Python for A-Level Mathematics and Beyond

One-Day Intensive Workshop

Chapter 1: Introduction to Python™

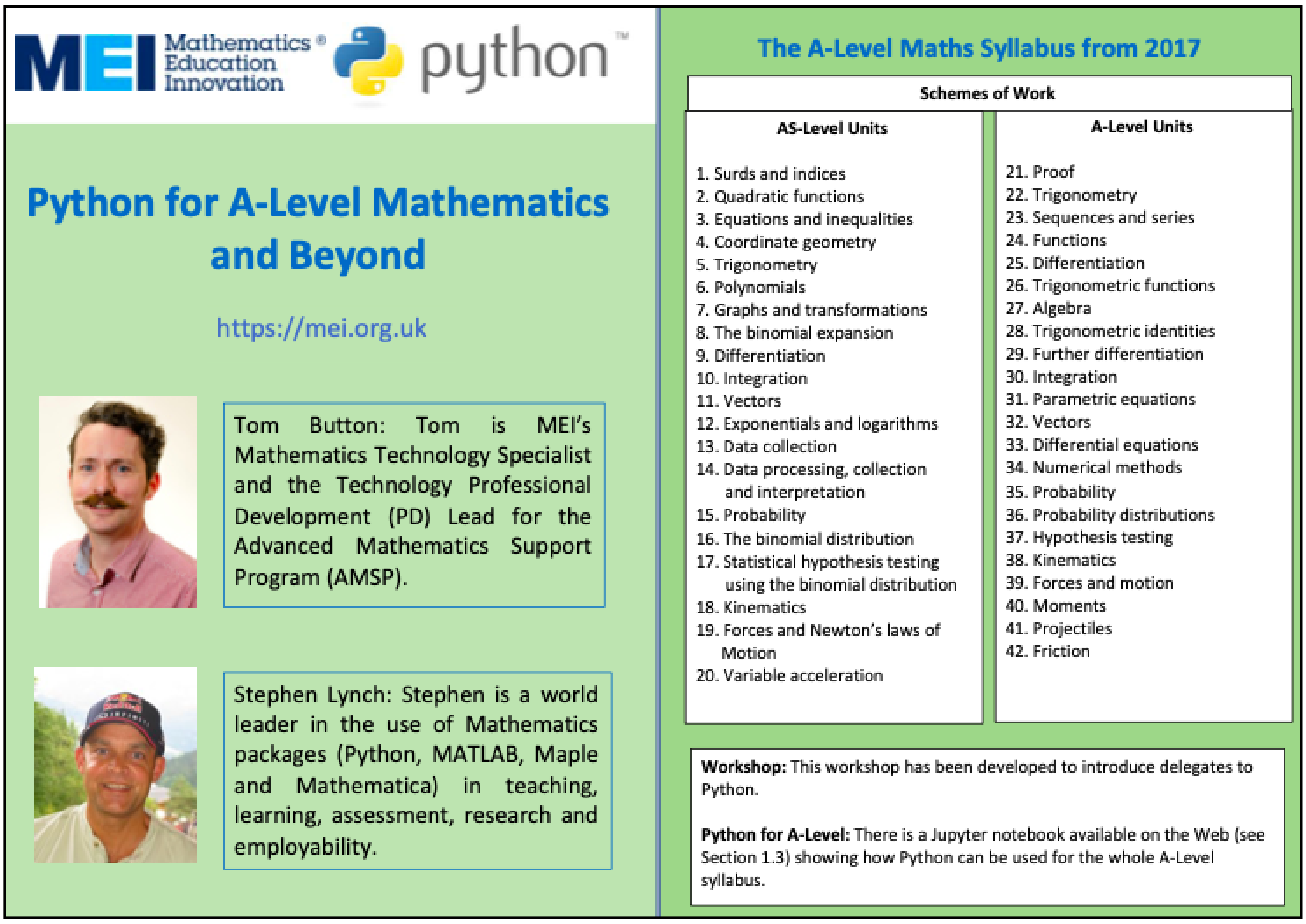
https://www.python.org

1.1 General Information

Python is a high-level technical computing language and supports functional, imperative, object-oriented, and procedural styles of programming. It was developed by Guido van Rossum and first released in 1991.

We will be using Python to help you with your A-Level Mathematics and Further Mathematics modules. You will also find Python extremely useful when studying other STEM subjects.

Below is the cover sheet for the worksheets for the MEI AS-Level and A-Level Maths schemes of work.



MEI Mathematics Education Innovation python™

Python for A-Level Mathematics and Beyond

https://mei.org.uk

Tom Button: Tom is MEI's Mathematics Technology Specialist and the Technology Professional Development (PD) Lead for the Advanced Mathematics Support Program (AMSP).

Stephen Lynch: Stephen is a world leader in the use of Mathematics packages (Python, MATLAB, Maple and Mathematica) in teaching, learning, assessment, research and employability.

The A-Level Maths Syllabus from 2017

Schemes of Work	
AS-Level Units	**A-Level Units**
1. Surds and indices	21. Proof
2. Quadratic functions	22. Trigonometry
3. Equations and inequalities	23. Sequences and series
4. Coordinate geometry	24. Functions
5. Trigonometry	25. Differentiation
6. Polynomials	26. Trigonometric functions
7. Graphs and transformations	27. Algebra
8. The binomial expansion	28. Trigonometric identities
9. Differentiation	29. Further differentiation
10. Integration	30. Integration
11. Vectors	31. Parametric equations
12. Exponentials and logarithms	32. Vectors
13. Data collection	33. Differential equations
14. Data processing, collection and interpretation	34. Numerical methods
15. Probability	35. Probability
16. The binomial distribution	36. Probability distributions
17. Statistical hypothesis testing using the binomial distribution	37. Hypothesis testing
18. Kinematics	38. Kinematics
19. Forces and Newton's laws of Motion	39. Forces and motion
20. Variable acceleration	40. Moments
	41. Projectiles
	42. Friction

Workshop: This workshop has been developed to introduce delegates to Python.

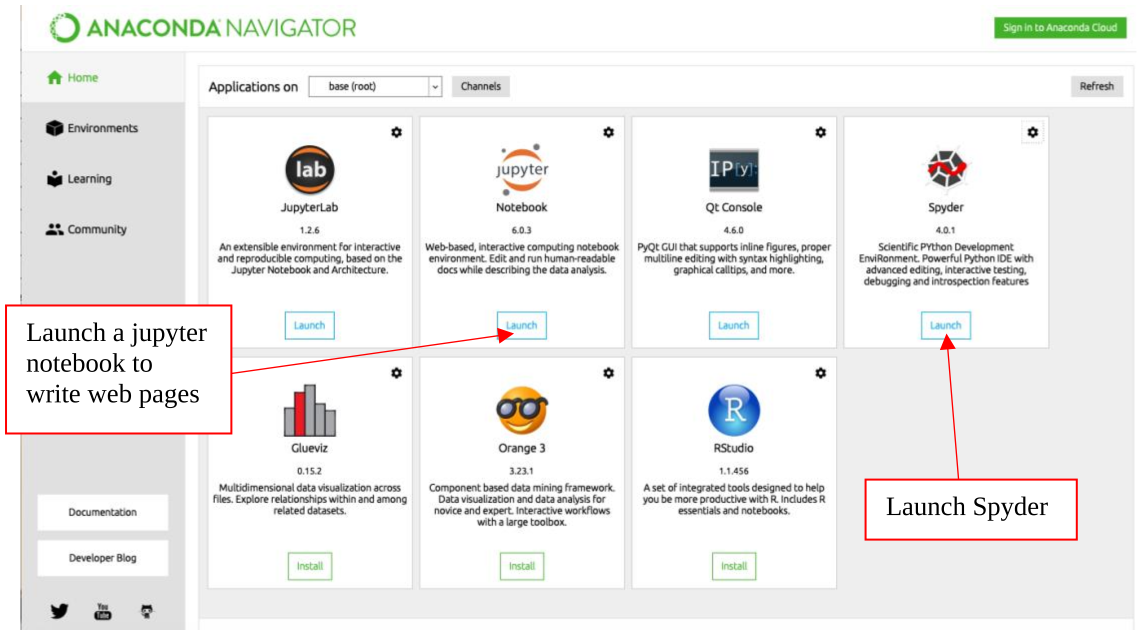
Python for A-Level: There is a Jupyter notebook available on the Web (see Section 1.3) showing how Python can be used for the whole A-Level syllabus.

