RStudio 1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install

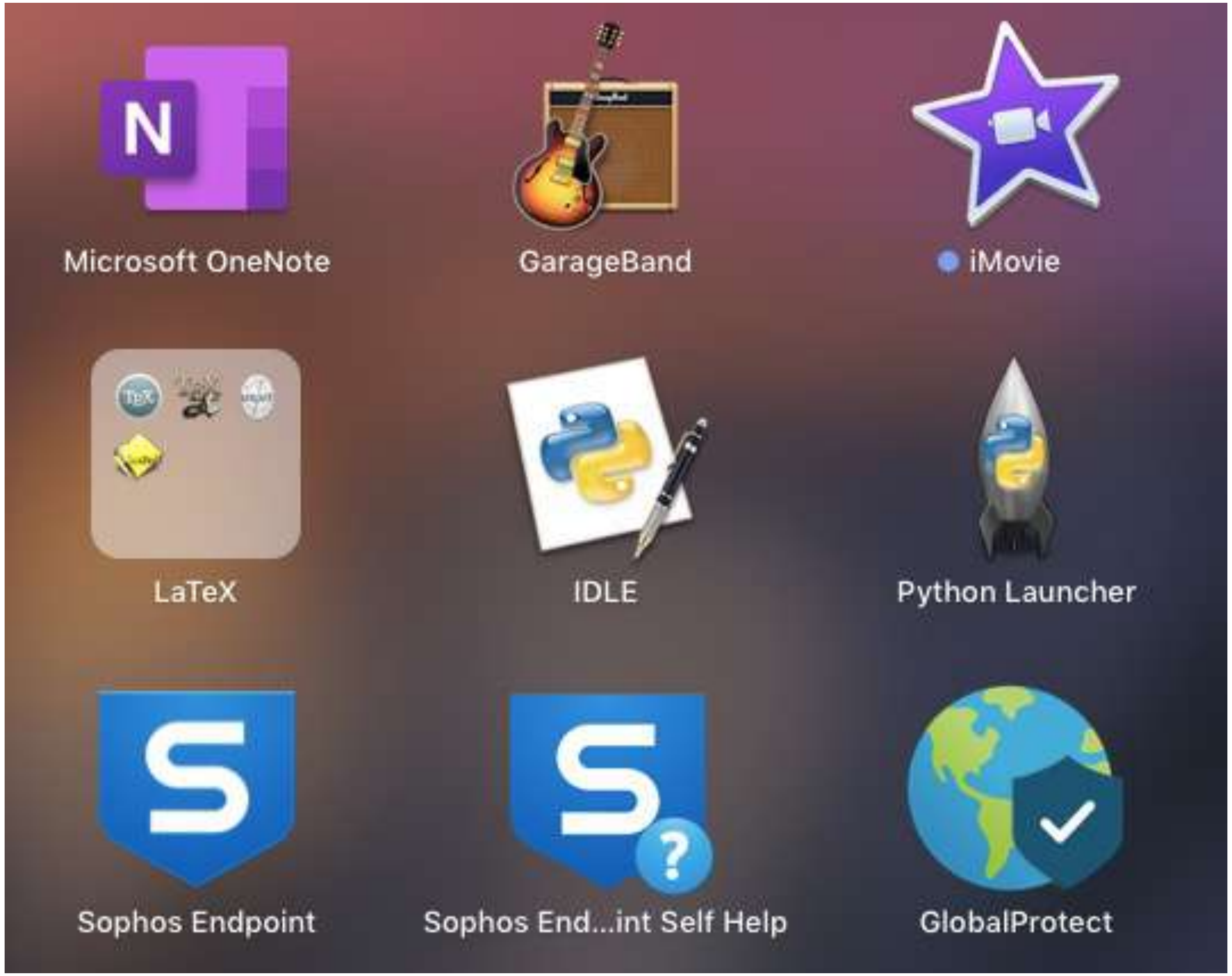
COCALC: A Virtual Online Workspace



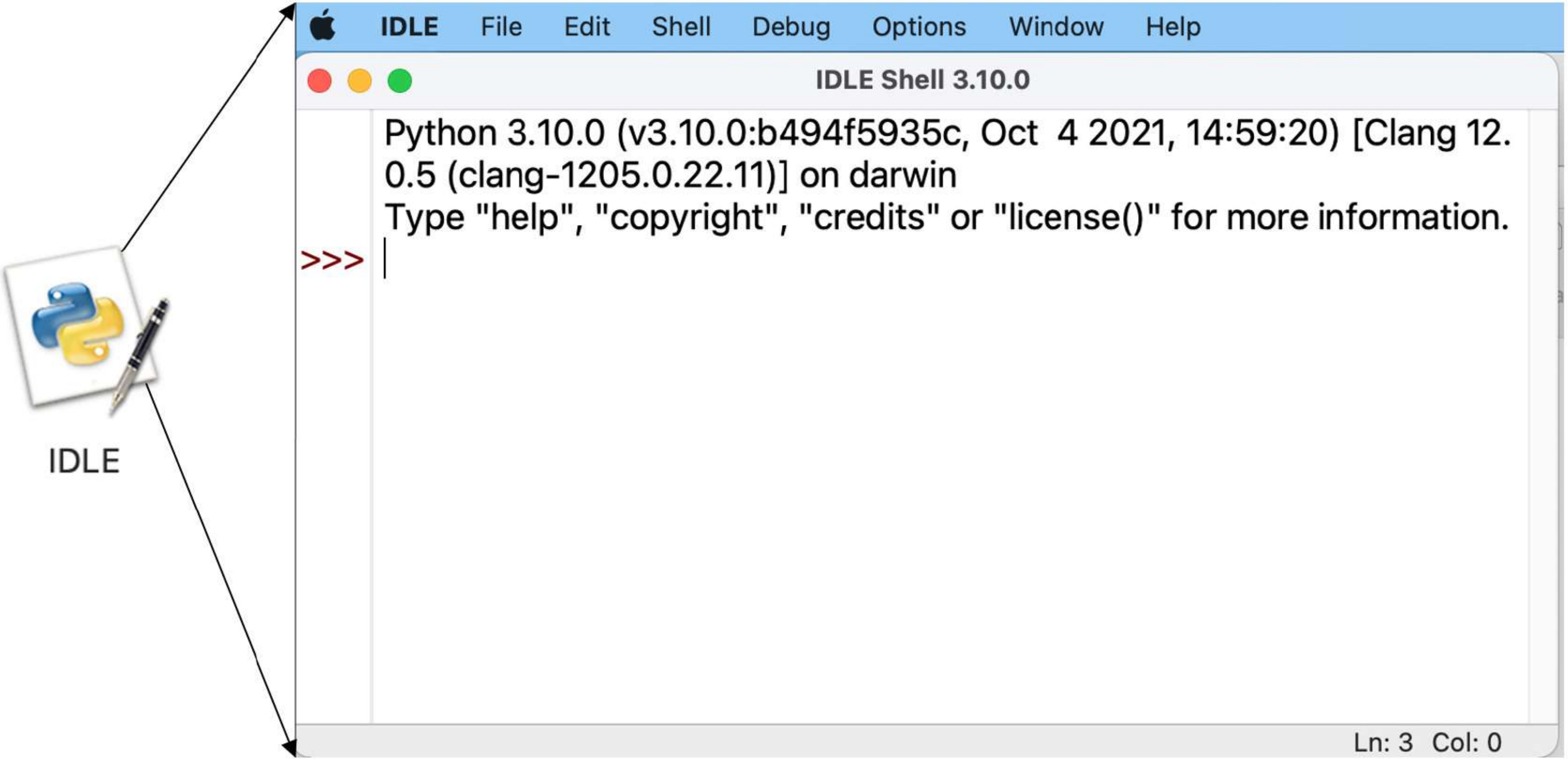
Your best choice for teaching remote scientific courses!

Save weeks of class time troubleshooting software and make your TA's more effective.