1.2. Accessing Python

The Anaconda free package manager, environment manager and Python distribution will be used throughout the course. Download **Anaconda** using the link:

https://www.anaconda.com/

It is assumed that pupils/staff are familiar with either the Windows, Mac or Unix platforms. It is also assumed that pupils/staff know how to login and how to open the Anaconda Navigator window:



JupyterLab 1.2.6: An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook 6.0.3: Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console 4.6.0: PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder 4.0.1: Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

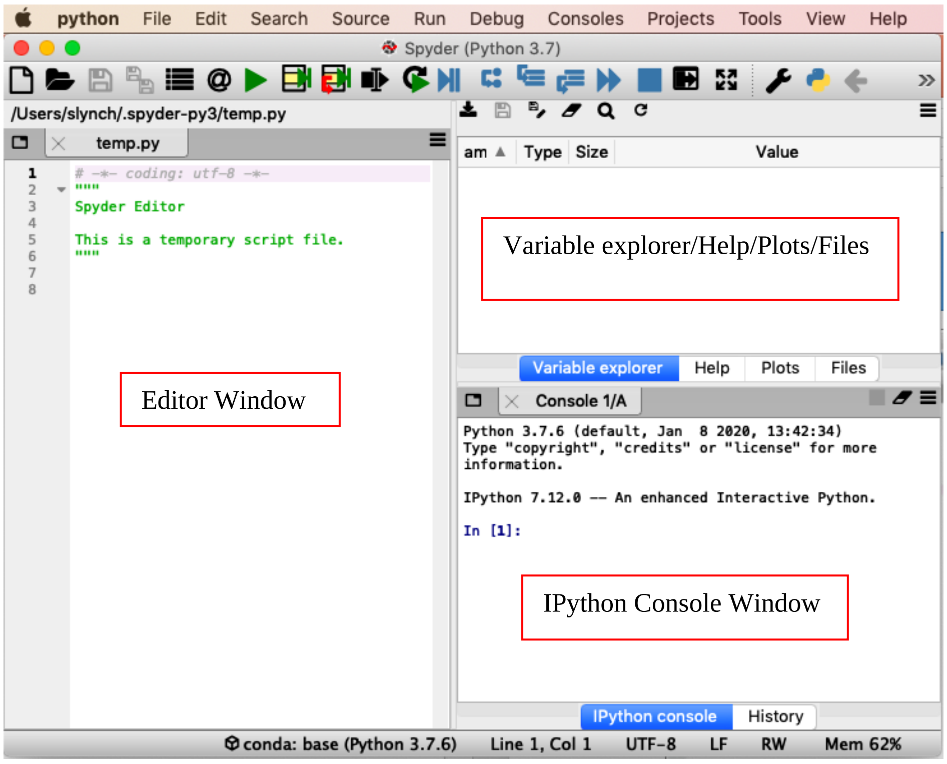
Glueviz 0.15.2: Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3 3.23.1: Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

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

1.3. The Spyder Integrated Development Environment (IDE)

Spyder is an acronym for "**S**cientific **PY**thon **D**evelopment **E**nvi**R**onment". When you click on the Spyder launch button, the following IDE opens:


python File Edit Search Source Run Debug Consoles Projects Tools View Help
Spyder (Python 3.7)
/Users/slynch/.spyder-py3/temp.py
temp.py
am Type Size Value
Variable explorer/Help/Plots/Files
Variable explorer Help Plots Files
Editor Window
Console 1/A
IPython Console Window
IPython console History
conda: base (Python 3.7.6) Line 1, Col 1 UTF-8 LF RW Mem 62%

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
```

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

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]:
```

IPython Console Window: This is where the user can use Python as a graphing calculator. The command prompt In [1]: indicates that Python is waiting for you to type commands.

Variable explorer/Help/Plots/Files Window: This window can display detailed help on variables or files and is useful when debugging. Your plots will be displayed here.

Editor Window: Used to write code (programs) and save to file.

Some Advantages of Learning Python

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

- Programming makes you extremely employable
- Programming can be fun
- You can check your hand written Mathematics work with Python
- You can make up your own examples
- You can solve real-world problems
- You can solve problems which are impossible without a computer
- You can create colourful graphics
- You can label graphs
- You can use the Greek alphabet and other mathematical symbols in plots
- Python will give you a deeper understanding of Mathematics

Python can be used to cover the whole A-Level syllabus. See my Jupyter notebook here:

https://drstephenlynch.github.io/webpages/Python_for_A_Level_Mathematics_and_Beyond.html

1.4. Using the IPython Console as a Powerful Calculator

The Console Window is where the Python commands are typed and executed and this is where the user can use Python just like a graphing calculator. When in the Console Window, command lines may be returned to and edited using the up and down arrow keys. To execute a command type RETURN or ENTER on the keyboard. If you do make a typing error, Spyder will give an Error: message. Do not re-type the line, simply use the up-arrow key and edit your mistake. To quit the Python kernel and start afresh simply type quit after the prompt.

In Subsection 1.4.1, copy the commands after the prompts in the Console Window and read the Comments to see what the commands do.

If you do not understand any of the commands or output, call over an instructor or look up the commands on Google by typing in: Python AND *command name* in a Web search engine.

1.4.1 Python Tutorial One: Calculator Commands (30 minutes)

Python Commands	**Comments**
In[1]: # This is a comment	# Writing comments in Python.
In[2]: 4 + 5 - 3	# Addition and subtraction.
In[3]: 2 * 3 / 6	# Multiplication and division.
In[4]: 2**8	# Exponentiation.
In[5]: import math	# Import the math module.
In[6]: help(math)	# List the functions.
In[7]: math.sqrt(9)	# You need the math prefix. Square root function.
In[8]: from math import *	# Import all math functions.
In[9]: factorial(52)	# Gives 52! How does this relate to a pack of cards?
In[10]: sin(0.5)	# The sine function (radians).
In[11]: asin(0.4794)	# Inverse sine function.
In[12]: log(2)	# Natural logarithm function.
In[13]: log10(10)	# Logarithm function base 10.
In[14]: cosh(0.3)	# Hyperbolic cosine function.
In[15]: exp(2)	# Exponential function.
In[16]: 1 / 3 + 1 / 4	# Fractions - floating point arithmetic.
In[17]: from fractions import Fraction	# Import Fraction function.
In[18]: Fraction(1 , 3) + Fraction(1 , 4)	# Symbolic fractions.
In[19]: floor(2.456)	# The largest integer <= 2.456.
In[20]: ceil(2.456)	# The smallest integer >= 2.456.
In[21]: trunc(5.645)	# Truncates nearest integral to 0.
In[22]: degrees(0.5)	# Convert radians to degrees.
In[23]: fmod(13 , 6)	# Gives 13 modulo 6.
In[24]: gcd(123 , 321)	# The greatest common divisor.
In[25]: 17 % 3	# The % operator returns the remainder.
In[26]: pi	# The number $\pi \approx 3.14159265$.
In[27]: round(_, 3)	# Round the last printed expression to 3 decimal places.
In[28]: e	# The number $e \approx 2.718281828$.
In[29]: tau	# The number $\tau = 2\pi \approx 6.2831853$.
In[30]: quit	# Quits the IPython console and restarts the kernel.

1.4.2 Python Tutorial Two: Symbolic Python (SymPy) (30 minutes)

Sympy is a computer algebra system and a Python library for symbolic mathematics written entirely in Python. For more detail, see the sympy help pages at:

http://docs.sympy.org/latest/index.html

Python Commands	**Comments**
In[1]: from sympy import *	# Import everything from the sympy library.
In[2]: x, y = symbols(' x y ')	# Declare x and y symbolic.
In[3]: factor(x**2 – y**2)	# Factorize $x^2 - y^2 = (x - y)(x + y)$.
In[4]: solve(x**2 – 4*x – 3, x)	# Solve an algebraic equation, $x^2 - 4x - 3 = 0$.
In[5]: apart(1 / ((x + 2)*(x + 1)))	# Partial fractions.
In[6]: trigsimp(cos(x) – cos(x)**3)	# Simplify a trig expression.
In[7]: limit(x / sin(x), x, 0)	# Limits, $\lim_{x \to 0} \frac{x}{\sin(x)}$.
In[8]: diff(x**2 - 7*x + 8, x)	# Differentiate with respect to x.
In[9]: diff(5*x**7, x, 3)	# Differentiate with respect to x three times.
In[10]: (exp(x)*cos(x)).series(x, 0, 10)	# Taylor series expansion around $x = 0$.
In[11]: integrate(x**2 - 7*x + 8, x)	# Indefinite integration, $\int x^2 - 7x + 8 \, dx$
In[12]: integrate(x**2 - 7*x + 8, (x, 1, 2))	# Definite integration, $\int_{x=1}^{2} x^2 - 7x + 8 \, dx$.
In[13]: summation(1 / x**2, (x, 1, oo))	# Infinite summation, $\sum_{x=1}^{\infty} \frac{1}{x^2}$.
In[14]: solve([x+5*y-2, -3*x+6*y-15],[x,y])	# Solve simultaneous equations.
In[15]: A = Matrix([[1, -1], [2, 3]])	# The matrix, $A = \begin{pmatrix} 1 & -1 \\ 2 & 3 \end{pmatrix}$.
In[16]: B = Matrix([[0, 2], [3, 3]])	# The matrix, $B = \begin{pmatrix} 0 & 2 \\ 3 & 3 \end{pmatrix}$.
In[17]: 2 * A + 3 * B	# Matrix algebra.
In[18]: A * B	# Matrix multiplication. Row by column.
In[19]: A.row(0)	# Access row 1 (Zero-based indexing).
In[20]: A.row(1)	# Access row 2 (Zero-based indexing).
In[21]: A.T	# The transpose matrix, swap rows ↔ columns.
In[22]: A[0, 1]	# The element in row one, column 2.
In[23]: A**(-1)	# The inverse matrix, A^{-1} .
In[24]: A.det()	# The determinant.
In[25]: zeros(3, 3)	# Gives a 3×3 matrix of zeros.
In[26]: ones(1, 5)	# A 1×5 matrix of ones.
In[27]: N(pi, 500)	# Numerical evaluation, π to 500 significant figures.
In[28]: quit	# Quits the IPython console and restarts the kernel.