Let’s Start with Python IDLE: Mac Launchpad

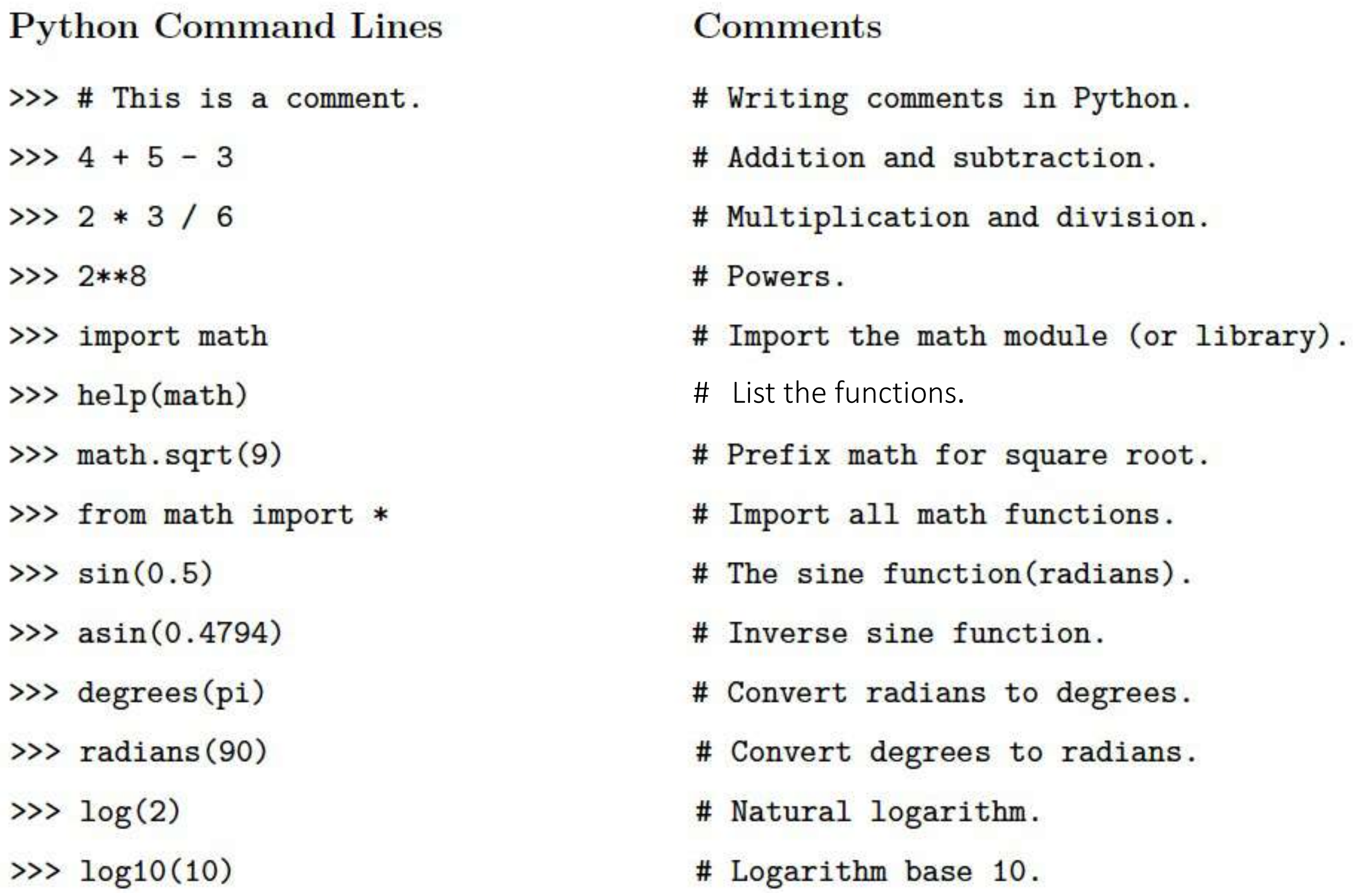


The IDLE (Integrated Development Learning Environment) Editor Window



```
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:20) [Clang 12.
0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

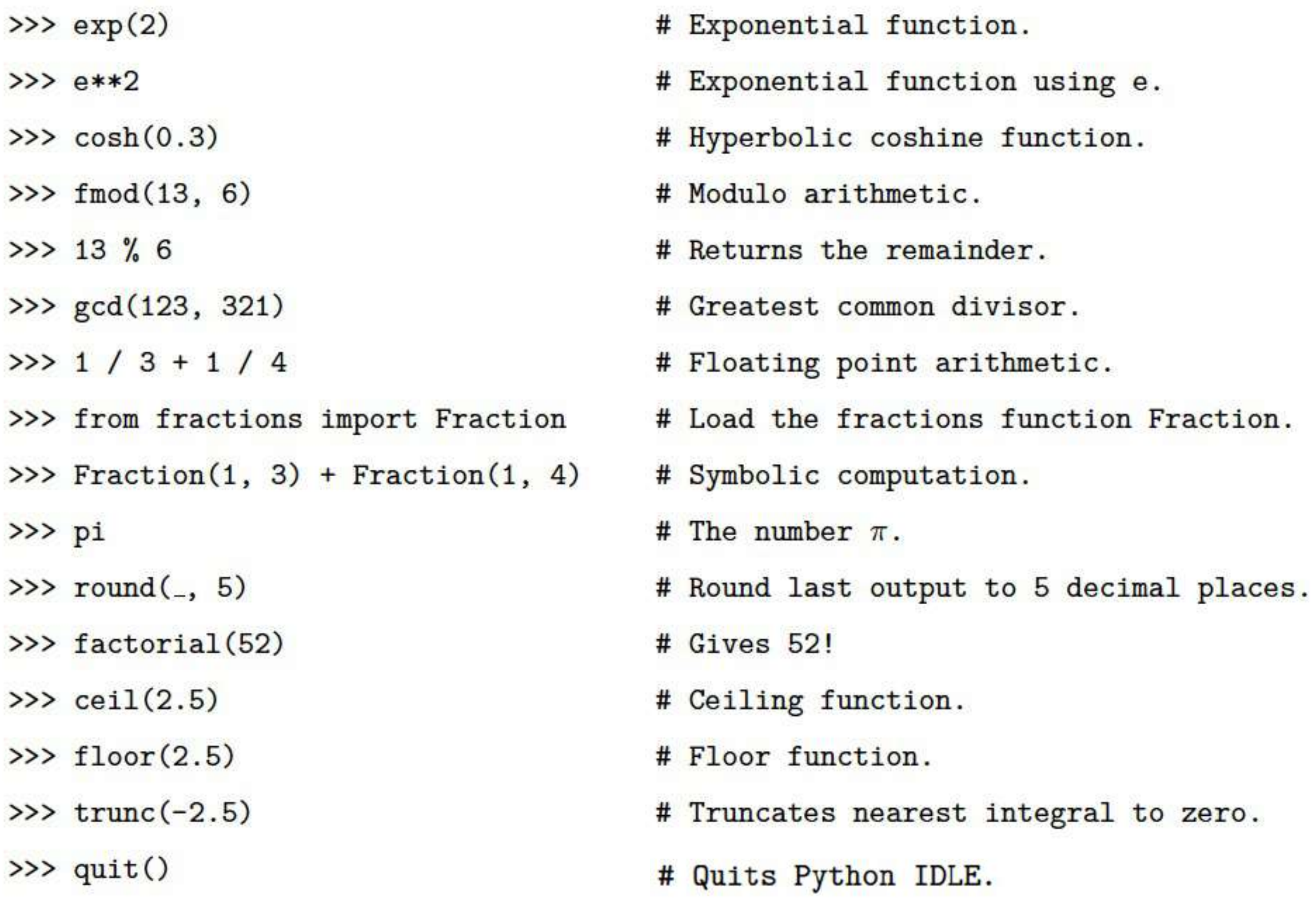
Using Python as a Powerful Calculator



Python Command Lines	Comments
`>>> # This is a comment.`	# Writing comments in Python.
`>>> 4 + 5 - 3`	# Addition and subtraction.
`>>> 2 * 3 / 6`	# Multiplication and division.
`>>> 2**8`	# Powers.
`>>> import math`	# Import the math module (or library).
`>>> help(math)`	# List the functions.
`>>> math.sqrt(9)`	# Prefix math for square root.
`>>> from math import *`	# Import all math functions.
`>>> sin(0.5)`	# The sine function(radians).
`>>> asin(0.4794)`	# Inverse sine function.
`>>> degrees(pi)`	# Convert radians to degrees.
`>>> radians(90)`	# Convert degrees to radians.
`>>> log(2)`	# Natural logarithm.
`>>> log10(10)`	# Logarithm base 10.

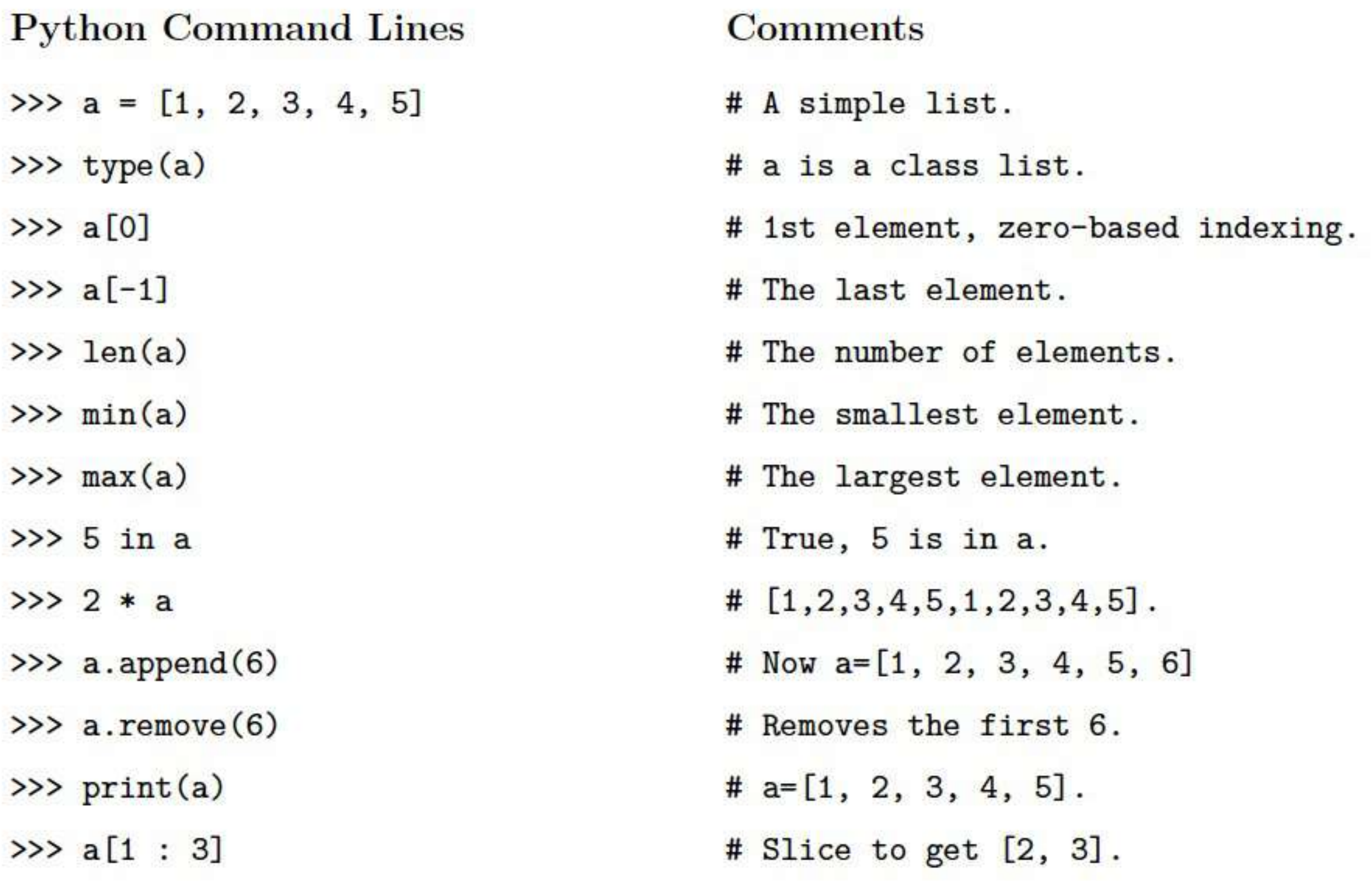
Manchester Metropolitan University

Using Python as a Powerful Calculator



```
>>> exp(2)                                # Exponential function.
>>> e**2                                  # Exponential function using e.
>>> cosh(0.3)                             # Hyperbolic coshine function.
>>> fmod(13, 6)                           # Modulo arithmetic.
>>> 13 % 6                                # Returns the remainder.
>>> gcd(123, 321)                         # Greatest common divisor.
>>> 1 / 3 + 1 / 4                         # Floating point arithmetic.
>>> from fractions import Fraction        # Load the fractions function Fraction.
>>> Fraction(1, 3) + Fraction(1, 4)       # Symbolic computation.
>>> pi                                    # The number π.
>>> round(_, 5)                           # Round last output to 5 decimal places.
>>> factorial(52)                         # Gives 52!
>>> ceil(2.5)                             # Ceiling function.
>>> floor(2.5)                            # Floor function.
>>> trunc(-2.5)                           # Truncates nearest integral to zero.
>>> quit()                                # Quits Python IDLE.
```

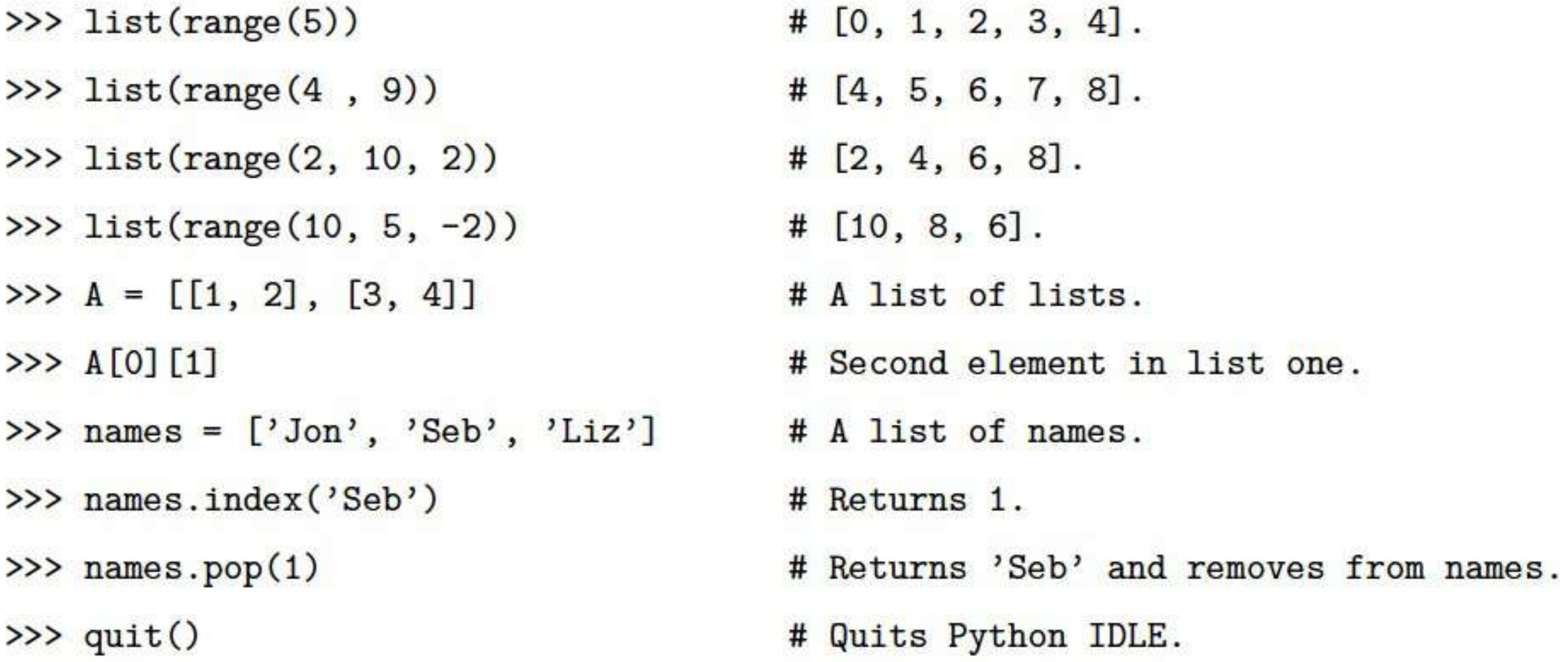
Using Python as a Powerful Calculator (Lists)



Python Command Lines	Comments
>>> a = [1, 2, 3, 4, 5]	# A simple list.
>>> type(a)	# a is a class list.
>>> a[0]	# 1st element, zero-based indexing.
>>> a[-1]	# The last element.
>>> len(a)	# The number of elements.
>>> min(a)	# The smallest element.
>>> max(a)	# The largest element.
>>> 5 in a	# True, 5 is in a.
>>> 2 * a	# [1,2,3,4,5,1,2,3,4,5].
>>> a.append(6)	# Now a=[1, 2, 3, 4, 5, 6]
>>> a.remove(6)	# Removes the first 6.
>>> print(a)	# a=[1, 2, 3, 4, 5].
>>> a[1 : 3]	# Slice to get [2, 3].

Using Python as a Powerful Calculator (Lists) (END SESSION 1)

```
>>> a[1:]                              # Slice to get [2, 3, 4, 5]
>>> a[:-2]                             # Slice to get [1, 2, 3]
```



```
>>> list(range(5))                     # [0, 1, 2, 3, 4].
>>> list(range(4 , 9))                 # [4, 5, 6, 7, 8].
>>> list(range(2, 10, 2))              # [2, 4, 6, 8].
>>> list(range(10, 5, -2))             # [10, 8, 6].
>>> A = [[1, 2], [3, 4]]               # A list of lists.
>>> A[0][1]                            # Second element in list one.
>>> names = [’Jon’, ’Seb’, ’Liz’]      # A list of names.
>>> names.index(’Seb’)                 # Returns 1.
>>> names.pop(1)                       # Returns ’Seb’ and removes from names.
>>> quit()                             # Quits Python IDLE.
```

Simple Programming Techniques (Session 2)

We will concentrate on three programming structures:

1. defining functions;

2. for and while loops;

3. if, elif, else constructs.

Programming: Philosophy of the Book

1. The reader is encouraged to learn programming from exemplar programs listed in the book and available to download online.

2. The reader should look up syntax to understand how the programs work.

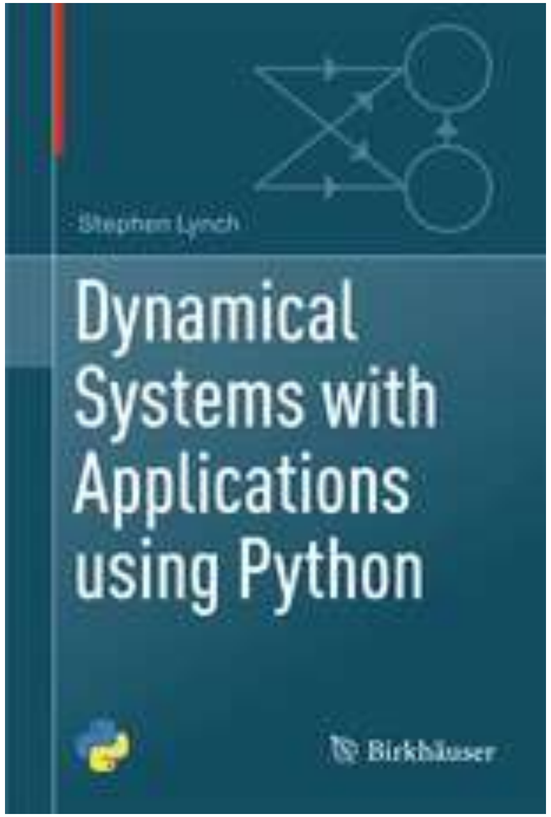
3. The reader should edit working programs before attempting to write their own code from scratch.

Download files from GitHub:

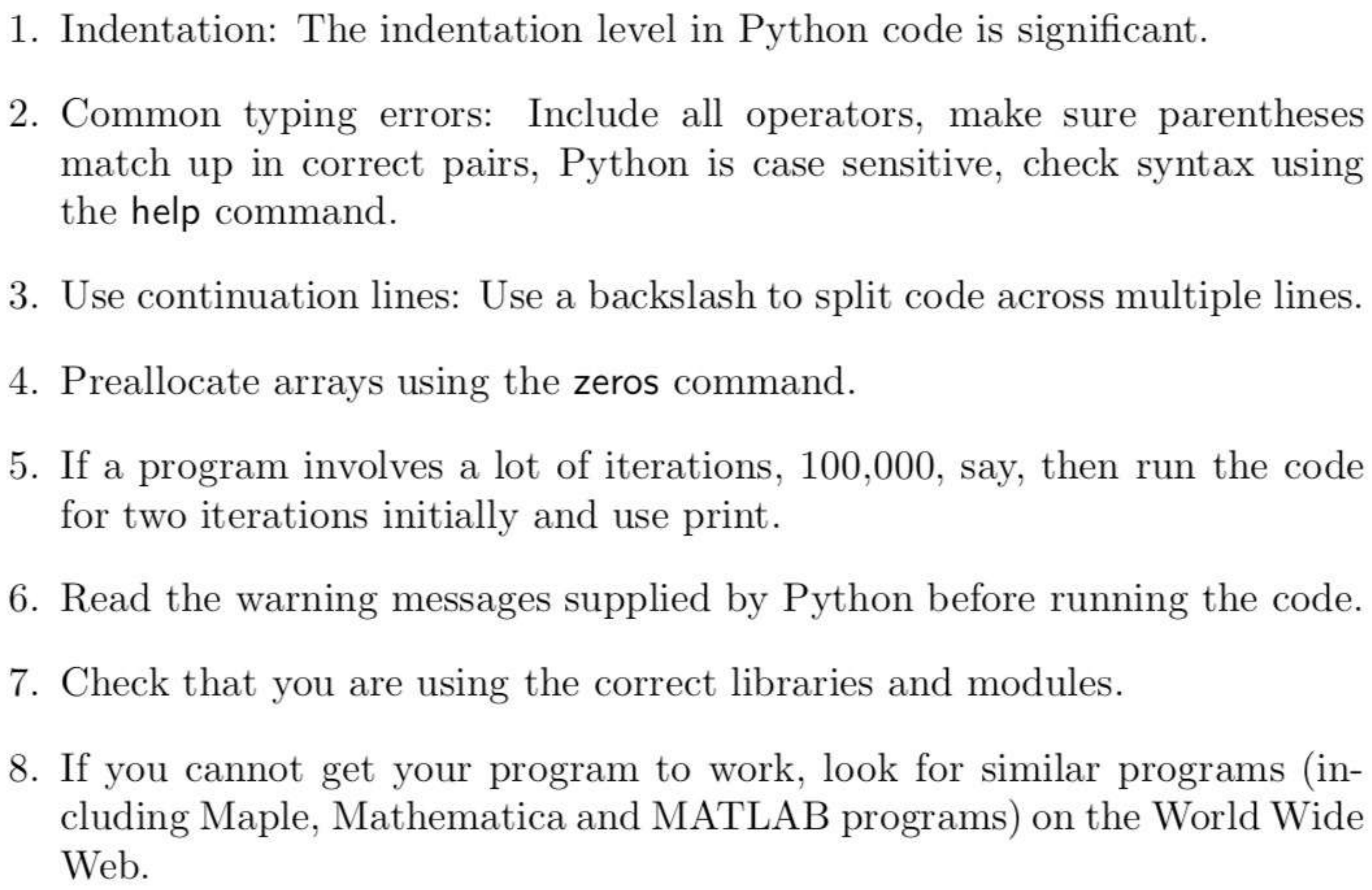
https://github.com/springer-math/dynamical-systems-with-applications-using-python

See the Jupyter Notebook online:

http://www.doc.mmu.ac.uk/STAFF/S.Lynch/DSAP_Jupyter_Notebook.html

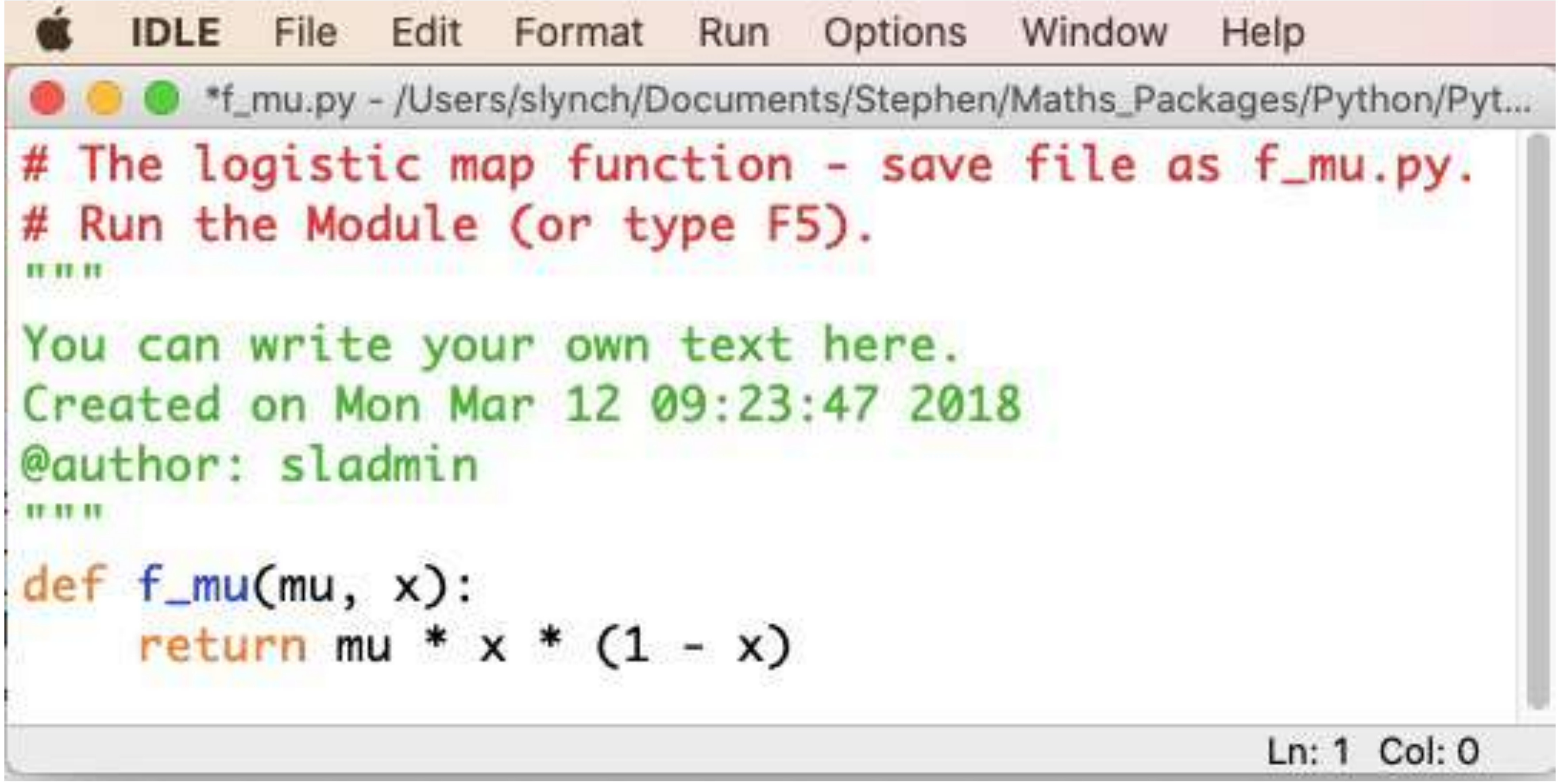


Hints for Programming

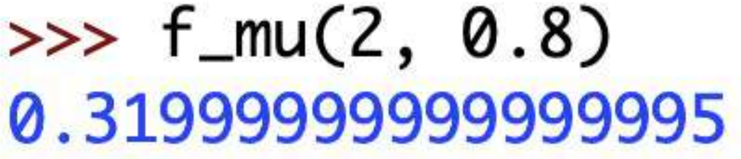


1. Indentation: The indentation level in Python code is significant.
2. Common typing errors: Include all operators, make sure parentheses match up in correct pairs, Python is case sensitive, check syntax using the help command.
3. Use continuation lines: Use a backslash to split code across multiple lines.
4. Preallocate arrays using the zeros command.
5. If a program involves a lot of iterations, 100,000, say, then run the code for two iterations initially and use print.
6. Read the warning messages supplied by Python before running the code.
7. Check that you are using the correct libraries and modules.
8. If you cannot get your program to work, look for similar programs (including Maple, Mathematica and MATLAB programs) on the World Wide Web.

Simple Programming (Functions)

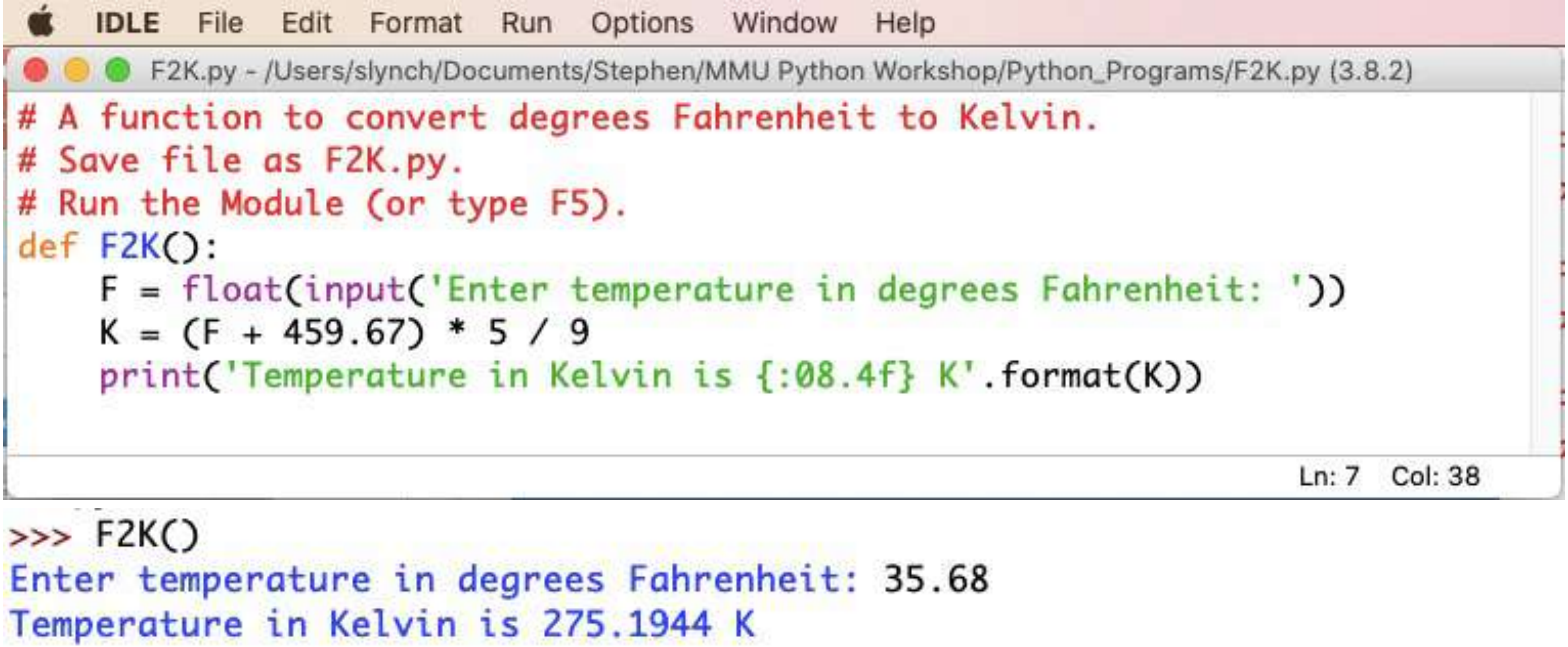

IDLE
File
Edit
Format
Run
Options
Window
Help
*f_mu.py - /Users/slynch/Documents/Stephen/Maths_Packages/Python/Pyt...
Ln: 1
Col: 0

```
# The logistic map function - save file as f_mu.py.
# Run the Module (or type F5).
"""
You can write your own text here.
Created on Mon Mar 12 09:23:47 2018
@author: sladmin
"""
def f_mu(mu, x):
    return mu * x * (1 - x)
```



```
>>> f_mu(2, 0.8)
0.31999999999999995
```

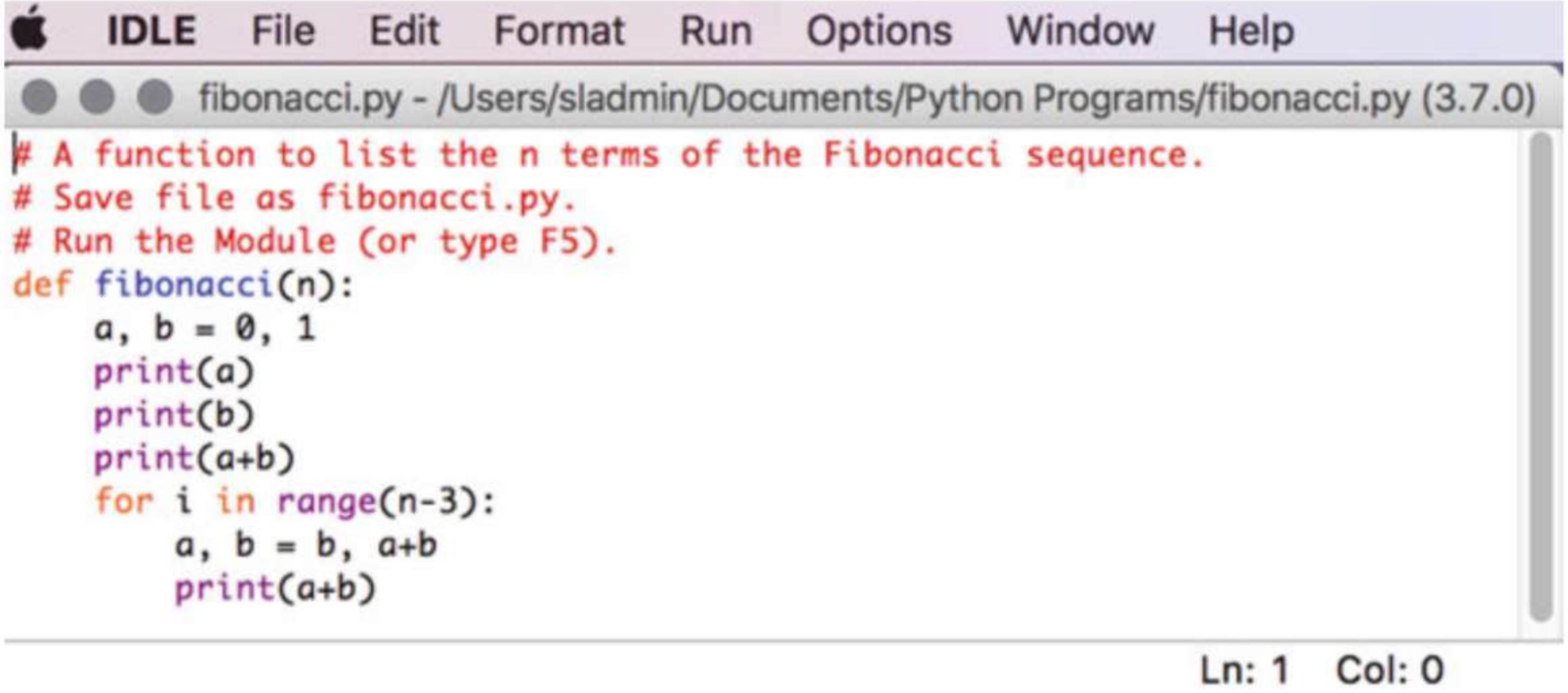
Simple Programming (Functions)


IDLE
File
Edit
Format
Run
Options
Window
Help
F2K.py - /Users/slynch/Documents/Stephen/MMU Python Workshop/Python_Programs/F2K.py (3.8.2)
Ln: 7
Col: 38