1.4.3 Python Tutorial Three: Numerical Python (NumPy), Lists and Arrays (30 minutes)

Numpy is a library that allows Python to compute with lists, arrays, vectors and tensors. Arrays, vectors and tensors are used a lot in Data Science. For more detail, see the numpy help pages at:

https://docs.scipy.org/doc/numpy/reference/

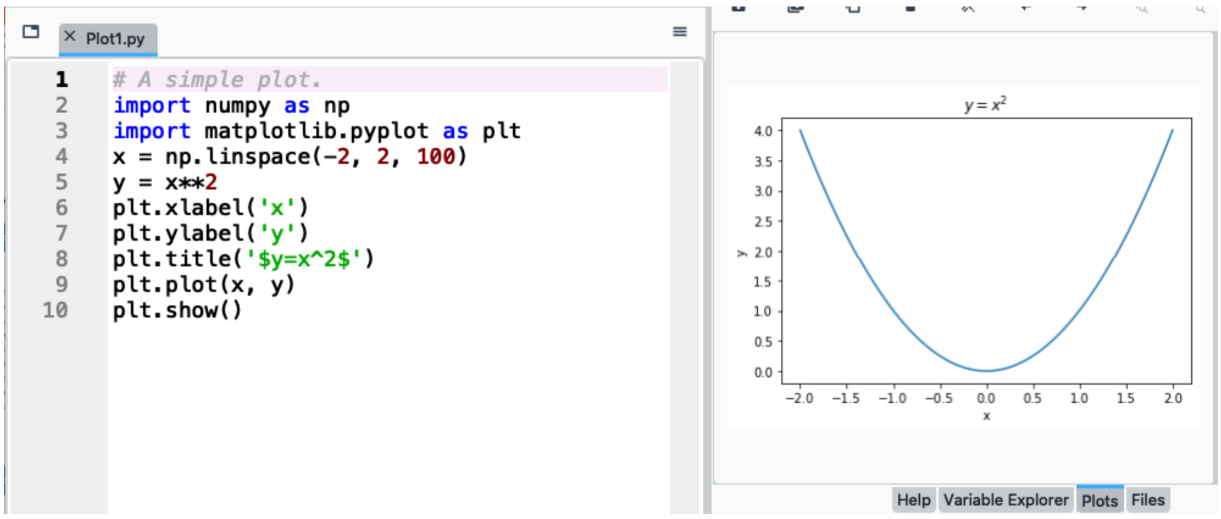
Python Commands	**Comments**
In[1]: import numpy as np	# Import numpy into the np namespace.
In[2]: a = [-1, 9, 10, 4, 3, 7, 8, 0, 9]	# A 1-D list.
In[3]: a[0]	# The first element a[0]= -1. Zero-based indexing.
In[4]: a[1]	# The second element a[1]=9.
In[5]: a[-1]	# The last element a[-1]=9.
In[6]: a[1 : 5 : 2]	# Slice to get the list [9, 4].
In[7]: a[2 : 8 : 3]	# Slice to get the list [10, 7].
In[8]: 2 * a	# A list, [-1,9,10,4,3,7,8,0,9,-1,9,10,4,3,7,8,0,9].
In[9]: a.append(10)	# Add 10 to the end of list, a=[-1,9,10,4,3,7,8,0,9,10].
In[10]: a.remove(9)	# Removes the first 9 in list, a=[-1,10,4,3,7,8,0,9,10].
In[11]: list(range(5))	# A list, [0,1,2,3,4].
In[12]: list(range(2, 10, 2))	# A list, [2,4,6,8].
In[13]: list(range(10, 5, -2))	# A list, [10,8,6].
In[14]: aa = [[1,2,3], [4,5,6]]	# A list of lists.
In[15]: aa[0]	# The first list, [1,2,3].
In[16]: aa[1][1]	# The second element in the second list.
In[17]: len(aa)	# The size of a list.
In[18]: A = np.arange(5)	# An array, A=array([0,1,2,3,4]).
In[19]: A.tolist()	# Convert an array to a list.
In[19]: B = np.arange(6).reshape(2, 3)	# A 2-D array, B=array([[0,1,2],[3,4,5]]).
In[20]: u = np.array([1, 2, 3])	# A 3-D vector and rank 1 tensor.
In[21]: C = np.array([[1, 1],[0, 1]])	# C is a 2-D array. C is also a rank 2 tensor.
In[22]: D = np.array([[2, 0],[3, 4]])	# D is a 2-D array.
In[23]: C * D	# Elementwise product [[2 0], [0 4]].
In[24]: C.dot(D)	# Matrix product [[5 4], [3 4]].
In[25]: np.dot(C, D)	# Matrix product.
In[26]: E = np.array([[1,0,0],[0,1,2],[3,4,5]])	# A 3 $\times$ 3 array.
In[27]: E.sum(axis = 0)	# Sum each column.
In[28]: E.min(axis = 1)	# The minimum of each row.
In[29]: E.cumsum(axis = 0)	# Cumulative sum for each column.
In[30]: quit	# Quits the IPython console.

1.4.4 Python Tutorial Four: MatPlotLib using the Editor Window (30 minutes)

Matplotlib is a Python plotting library. For more detail, see the matplotlib help pages at:

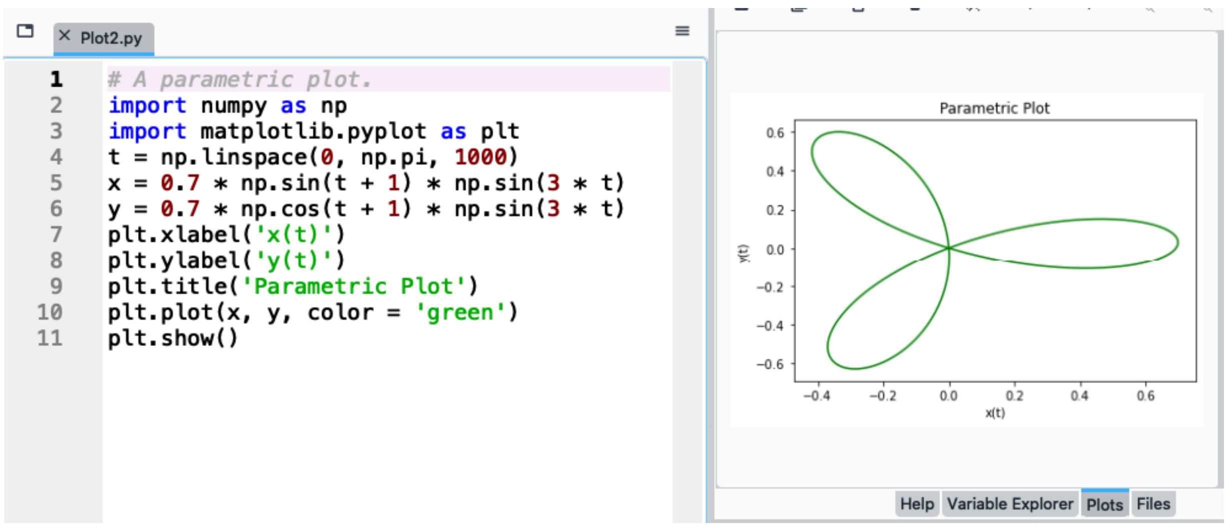
https://matplotlib.org/

Example 1.4.4.1 Use the Editor Window to plot a parabola.