```python
# A function to convert degrees Fahrenheit to Kelvin.
# Save file as F2K.py.
# Run the Module (or type F5).
def F2K():
    F = float(input('Enter temperature in degrees Fahrenheit: '))
    K = (F + 459.67) * 5 / 9
    print('Temperature in Kelvin is {:08.4f} K'.format(K))
```

```
>>> F2K()
Enter temperature in degrees Fahrenheit: 35.68
Temperature in Kelvin is 275.1944 K
```

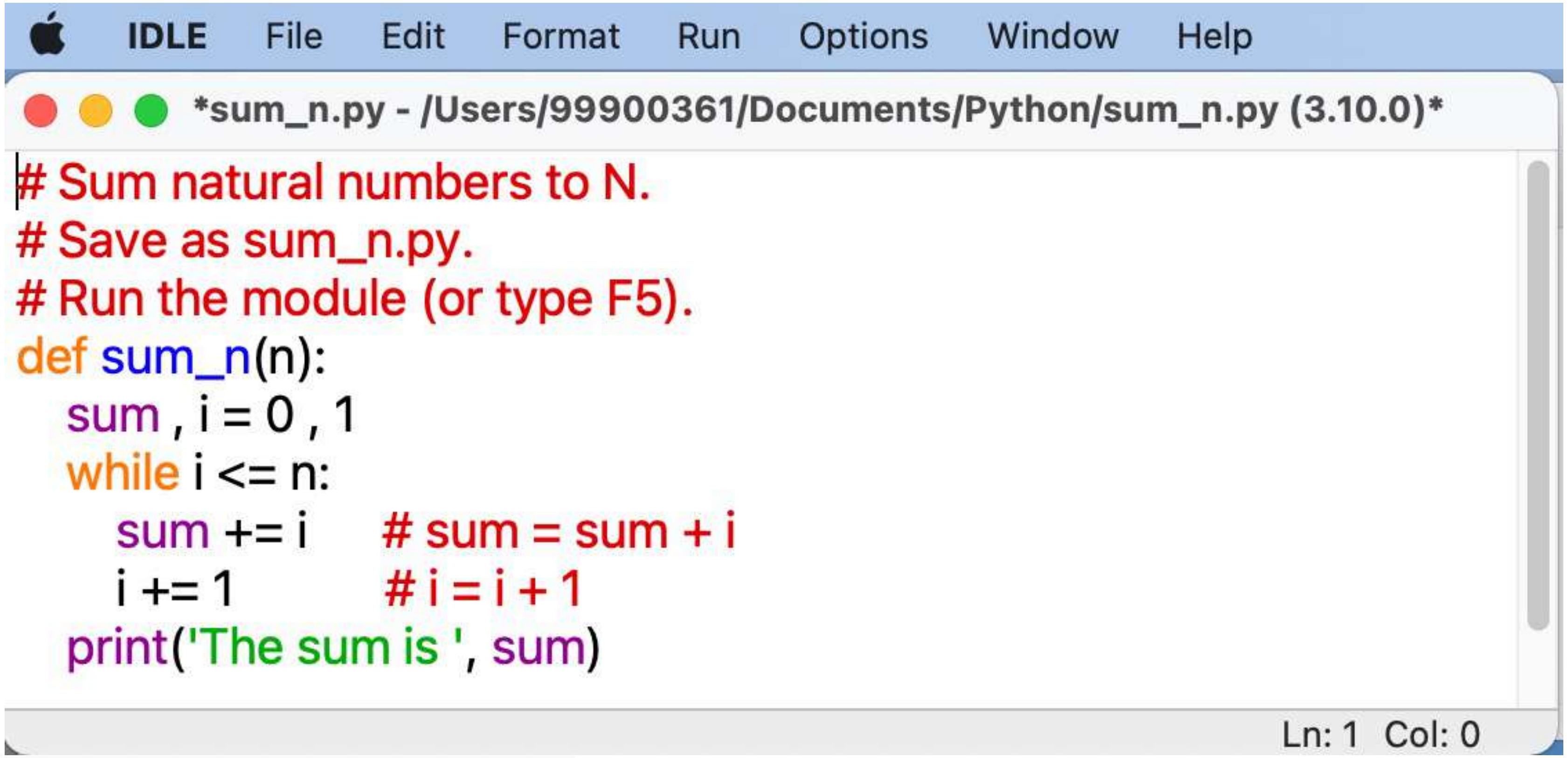
Simple Programming (For Loops)


IDLE
File
Edit
Format
Run
Options
Window
Help
fibonacci.py - /Users/sladmin/Documents/Python Programs/fibonacci.py (3.7.0)
Ln: 1
Col: 0

```
# A function to list the n terms of the Fibonacci sequence.
# Save file as fibonacci.py.
# Run the Module (or type F5).
def fibonacci(n):
    a, b = 0, 1
    print(a)
    print(b)
    print(a+b)
    for i in range(n-3):
        a, b = b, a+b
        print(a+b)
```

```
>>> fibonacci(20)
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181
```

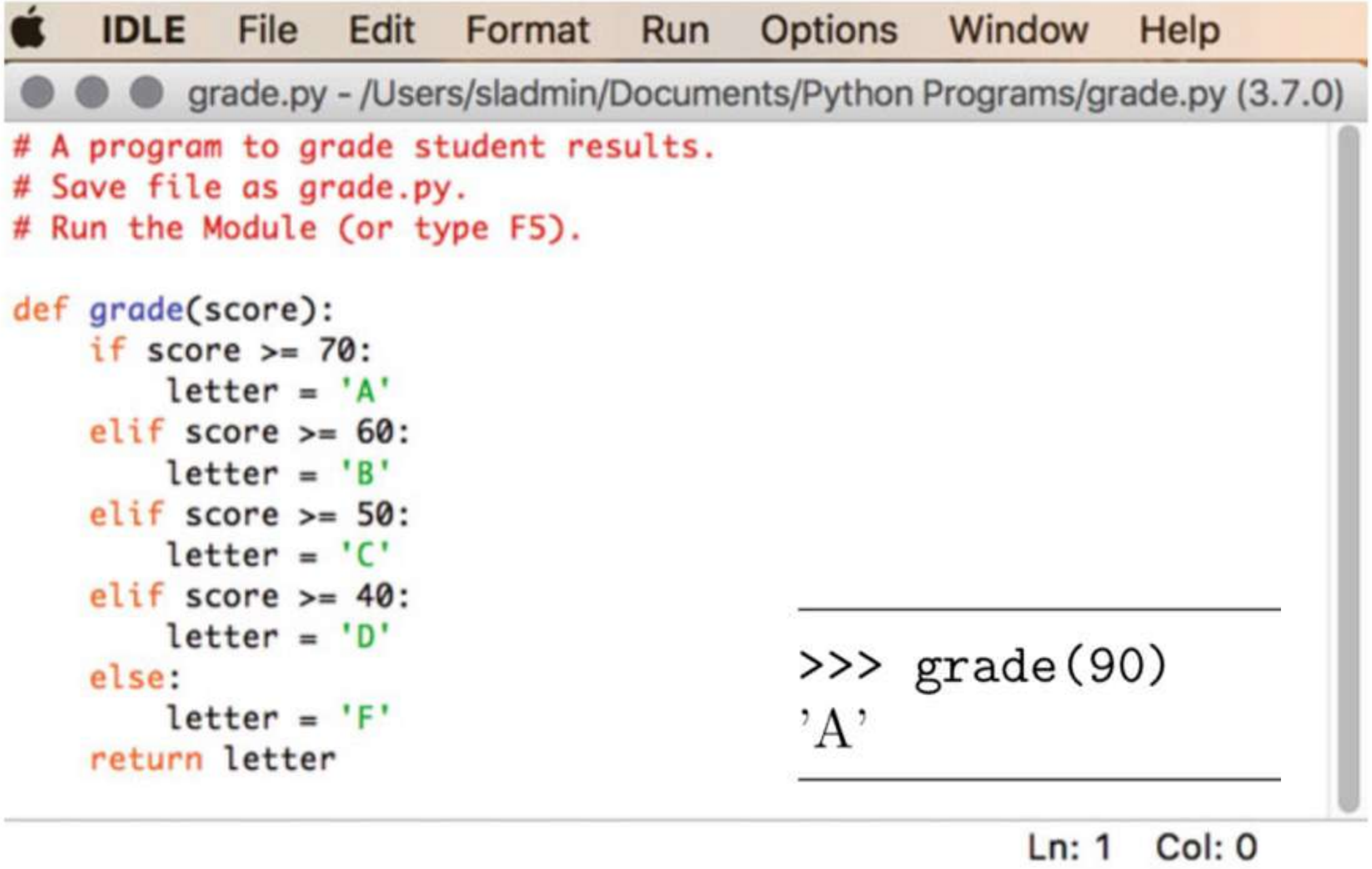
Simple Programming (While Loops)


IDLE
File
Edit
Format
Run
Options
Window
Help
sum_n.py - /Users/99900361/Documents/Python/sum_n.py (3.10.0)
Ln: 1 Col: 0

```
# Sum natural numbers to N.
# Save as sum_n.py.
# Run the module (or type F5).
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)
The sum is 5050
```

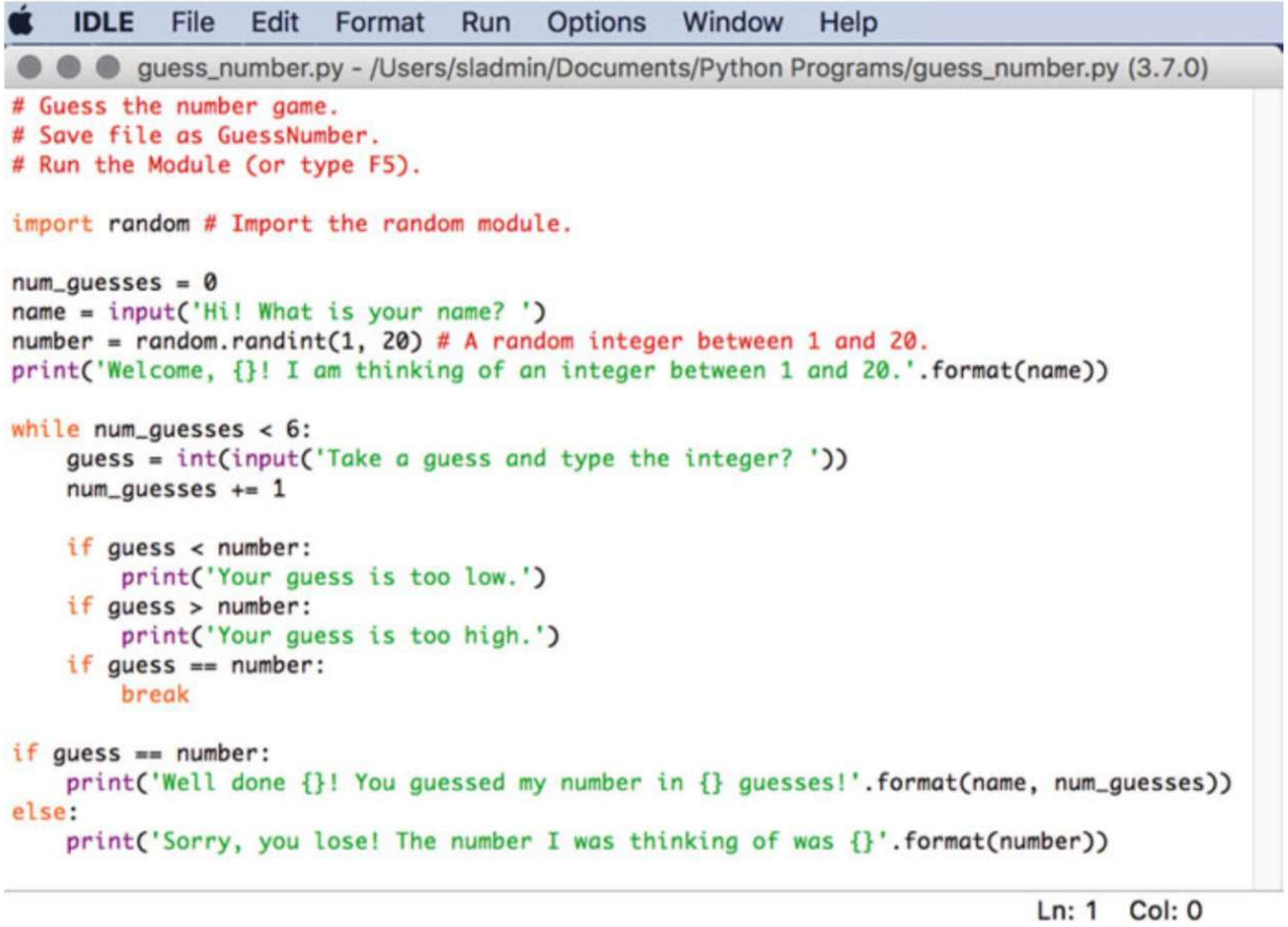
Simple Programming (If, Elif, Else)


IDLE
File
Edit
Format
Run
Options
Window
Help
grade.py - /Users/sladmin/Documents/Python Programs/grade.py (3.7.0)
Ln: 1
Col: 0