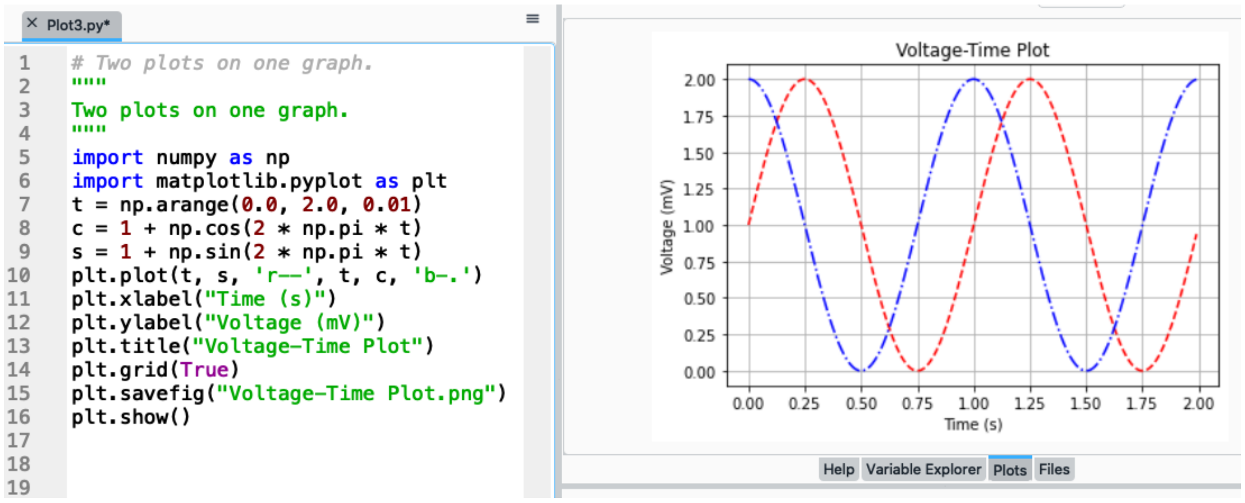
```python
# A simple plot.
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-2, 2, 100)
y = x**2
plt.xlabel('x')
plt.ylabel('y')
plt.title('$y=x^2$')
plt.plot(x, y)
plt.show()
```

Example 1.4.4.2 Use the Editor Window for a parametric plot.



```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()
```

Example 1.4.4.3 Use the Editor Window to get two plots on one graph.



```python
# Two plots on one graph.
"""
Two plots on one graph.
"""
import numpy as np
import matplotlib.pyplot as plt
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()
```

1.5. Python Exercises

1.5.1 Calculator Commands

1.5.1.1 Use Python to evaluate:

(a) $2-(3+6)$; (b) $2(5-8(3+6))$; (c) $2(2-2(3-6+5(4-7)))$;

(d) $\frac{2(5-4(3+8))}{3(4-(3-5))}$; (e) $\frac{2\times 4^5}{81-5!}$.

1.5.1.2 Use Python to evaluate:

(a) $\sqrt{4+6^5+5!}$; (b) $\cos(0.8)$; (c) atan (-0.4);

(d) $ln(87.95)$; (e) $log_{10}(725.345)$.

1.5.1.3 Use Python to evaluate symbolically:

(a) $\frac{1}{3}\times\frac{1}{4}-\frac{1}{5}$; (b) $\left(\frac{1}{3}-\frac{2}{5}\right)^3$; (c) $\frac{1}{7}\times\frac{4}{23}\div\frac{37}{125}$;

(d) $\left(\sqrt{2}-3\right)^4$; (e) $\frac{1}{\sqrt{2}+3}$.

1.5.1.4 Use Python to evaluate:

(a) $cos^2(0.9)$; (b) $\ln(8)-log_{10}(56)$; (c) 12!;

(d) $5186 \; mod \; 8$; (e) floor$(5\pi-2e)$.

1.5.2 Symbolic Computation

1.5.2.1 Use Python to:

(a) factorize x^3-y^3; (b) solve $x^2-7x-30=0$;

(c) split $\frac{3x}{(x-1)(x+2)(x-5)}$; (d) simplify $sin^4(x)-2cos^2(x)sin^2(x)+cos^4(x)$;

(e) expand $(y+x-3)(x^2-y+4)$.

1.5.2.2 Determine:

(a) $\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$; (b) $\frac{d}{dx}\left(3x^4-6x^3\right)$; (c) $\frac{d^3}{dx^3}\left(3x^4-6x^3\right)$;

(d) $\int_{x=0}^{1} x^2-2x-3\,dx$; (e) first 10 terms of the Taylor series of $e^x\sin(x)$ about $x=1$.

1.5.2.3 Use Python to:

(a) sum $\sum_{n=1}^{\infty}\frac{1}{n}$; (b) sum $\sum_{n=1}^{20} 2+3(n-1)$; (c) sum $\sum_{n=1}^{20} 2\times 3^n$;

(d) solve $\begin{matrix} 2x - y = 9 \\ 3x + y = 8 \end{matrix}$;   (e) solve $\begin{matrix} x^2 - y = 2 \\ x + y = 1 \end{matrix}$.

1.5.2.4 Given that

$$A = \begin{pmatrix} -1 & 2 & 4 \\ 0 & 3 & 2 \\ 1 & 4 & 6 \end{pmatrix} \text{ and } B = \begin{pmatrix} 2 & 3 & 3 \\ 0 & 3 & 0 \\ -6 & 1 & 1 \end{pmatrix}, \text{ use Python to evaluate:}$$

(a) $3A - 5B$;   (b) A × B ;   (c) the inverse of A ;

(d) the first column of A × B;   (e) the transpose of A × B .

1.5.3 Numerical Python and MatPlotLib

1.5.3.1 Use Python to generate the following lists:

(a) a list of integers from 5 to 200;   (b) a list of even integers from -4 to 200;

(c) a list of the form [3,6,9,…,300];   (d) a list of the form [30,27,24,…,-3];

(e) a list of the form [[1,2,…,100],[101,102,…,200],[201,202,…,300]].

1.5.3.2 Given the array of numbers

$$\begin{matrix} 2 & 6 & 0 & 6 & -5 & 2 & 6 & 0 & 8 & -9 \\ -1 & 8 & 7 & 4 & 1 & -1 & 9 & 9 & 4 & 1 \\ 0 & 24 & 8 & 2 & 12 & 0 & 14 & 8 & 1 & 2 \end{matrix}$$

determine:

(a) the sum of each row;   (b) the sum of each column;   (c) the minimum of each row;

(d) the maximum of each column;   (e) the cumulative sum for each row.

1.5.3.3 Plot the following functions:

(a) $y = x^2 - 3x - 18$;   (b) $y = \cos(2x)$;   (c) $y = sin^2(x)$;

(d) $y = 4x^3 - 3x^4$;   (e) $y = \cosh(x)$.

1.5.3.4 Read the MatPlotLib3d tutorial:

https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

Plot the surface defined by $z(x, y) = \sin(x) + \cos(y)$, and rotate with Spyder.

Solutions:

https://drstephenlynch.github.io/webpages/Python_for_A_Level_Maths_and_Beyond_Solutions.html

Python for A-Level Mathematics and Beyond

Chapter 2: Python Programming

https://www.python.org/

2.1. General Information

Programming languages such as Java, Javascript, C++, C#, PHP, MATLAB, Swift, SQL, Ruby and Python are vitally important in the technological age. Scientific computing, artificial intelligence, bitcoins, the blockchain, cloud computing, IOT (Internet of Things), machine learning and deep learning, for example, are only possible because of programming languages.

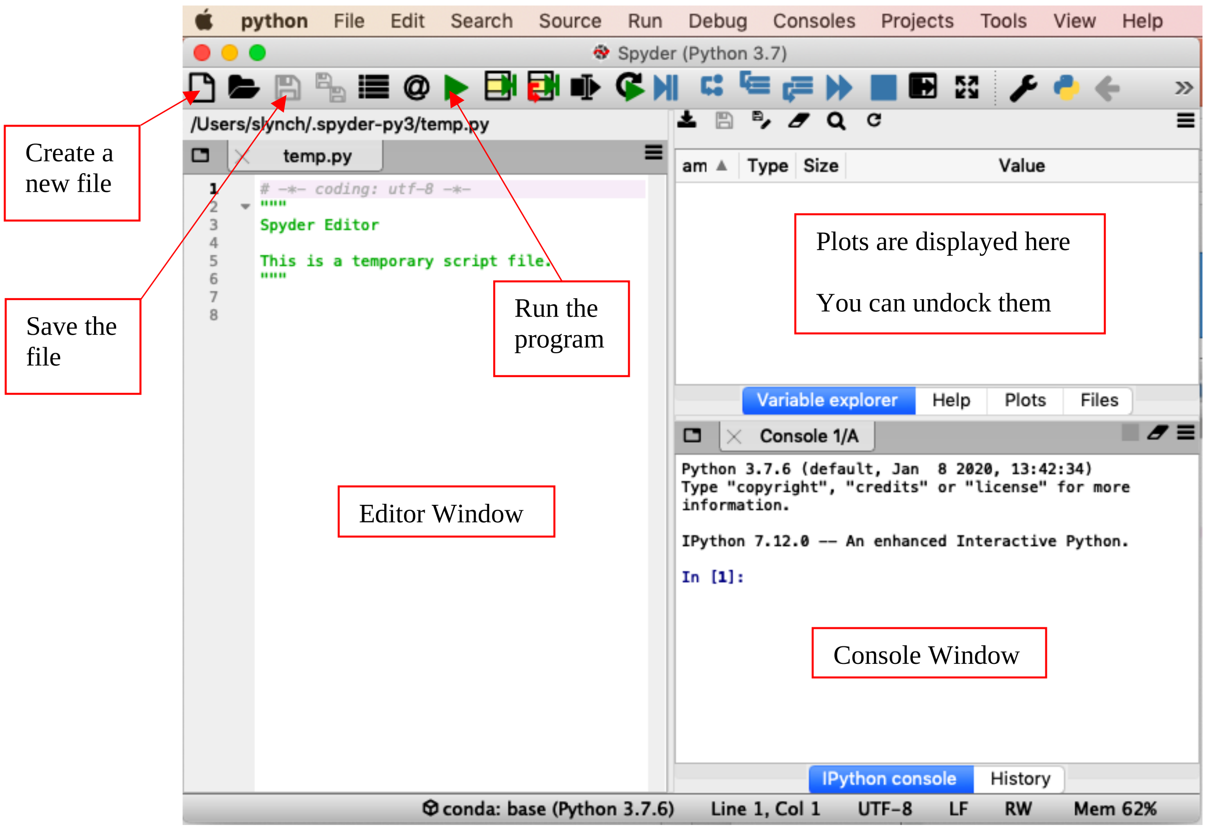
A government report published in 2018, the Bond Review:

https://www.ukri.org/wp-content/uploads/2022/07/EPSRC-050722-TheEraMathematics.pdf

recommended that all Mathematics undergraduates and postgraduates should have a working knowledge of at least one programming language. Programming and Mathematics will make you extremely employable.

2.2. Introduction to Programming

Chapter 1 introduced you to the Console Window and its interactive nature. More involved tasks will require more lines of code and it is better to use Python program files. To open a new script file, click on the 'New' button under the File tab. To save the file and give it a name click on the Save button.



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
```

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

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]:
```

We will introduce three programming structures:

1. Defining functions.
2. Loops for repetitive tasks.
3. Conditional statements – if, elif, else etc.

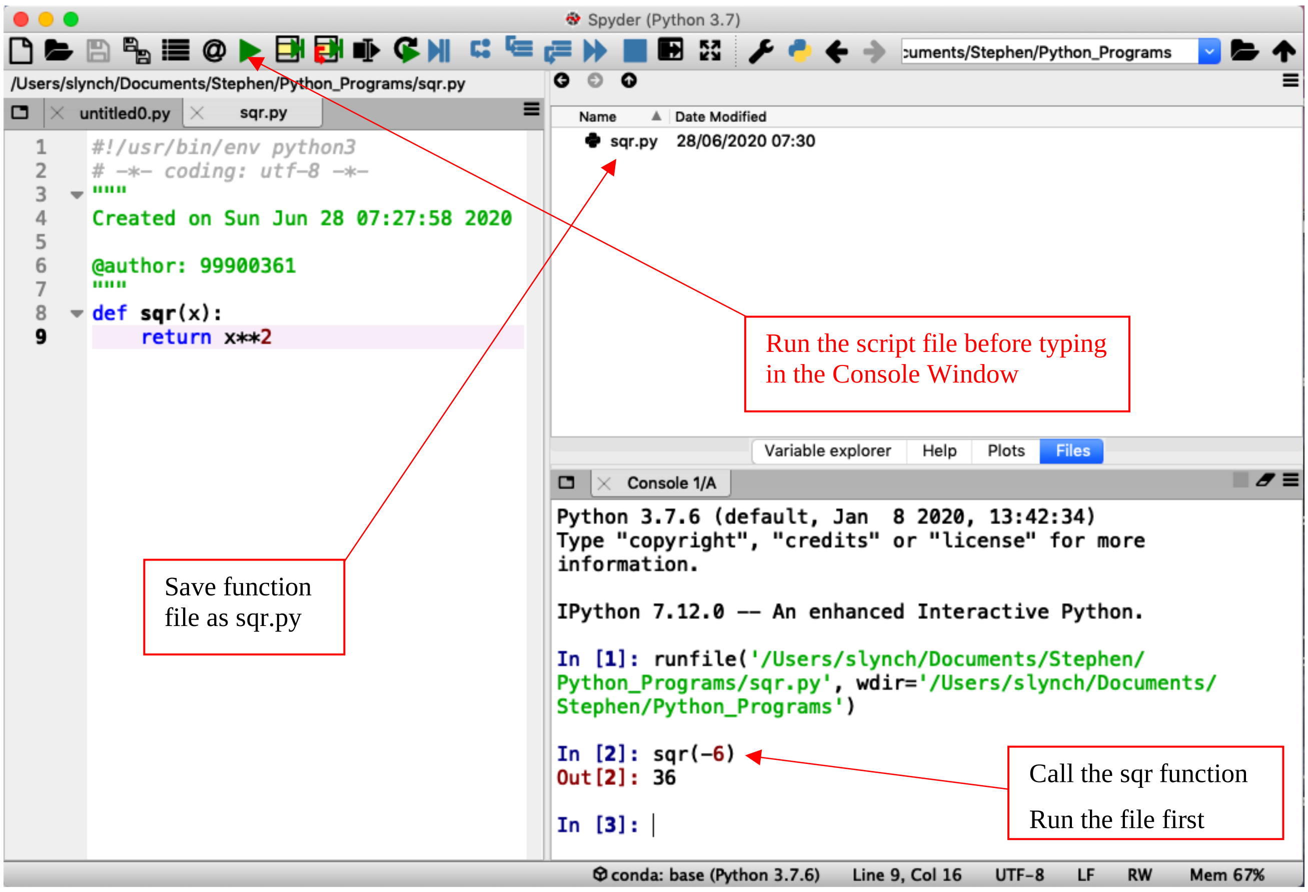
With these three constructs you can create any program that will be required on any Mathematics course!

2.3. Defining Functions (def)

You have already been using functions in Chapter 1. Now let us define our own functions. The functions are displayed in the Editor window as it would appear in Spyder.

Example 1 For your first program, define a function that squares a number and label the function sqr.

Solution The def command defines the function. At the end of the def line one types a colon and Python automatically indents the line after you hit ENTER. In the final line, one types return followed by your choice of function.



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 28 07:27:58 2020

@author: 99900361
"""
def sqr(x):
    return x**2
```

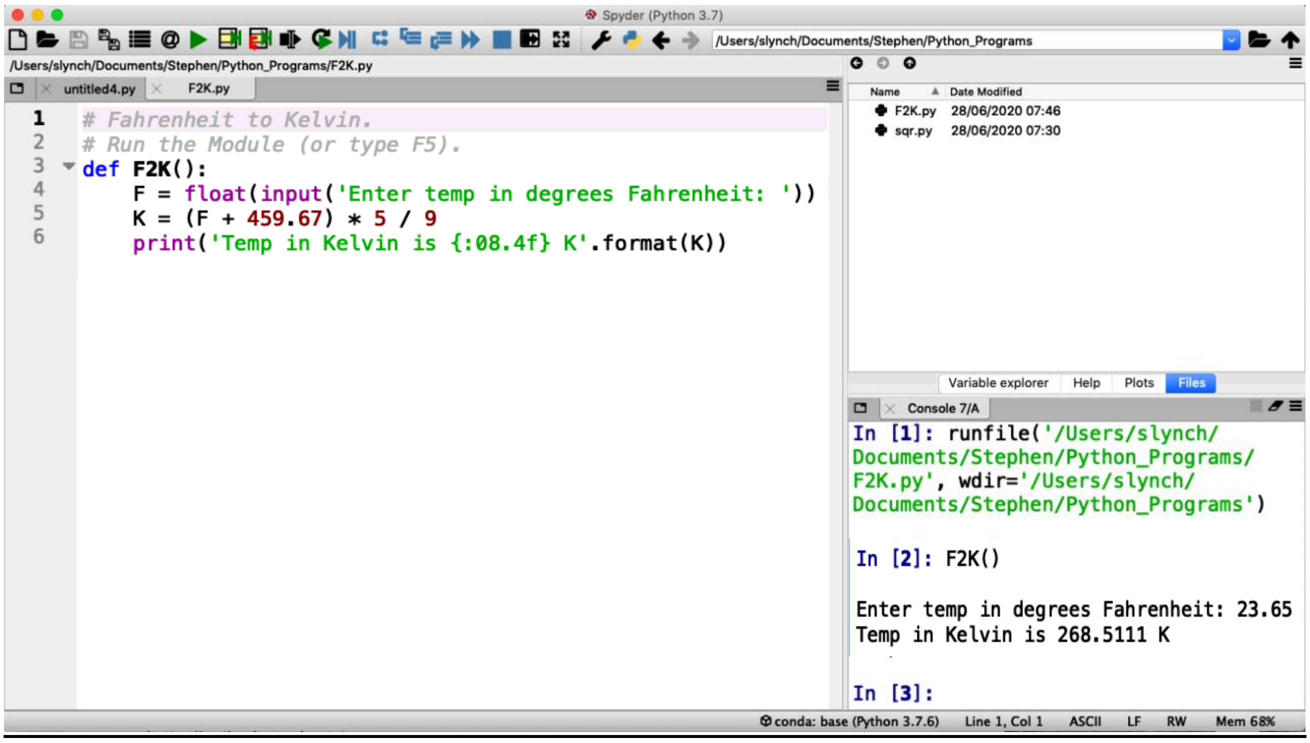
```
Python 3.7.6 (default, Jan  8 2020, 13:42:34)
Type "copyright", "credits" or "license" for more
information.