```
# A program to grade student results.
# Save file as grade.py.
# Run the Module (or type F5).

def grade(score):
    if score >= 70:
        letter = 'A'
    elif score >= 60:
        letter = 'B'
    elif score >= 50:
        letter = 'C'
    elif score >= 40:
        letter = 'D'
    else:
        letter = 'F'
    return letter
```

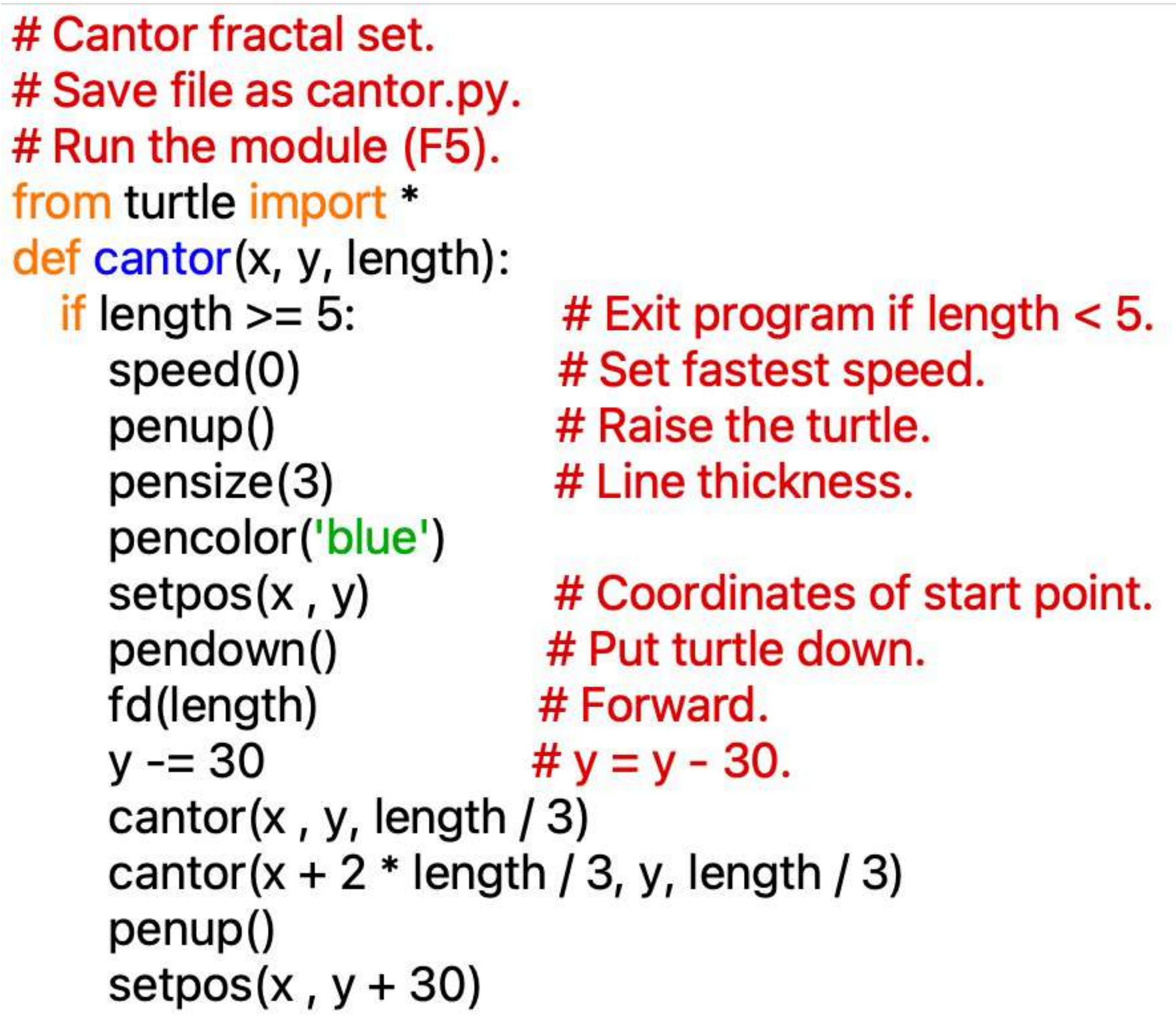
```
>>> grade(90)
'A'
```

Simple Programming (If, Else) (END SESSION 2)


IDLE File Edit Format Run Options Window Help
guess_number.py - /Users/sladmin/Documents/Python Programs/guess_number.py (3.7.0)
Ln: 1 Col: 0

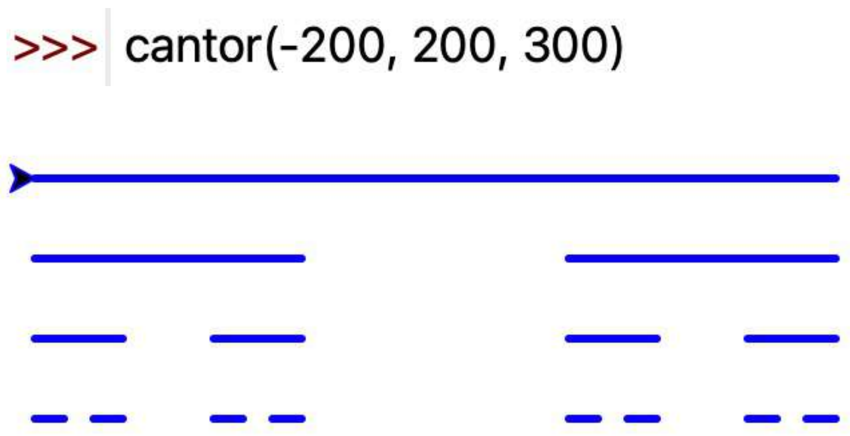
```
# Guess the number game.
# Save file as GuessNumber.
# Run the Module (or type F5).

import random # Import the random module.

num_guesses = 0
name = input('Hi! What is your name? ')
number = random.randint(1, 20) # A random integer between 1 and 20.
print('Welcome, {}! I am thinking of an integer between 1 and 20.'.format(name))

while num_guesses < 6:
    guess = int(input('Take a guess and type the integer? '))
    num_guesses += 1

    if guess < number:
        print('Your guess is too low.')
    if guess > number:
        print('Your guess is too high.')
    if guess == number:
        break

if guess == number:
    print('Well done {}! You guessed my number in {} guesses!'.format(name, num_guesses))
else:
    print('Sorry, you lose! The number I was thinking of was {}'.format(number))
```

The Turtle Module (The Cantor Set): Start Session 3



```
# Cantor fractal set.
# Save file as cantor.py.
# Run the module (F5).
from turtle import *
def cantor(x, y, length):
    if length >= 5:                # Exit program if length < 5.
        speed(0)                   # Set fastest speed.
        penup()                    # Raise the turtle.
        pensize(3)                 # Line thickness.
        pencolor('blue')
        setpos(x , y)              # Coordinates of start point.
        pendown()                  # Put turtle down.
        fd(length)                 # Forward.
        y -= 30                    # y = y - 30.
        cantor(x , y, length / 3)
        cantor(x + 2 * length / 3, y, length / 3)
        penup()
        setpos(x , y + 30)
```

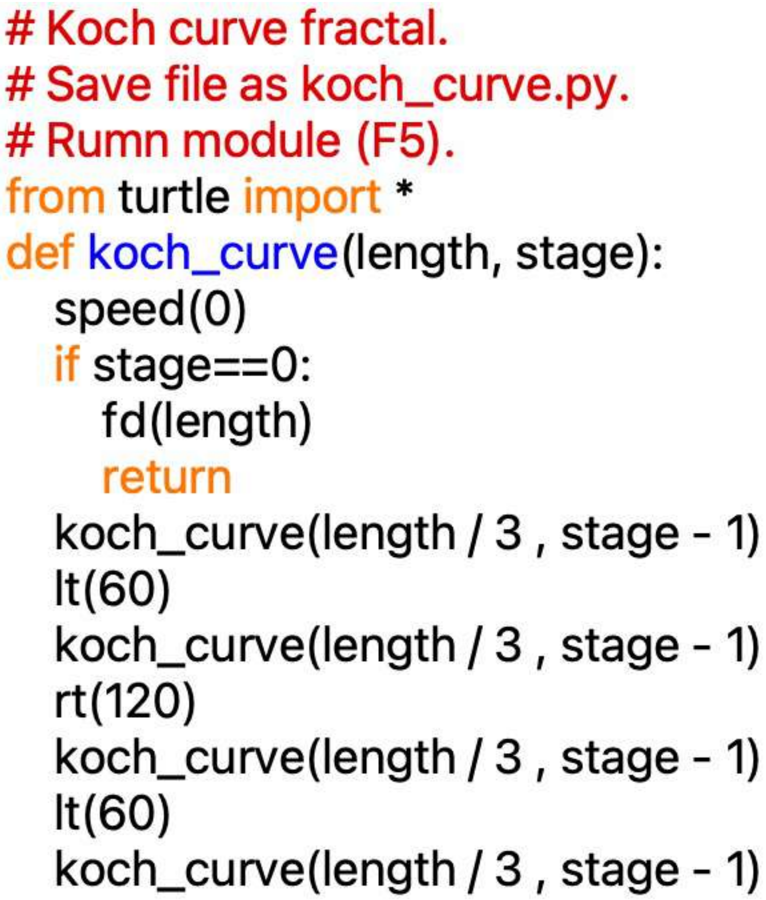
Cantor set: Remove the middle third segment at each stage.



```
>>> cantor(-200, 200, 300)
```

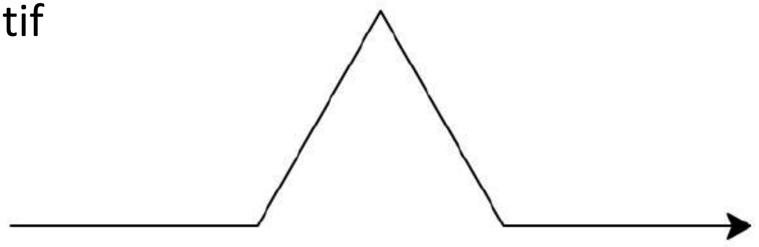
Problem: Edit the program to plot a variant of the Cantor set where the two middle fifth segments are removed at each stage.

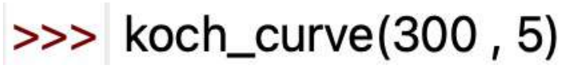
The Turtle Module (The Koch Curve)



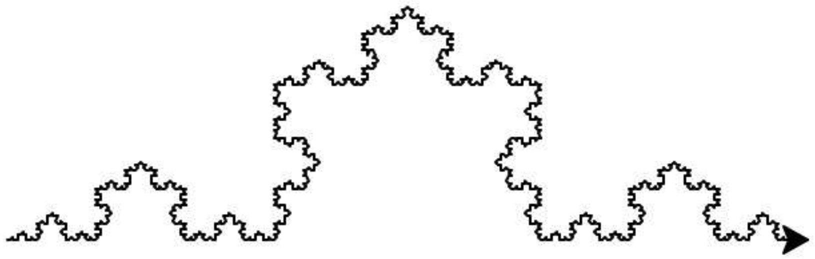
```
# Koch curve fractal.
# Save file as koch_curve.py.
# Rumn module (F5).
from turtle import *
def koch_curve(length, stage):
    speed(0)
    if stage==0:
        fd(length)
        return
    koch_curve(length / 3 , stage - 1)
    lt(60)
    koch_curve(length / 3 , stage - 1)
    rt(120)
    koch_curve(length / 3 , stage - 1)
    lt(60)
    koch_curve(length / 3 , stage - 1)
```

Motif



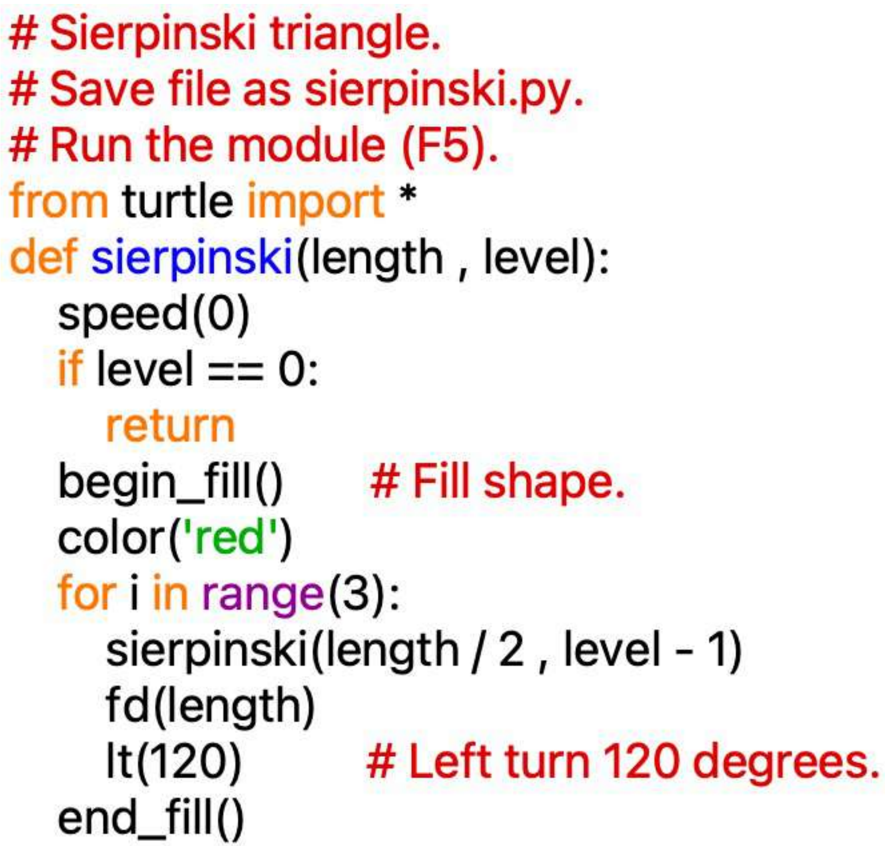