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/slynch/Documents/Stephen/
Python_Programs/sqr.py', wdir='/Users/slynch/Documents/
Stephen/Python_Programs')

In [2]: sqr(-6)
Out[2]: 36

In [3]:
```

Essentially, you are creating a new button (function) on your calculator and calling it sqr.

IMPORTANT: You MUST first run the script file (hit the green play button or type F5) in order to call the function in the Console Window.

<u>Example 2</u> Write a Python function file that converts degrees Fahrenheit to Kelvin. Output the results in floating point format.

<u>Solution</u>



```
# Fahrenheit to Kelvin.
# Run the Module (or type F5).
def F2K():
    F = float(input('Enter temp in degrees Fahrenheit: '))
    K = (F + 459.67) * 5 / 9
    print('Temp in Kelvin is {:08.4f} K'.format(K))
```

```
In [1]: runfile('/Users/slynch/Documents/Stephen/Python_Programs/F2K.py', wdir='/Users/slynch/Documents/Stephen/Python_Programs')

In [2]: F2K()

Enter temp in degrees Fahrenheit: 23.65
Temp in Kelvin is 268.5111 K

In [3]:
```

Within print, {:08.4f} specifies a floating-point number with a field width of eight digits, including four digits after the decimal point.

<u>2.4. Loops for Repetitive Tasks (while and for)</u>

Loops are used to repeat a block of code. In Python we use while and for loops.

The syntax for these loops is:

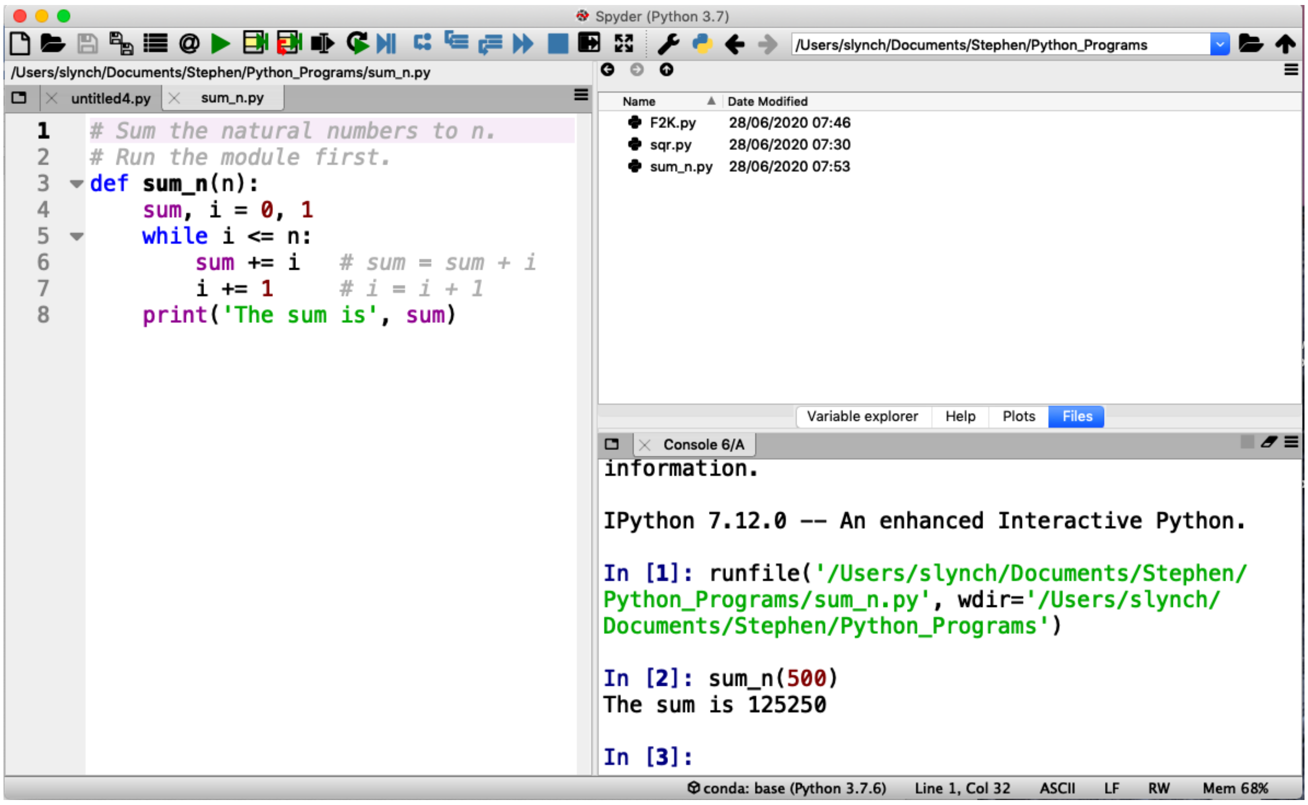
The while loop:	**The for loop:**

while *expression:*
    statements
	for *index* in *range(N):*
	    statements

Notice again that Python automatically indents after you hit ENTER after typing the colon. We will illustrate these loops with examples.

Example 3 Write a Python function file that sums the first n natural numbers using a while loop.

Solution

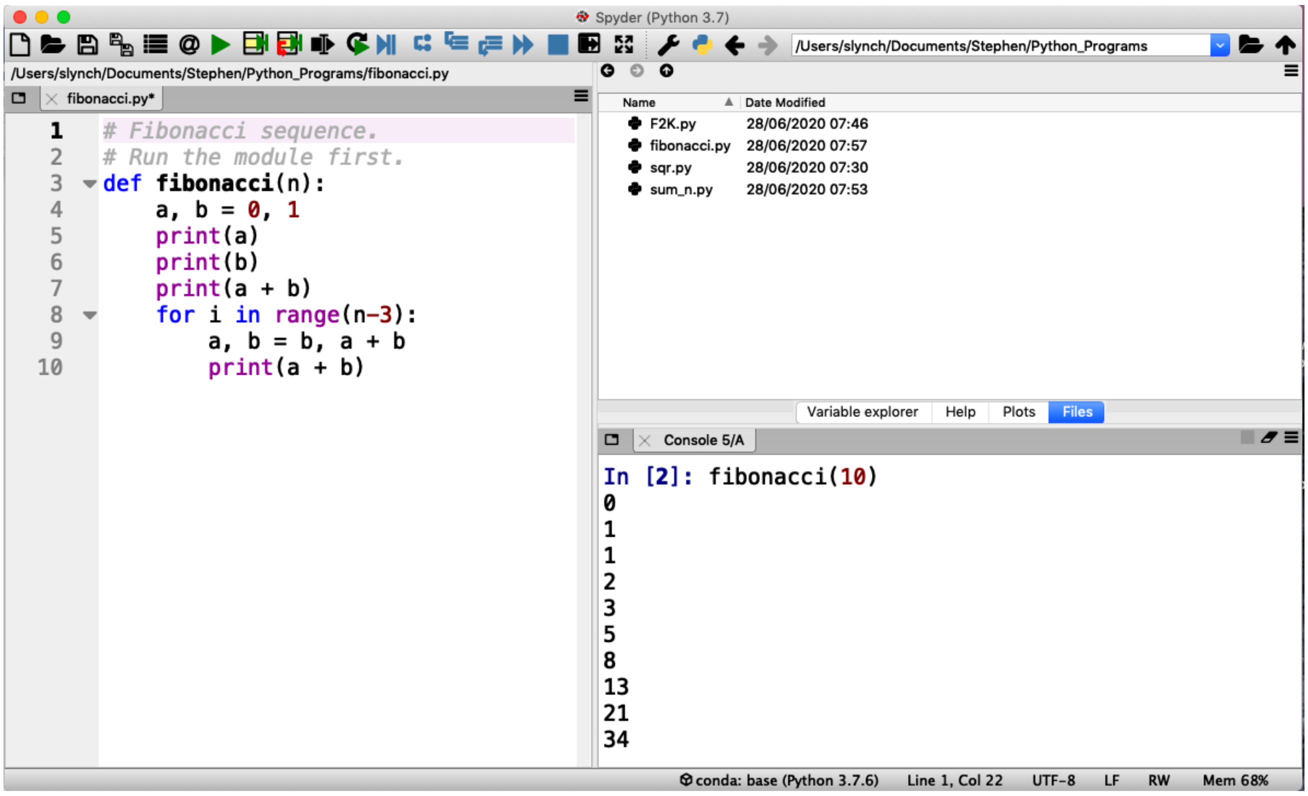


```python
# Sum the natural numbers to n.
# Run the module first.
def sum_n(n):
    sum, i = 0, 1
    while i <= n:
        sum += i    # sum = sum + i
        i += 1      # i = i + 1
    print('The sum is', sum)
```

```
information.