```
>>> koch_curve(300 , 5)
```

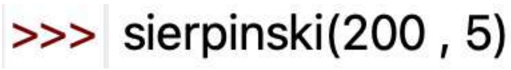


Problem: Edit the program to plot a Koch square fractal, where one segment (one third length) is replaced with 5 segments.

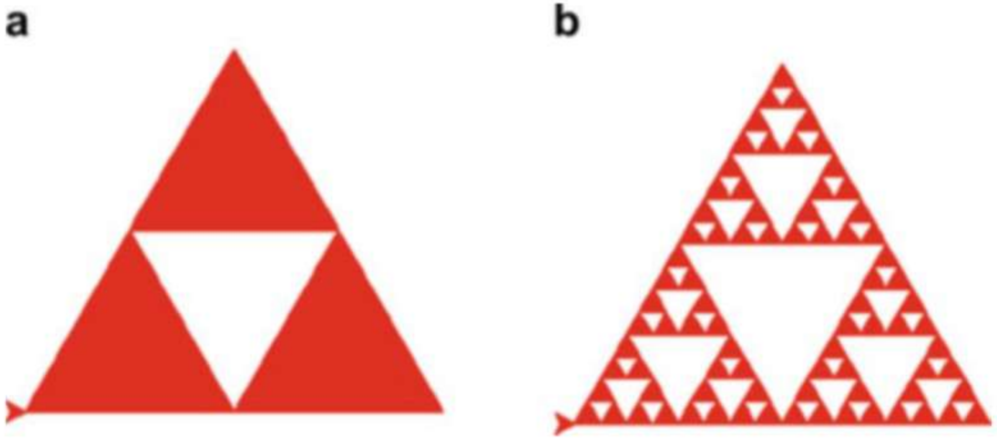
The Turtle Module (The Sierpinski Triangle Fractal)



```
# Sierpinski triangle.
# Save file as sierpinski.py.
# Run the module (F5).
from turtle import *
def sierpinski(length , level):
    speed(0)
    if level == 0:
        return
    begin_fill()      # Fill shape.
    color('red')
    for i in range(3):
        sierpinski(length / 2 , level - 1)
        fd(length)
        lt(120)       # Left turn 120 degrees.
    end_fill()
```

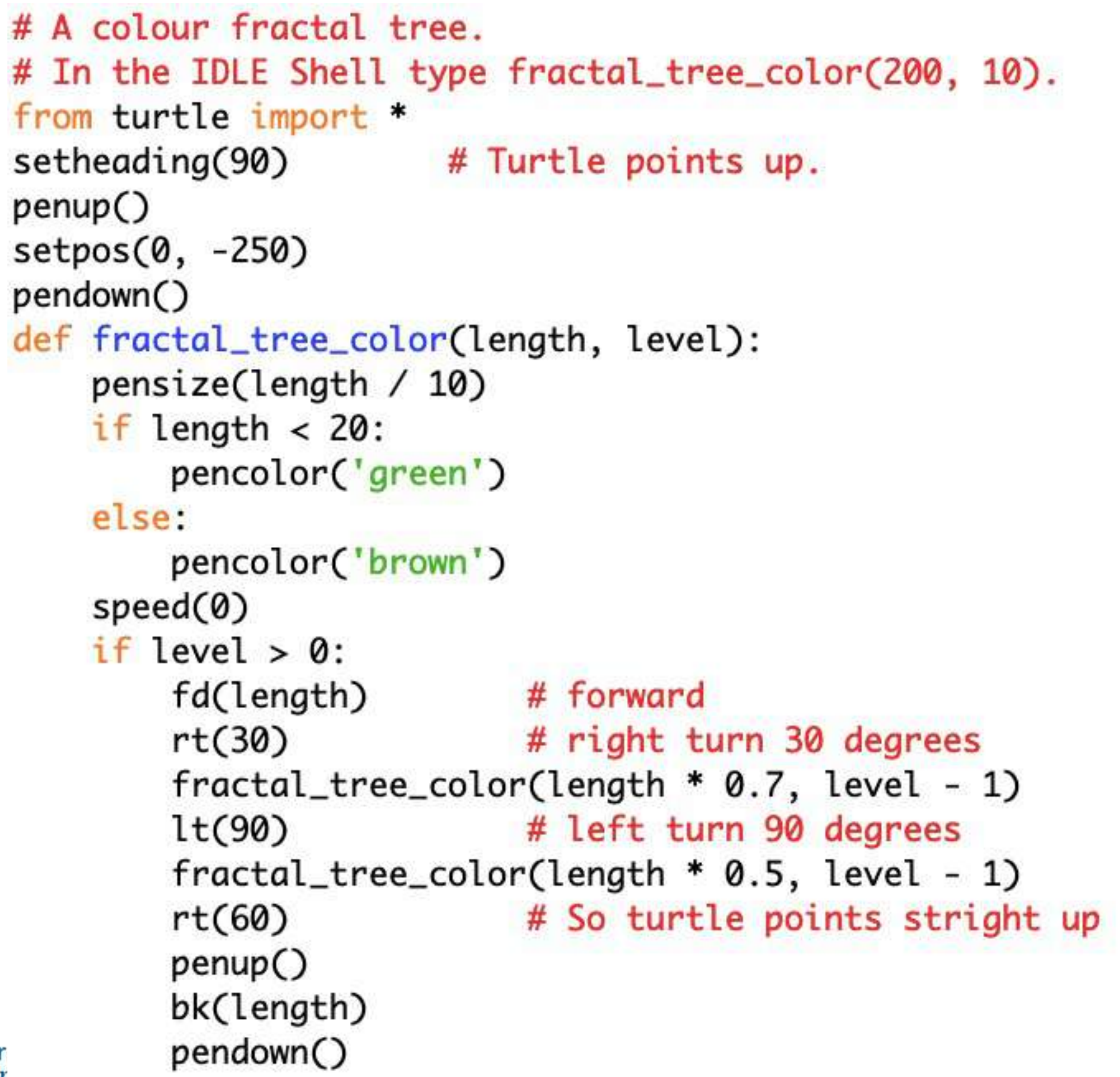


```
>>> sierpinski(200 , 5)
```

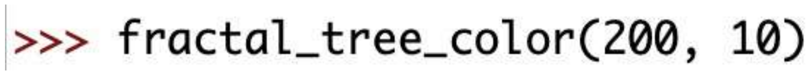


Problem: Edit the program to plot a Sierpinski square fractal, where the central square is removed at each stage.

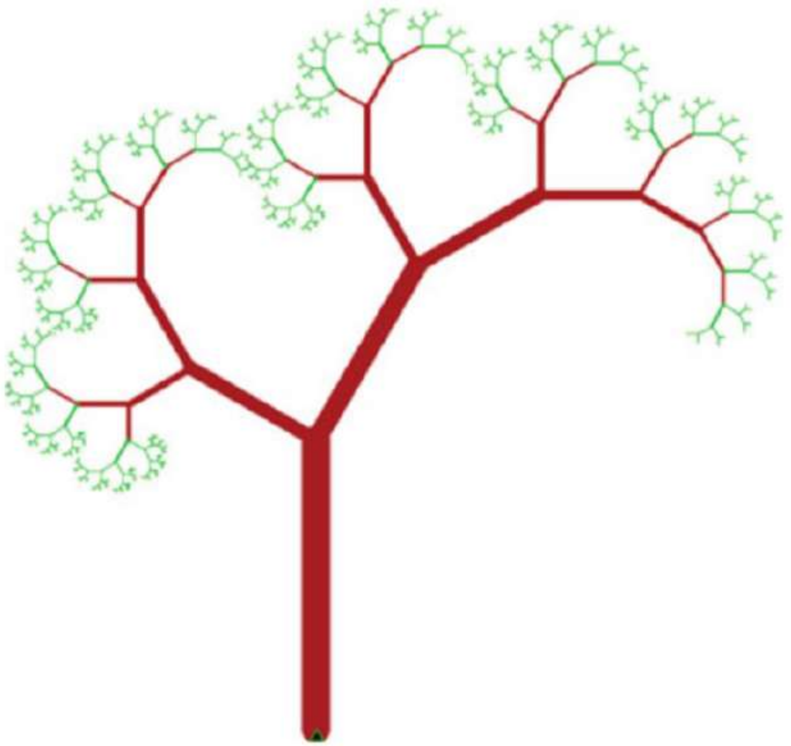
The Turtle Module (A Fractal Tree)



```
# A colour fractal tree.
# In the IDLE Shell type fractal_tree_color(200, 10).
from turtle import *
setheading(90)          # Turtle points up.
penup()
setpos(0, -250)
pendown()
def fractal_tree_color(length, level):
    pensize(length / 10)
    if length < 20:
        pencolor('green')
    else:
        pencolor('brown')
    speed(0)
    if level > 0:
        fd(length)          # forward
        rt(30)              # right turn 30 degrees
        fractal_tree_color(length * 0.7, level - 1)
        lt(90)              # left turn 90 degrees
        fractal_tree_color(length * 0.5, level - 1)
        rt(60)              # So turtle points stright up
        penup()
        bk(length)
        pendown()
```

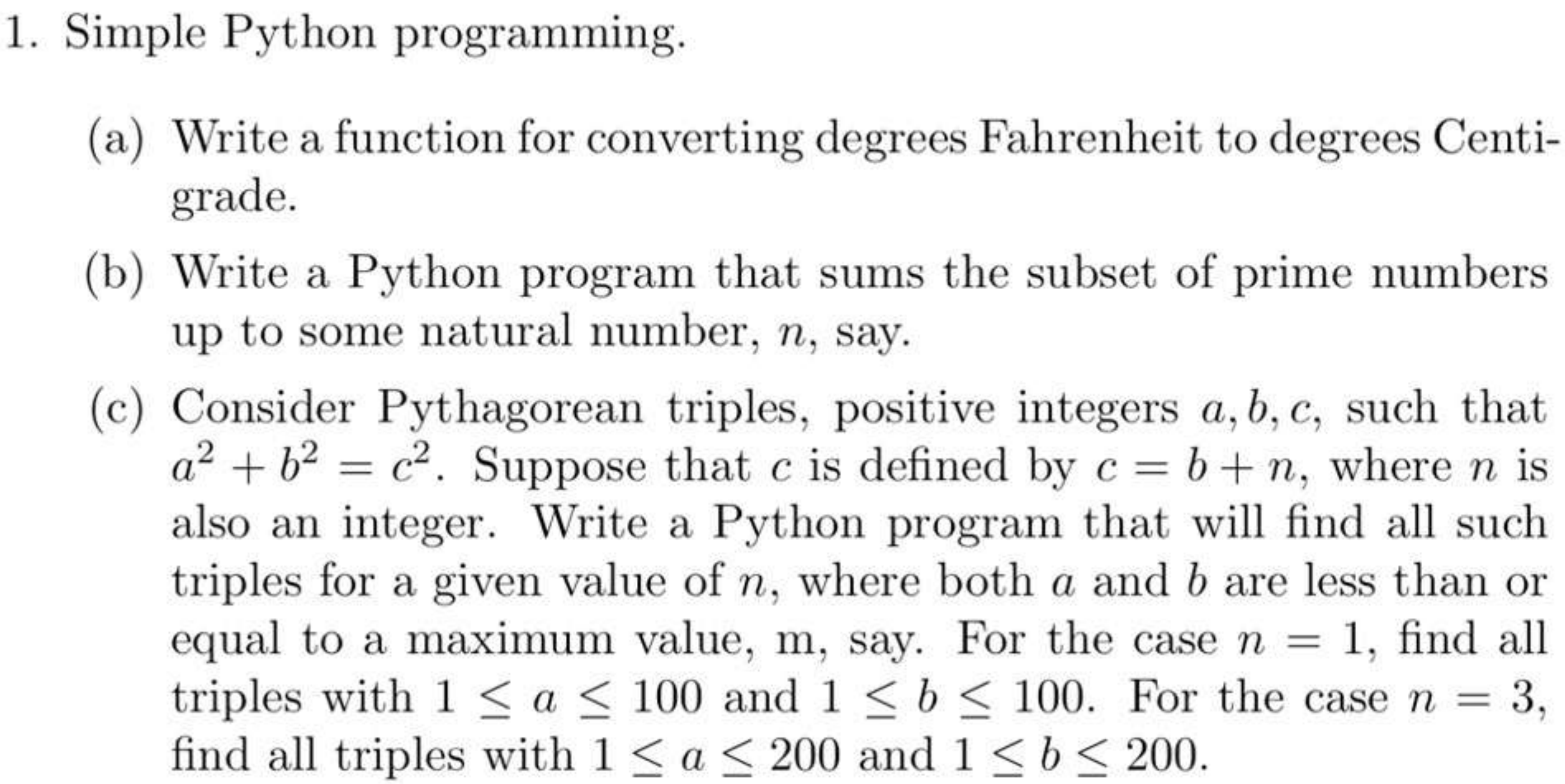


```
>>> fractal_tree_color(200, 10)
```



Problem: Can you plot a trifurcating tree?

Exercises



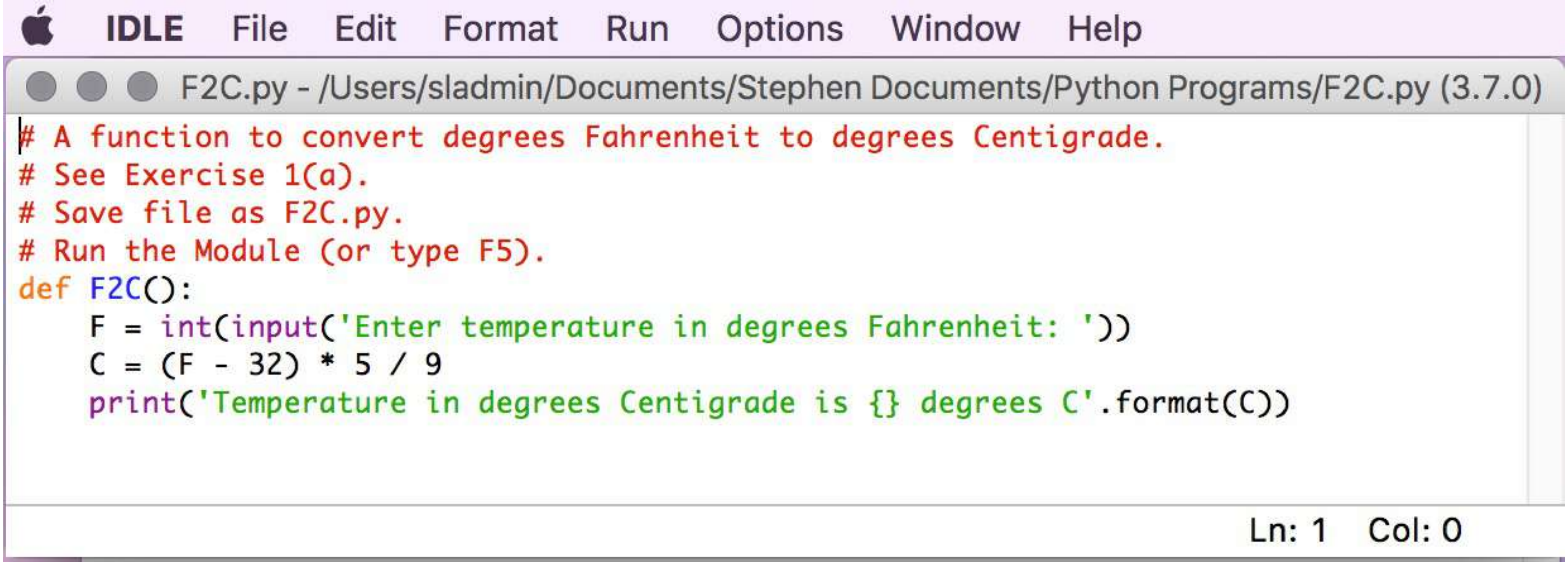
1. Simple Python programming.

   (a) Write a function for converting degrees Fahrenheit to degrees Centigrade.

   (b) Write a Python program that sums the subset of prime numbers up to some natural number, n, say.

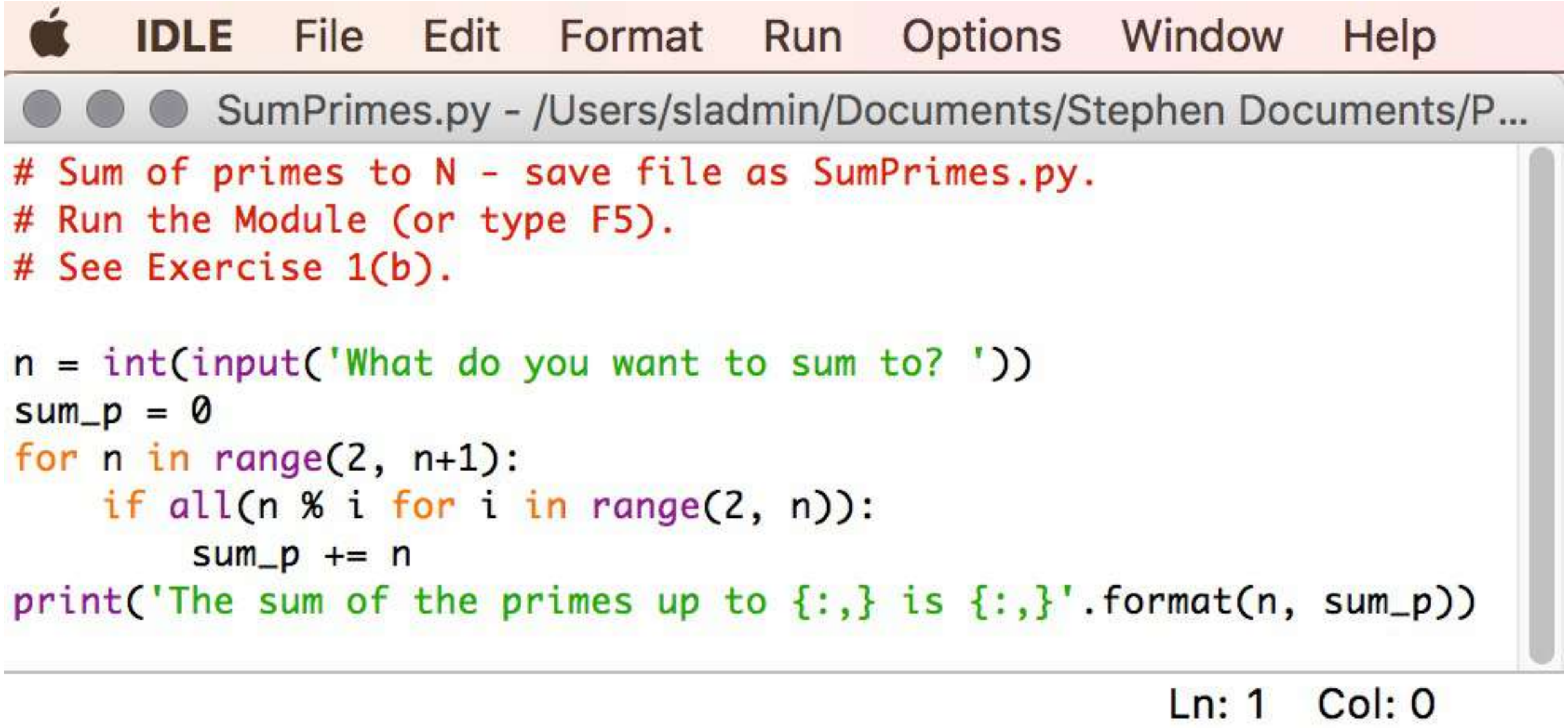
   (c) Consider Pythagorean triples, positive integers a, b, c, such that $a^2 + b^2 = c^2$. Suppose that c is defined by $c = b + n$, where n is also an integer. Write a Python program that will find all such triples for a given value of n, where both a and b are less than or equal to a maximum value, m, say. For the case $n = 1$, find all triples with $1 \leq a \leq 100$ and $1 \leq b \leq 100$. For the case $n = 3$, find all triples with $1 \leq a \leq 200$ and $1 \leq b \leq 200$.

Solutions 1(a)


IDLE File Edit Format Run Options Window Help
F2C.py - /Users/sladmin/Documents/Stephen Documents/Python Programs/F2C.py (3.7.0)
Ln: 1 Col: 0

```
# A function to convert degrees Fahrenheit to degrees Centigrade.
# See Exercise 1(a).
# Save file as F2C.py.
# Run the Module (or type F5).
def F2C():
    F = int(input('Enter temperature in degrees Fahrenheit: '))
    C = (F - 32) * 5 / 9
    print('Temperature in degrees Centigrade is {} degrees C'.format(C))
```

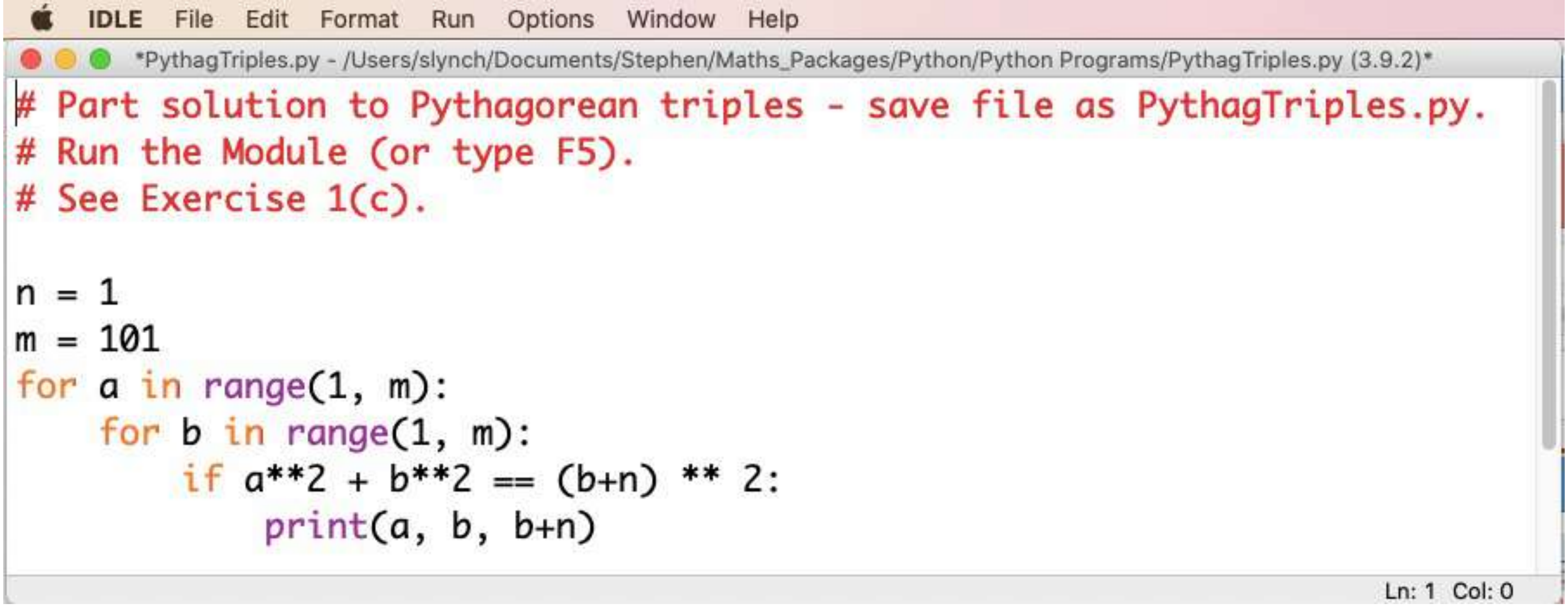
Solutions 1(b)



IDLE File Edit Format Run Options Window Help
SumPrimes.py - /Users/sladmin/Documents/Stephen Documents/P...
Ln: 1 Col: 0

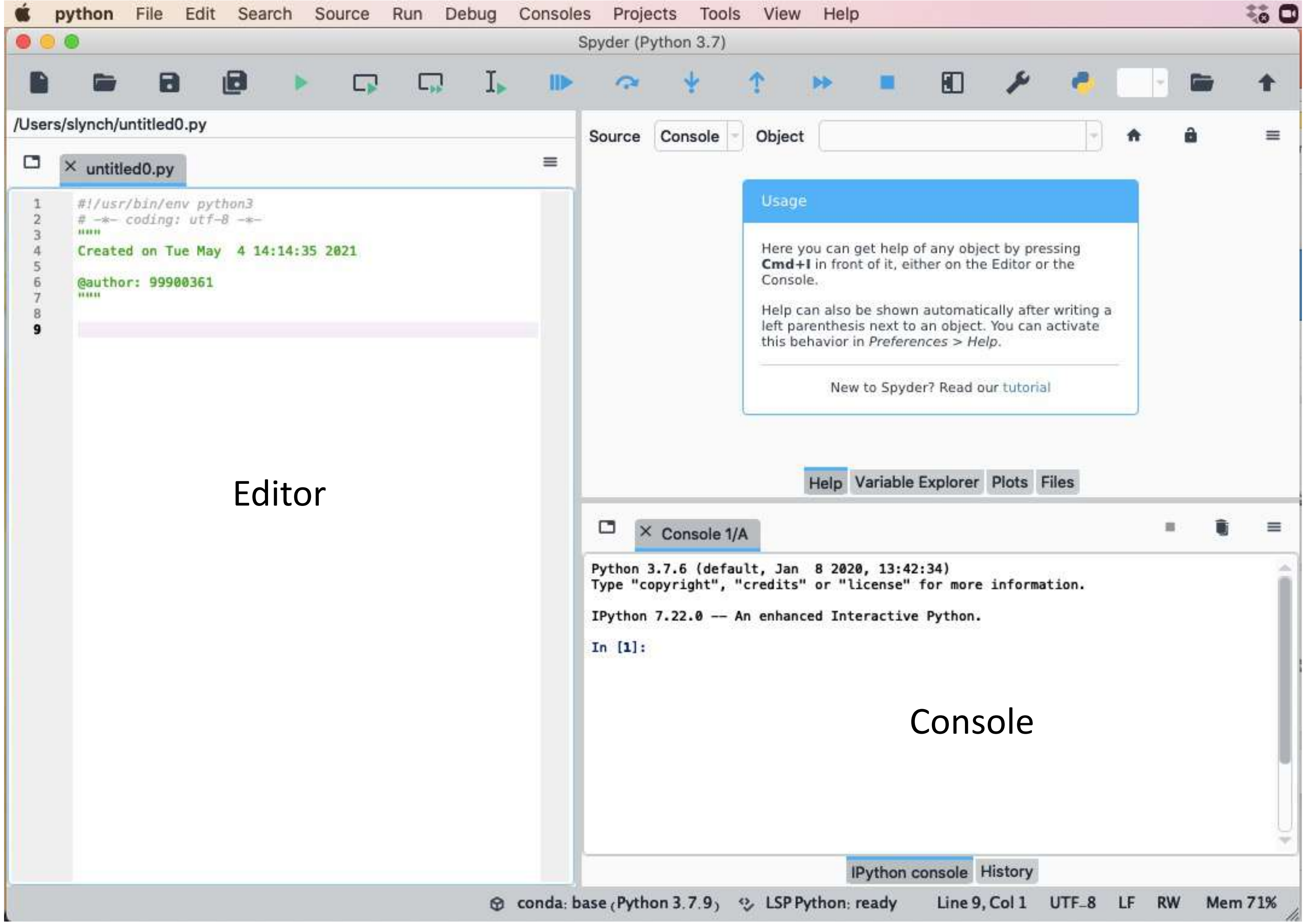
```python
# Sum of primes to N - save file as SumPrimes.py.
# Run the Module (or type F5).
# See Exercise 1(b).

n = int(input('What do you want to sum to? '))
sum_p = 0
for n in range(2, n+1):
    if all(n % i for i in range(2, n)):
        sum_p += n
print('The sum of the primes up to {:,} is {:,}'.format(n, sum_p))
```

Part solution 1(c): End Session 3


IDLE
File
Edit
Format
Run
Options
Window
Help
PythagTriples.py - /Users/slynch/Documents/Stephen/Maths_Packages/Python/Python Programs/PythagTriples.py (3.9.2)
Ln: 1 Col: 0

```
# Part solution to Pythagorean triples - save file as PythagTriples.py.
# Run the Module (or type F5).
# See Exercise 1(c).

n = 1
m = 101
for a in range(1, m):
    for b in range(1, m):
        if a**2 + b**2 == (b+n) ** 2:
            print(a, b, b+n)
```

Spyder: Launch from Anaconda: Start Session 4



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue May  4 14:14:35 2021

@author: 99900361
"""

```

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
Python 3.7.6 (default, Jan  8 2020, 13:42:34)
Type "copyright", "credits" or "license" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.