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/slynch/Documents/Stephen/
Python_Programs/sum_n.py', wdir='/Users/slynch/
Documents/Stephen/Python_Programs')

In [2]: sum_n(500)
The sum is 125250

In [3]:
```

Example 4 Write a Python program that lists the first n terms of the Fibonacci sequence using a for loop.

Solution


Spyder (Python 3.7)
/Users/slynch/Documents/Stephen/Python_Programs
/Users/slynch/Documents/Stephen/Python_Programs/fibonacci.py
fibonacci.py*
Name
Date Modified
F2K.py 28/06/2020 07:46
fibonacci.py 28/06/2020 07:57
sqr.py 28/06/2020 07:30
sum_n.py 28/06/2020 07:53
Variable explorer
Help
Plots
Files
Console 5/A
conda: base (Python 3.7.6)
Line 1, Col 22
UTF-8
LF
RW
Mem 68%

```python
# Fibonacci sequence.
# Run the module first.
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)
```

```
In [2]: fibonacci(10)
0
1
1
2
3
5
8
13
21
34
```

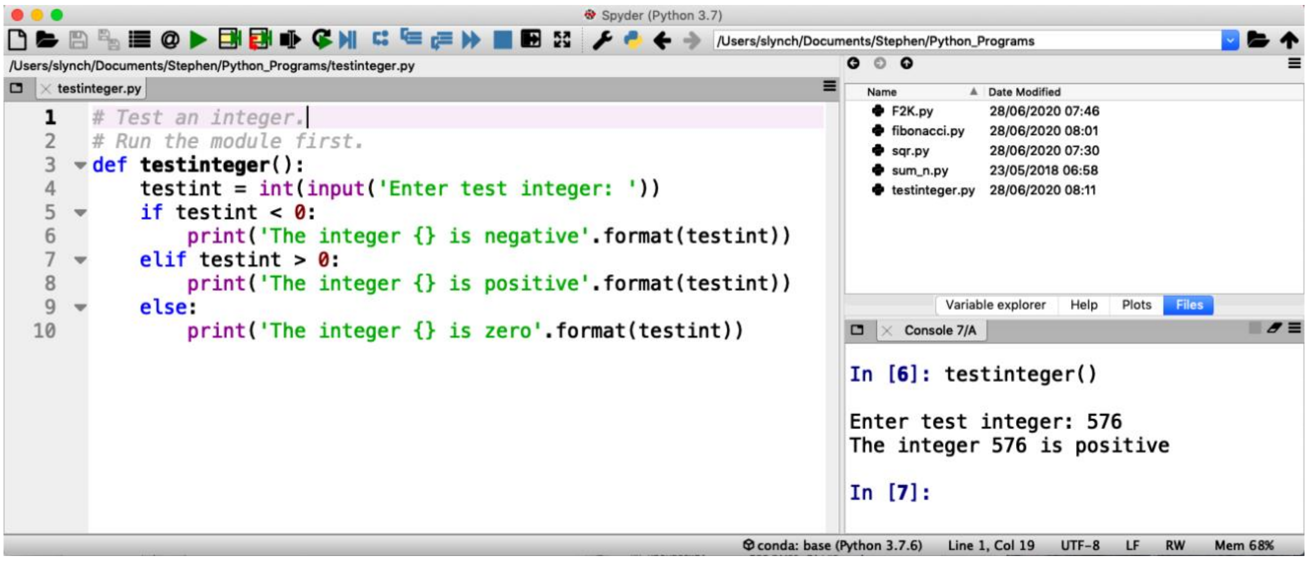
2.5. Conditional Statements (if, elif, else)

In Python, the syntax for the if command is:

if *expression:*
    statements
elif *expression:*
    statements
else:
    statements

Example 5 Write a Python function that tests whether an integer is zero, positive or negative.

Solution



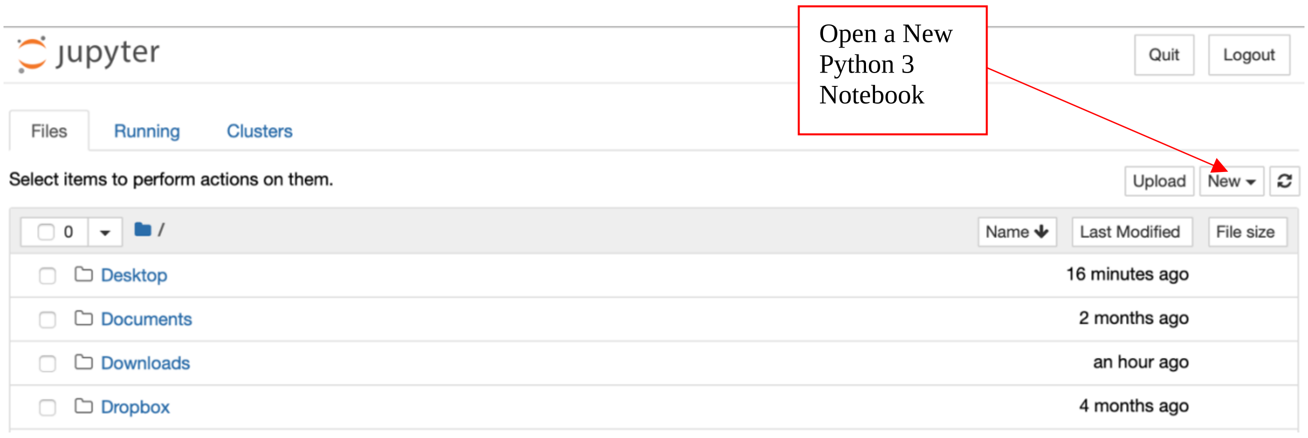
```
# Test an integer.
# Run the module first.
def testinteger():
    testint = int(input('Enter test integer: '))
    if testint < 0:
        print('The integer {} is negative'.format(testint))
    elif testint > 0:
        print('The integer {} is positive'.format(testint))
    else:
        print('The integer {} is zero'.format(testint))
```

```
In [6]: testinteger()