In [1]:
```

Editor

Console

Manchester Metropolitan University

Numpy (NUMeric PYthon) in the Spyder Console

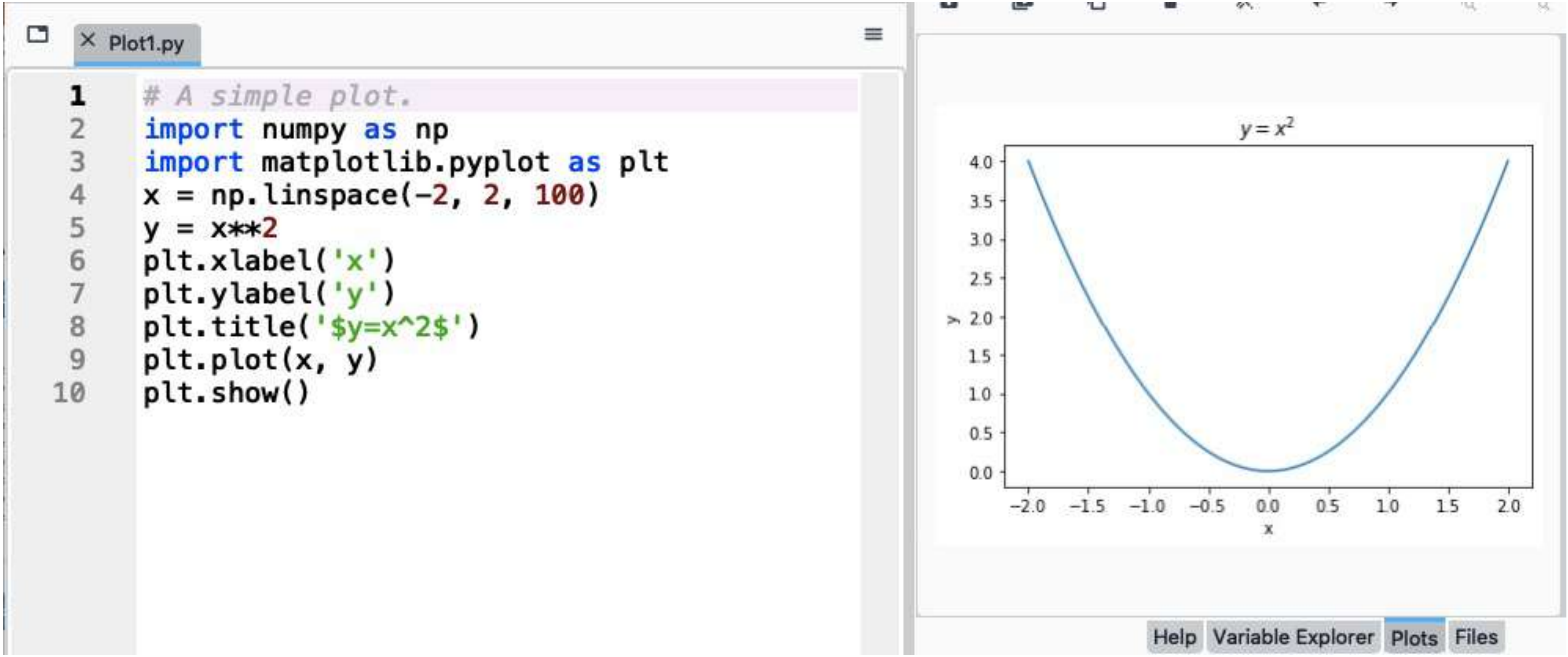
```
Python Commands                         Comments

In[1]: import numpy as np               # Import numpy into the np namespace.

In[2]: a = np.arange(5)                 # A 1d array [0 1 2 3 4].

In[3]: b = np.arange(6).reshape(2, 3)  # A 2d array [[0 1 2], [3 4 5]].

In[4]: v = np.array([1, 2, 3, 4])       # A 1d array and a vector.

In[5]: A = np.array([[1, 1], [0, 1]])   # A 2d array.

In[6]: B = np.array([[2, 0], [3, 4]])   # A 2d array.

In[7]: A ∗ B                            # Elementwise product.

In[8]: np.dot(A, B)                     # Matrix product [[5, 4], [3, 4]].
```

Numpy (NUMeric PYthon) in the Spyder Console

```
Python Commands                           Comments

In[9]: c = np.arange(12).reshape(3, 4)   # A 2d array.

In[10]: c.sum(axis = 0)                  # Sum each column.

In[11]: c.max(axis = 1)                  # The maximum of each row.

In[12]: c.min(axis = 1)                  # The minimum of each row.

In[13]: c.cumsum(axis = 0)               # Cumulative sum of each column.

In[14]: np.linspace(0, 6, 4)             # An array([0, 2, 4, 6]).

In[15]: x = np.linspace(-2, 2, 100)      # Set up a domain.

In[16]: y = x**2                         # A set of y values.
```

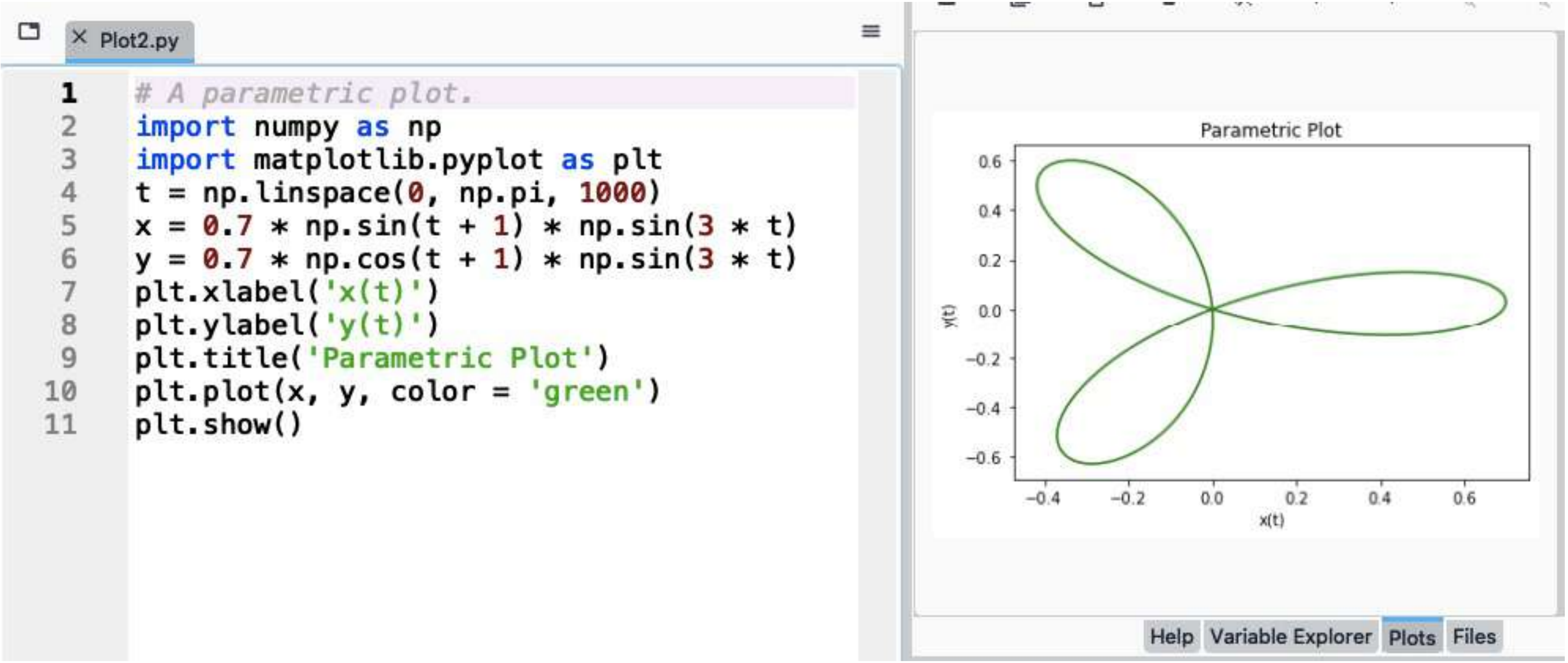
Manchester Metropolitan University

Matplotlib (MATrix PLOTting LIBrary) in Spyder



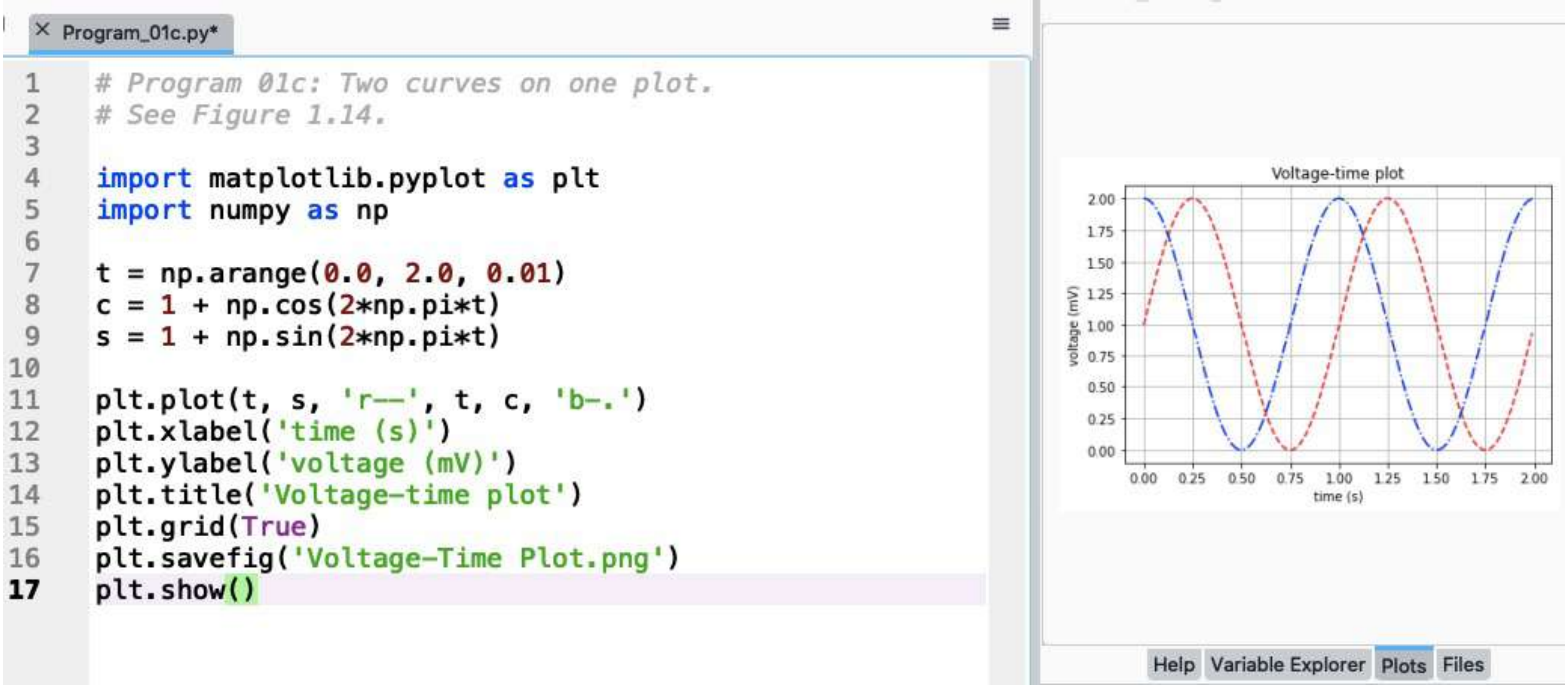
```python
# A simple plot.
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-2, 2, 100)
y = x**2
plt.xlabel('x')
plt.ylabel('y')
plt.title('$y=x^2$')
plt.plot(x, y)
plt.show()
```

Matplotlib (MATrix PLOTting LIBrary) in Spyder



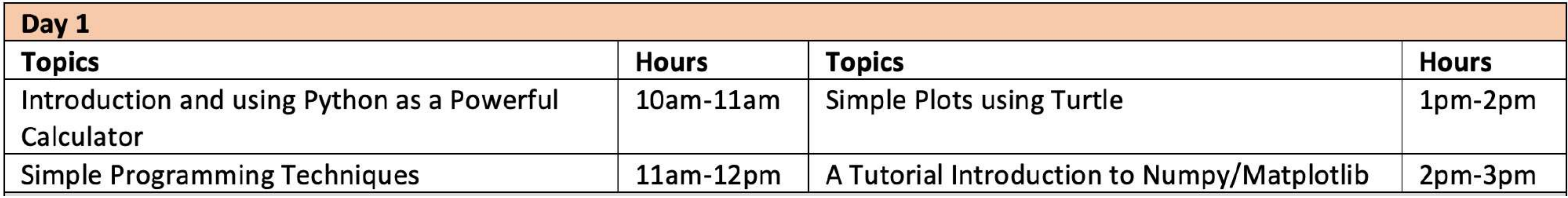
```python
# A parametric plot.
import numpy as np
import matplotlib.pyplot as plt
t = np.linspace(0, np.pi, 1000)
x = 0.7 * np.sin(t + 1) * np.sin(3 * t)
y = 0.7 * np.cos(t + 1) * np.sin(3 * t)
plt.xlabel('x(t)')
plt.ylabel('y(t)')
plt.title('Parametric Plot')
plt.plot(x, y, color = 'green')
plt.show()
```

Matplotlib (MATrix PLOTting LIBrary) in Spyder



```python
# Program 01c: Two curves on one plot.
# See Figure 1.14.

import matplotlib.pyplot as plt
import numpy as np

t = np.arange(0.0, 2.0, 0.01)
c = 1 + np.cos(2*np.pi*t)
s = 1 + np.sin(2*np.pi*t)

plt.plot(t, s, 'r--', t, c, 'b-.')
plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('Voltage-time plot')
plt.grid(True)
plt.savefig('Voltage-Time Plot.png')
plt.show()
```

End Day 1 Summary



Day 1			
Topics	**Hours**	**Topics**	**Hours**
Introduction and using Python as a Powerful Calculator	10am-11am	Simple Plots using Turtle	1pm-2pm
Simple Programming Techniques	11am-12pm	A Tutorial Introduction to Numpy/Matplotlib	2pm-3pm

You may also find the Jupyter notebook for A-level Mathematics useful:

http://www.doc.mmu.ac.uk/STAFF/S.Lynch/Python_for_A_Level_Mathematics_and_Beyond.html

Python for A-level Mathematics, undergraduate Mathematics and employability:

https://www.mathscareers.org.uk/python-for-a-level-maths-undergraduate-maths-and-employability/