Enter test integer: 576
The integer 576 is positive

In [7]:
```

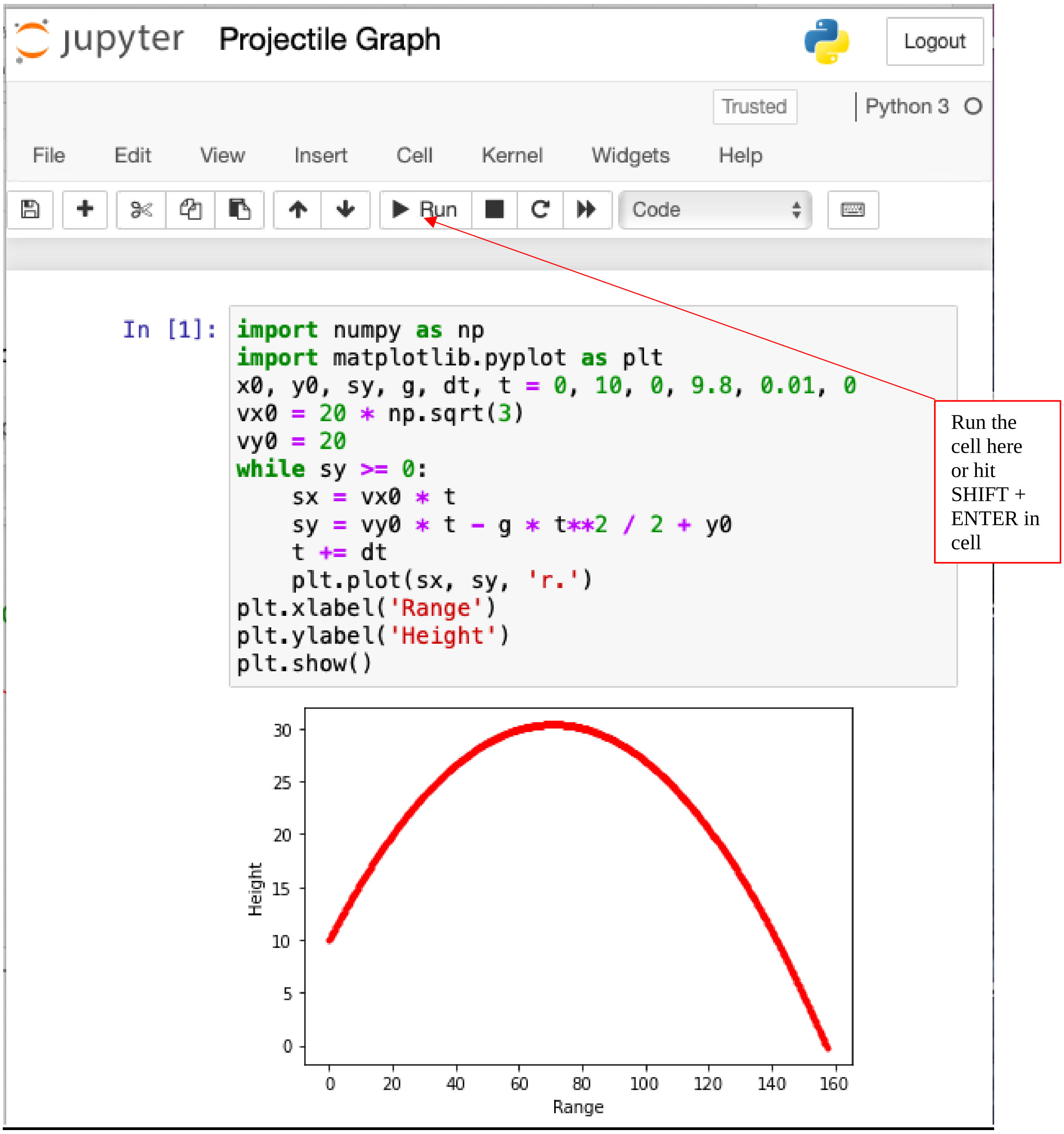
Spyder (Python 3.7)
/Users/slynch/Documents/Stephen/Python_Programs
/Users/slynch/Documents/Stephen/Python_Programs/testinteger.py
testinteger.py
Name
Date Modified
F2K.py 28/06/2020 07:46
fibonacci.py 28/06/2020 08:01
sqr.py 28/06/2020 07:30
sum_n.py 23/05/2018 06:58
testinteger.py 28/06/2020 08:11
Variable explorer
Help
Plots
Files
Console 7/A
conda: base (Python 3.7.6)
Line 1, Col 19
UTF-8
LF
RW
Mem 68%

2.6. Jupyter Notebooks and Google Colab

Launch a jupyter notebook by clicking on the icon in the Anaconda Navigator Window (see Section 1.2). The following window opens on your web browser:



You can run Python commands and programs within the jupyter notebook.



In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
x0, y0, sy, g, dt, t = 0, 10, 0, 9.8, 0.01, 0
vx0 = 20 * np.sqrt(3)
vy0 = 20
while sy >= 0:
    sx = vx0 * t
    sy = vy0 * t - g * t**2 / 2 + y0
    t += dt
    plt.plot(sx, sy, 'r.')
plt.xlabel('Range')
plt.ylabel('Height')
plt.show()
```

You can run jupyter notebooks freely online through Google Colab (you need a Google account):

https://colab.research.google.com

In this case you can perform cloud computing without having Python on your computer!

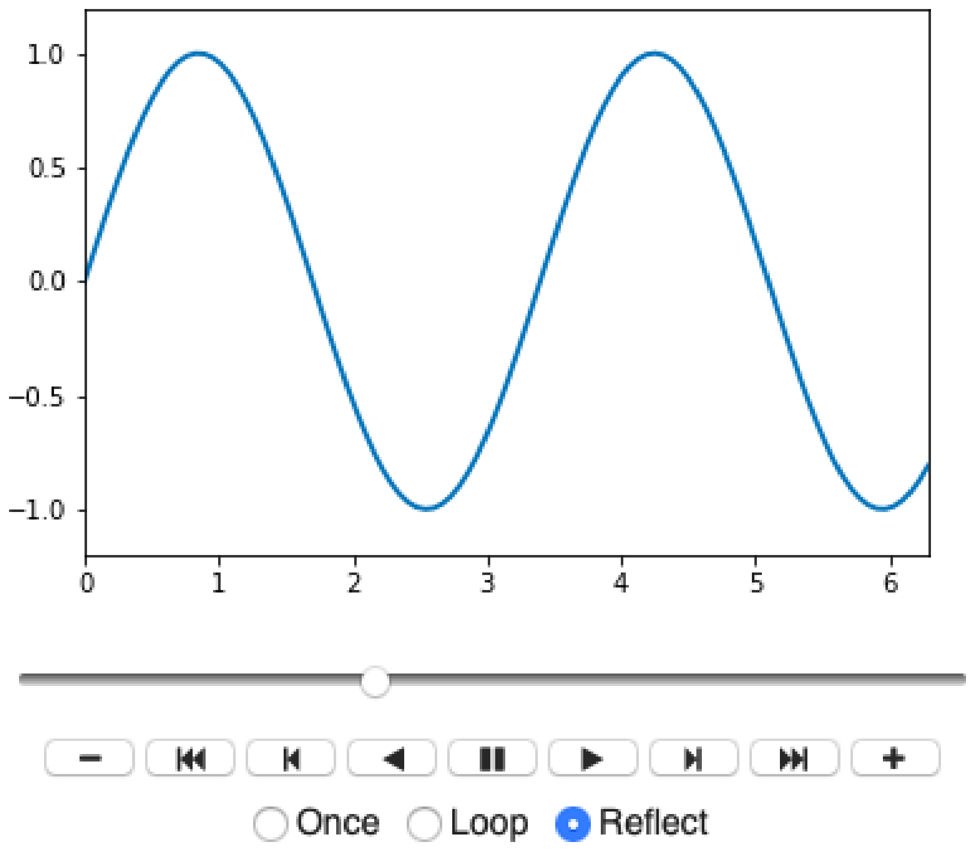
An Example in Google Colab

The code below animates the function:

$$y = sin(0.05xn),$$

where n ranges from 1 to 100 (frames = 101). A slider is given below the figure.

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation, rc
from IPython.display import HTML

# Set up figure.
fig, ax = plt.subplots()
plt.close()
# Set domain and range.
ax.set_xlim(( 0, 2 * np.pi))
ax.set_ylim((- 1.2, 1.2))
# Set line width.
line, = ax.plot([], [], lw=2)
# Initialization function: plot the background of each frame.
def init():
    line.set_data([], [])
    return (line,)
# Animation function. This is called sequentially.
def animate(n):
    x = np.linspace(0, 2 * np.pi, 100)
    y = np.sin(0.05 * n * x)
    line.set_data(x, y)
    return (line,)
# Animate, interval sets the speed of animation.
anim = animation.FuncAnimation(fig, animate, init_func=init,
                               frames=101, interval=100, blit=True)
# Note: below is the part which makes it work on Colab.
rc('animation', html='jshtml')
anim
```



2.7. Exercises

2.7.1 Write a Python program file for each of the following:

(a) A function called sgn that returns -1 if $x < 0$, 0 if $x = 0$ and $+1$ if $x > 0$. This is the signum function.

(b) A program that returns the mean, median and mode of a list of inputted numbers.

(c) A function that returns the sum of the squares of the first n natural numbers. Use the sqr function given in Example 1.

(d) A function called cylinder_volume that inputs two variables, the radius and height of a cylinder, and gives the volume of the cylinder.

(e) A function called cone_mass that inputs three variables, the height, base-radius and density, and returns the mass of the cone.

2.7.2 Write a Python program that animates the parametric curve defined by $x(t) = \cos(t)\,, y(t) = \sin\,(nt)$, for n from 0 to 4, and $0 \leq t < 2\pi$.

2.7.3 (a) Write a Python program to convert degrees Centigrade to degrees Fahrenheit.

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

2.7.4 Write an interactive Python program to play a "guess the number" game. The computer should think of a random natural number between 1 and 20 and the user (player) has to try to guess the number within six attempts. The program should tell the player if the guess is too high or too low.

2.7.5 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:

https://drstephenlynch.github.io/webpages/Python_for_A_Level_Maths_and_Beyond_Solutions.html