Mid-Module Questionnaire

- This is your chance to have a say on how the course is run and raise any issues you may have
  - There will also be another questionnaire at the end of the module
- We will take your views on board
  - Reporting back common items in the lecture and canvas
  - Making changes to alleviate issues where possible
- Please pick up a form from the front of the lecture theatre
  - And take 5 minutes to fill this is now
- At the end of the lecture please bring your forms to the front.

Attendance code: 277907

Introduction to Computational Physics (PHYS105)

Lecture 7: Good Practice and Debugging

Carl Gwilliam

(C.Gwilliam@liverpool.ac.uk)

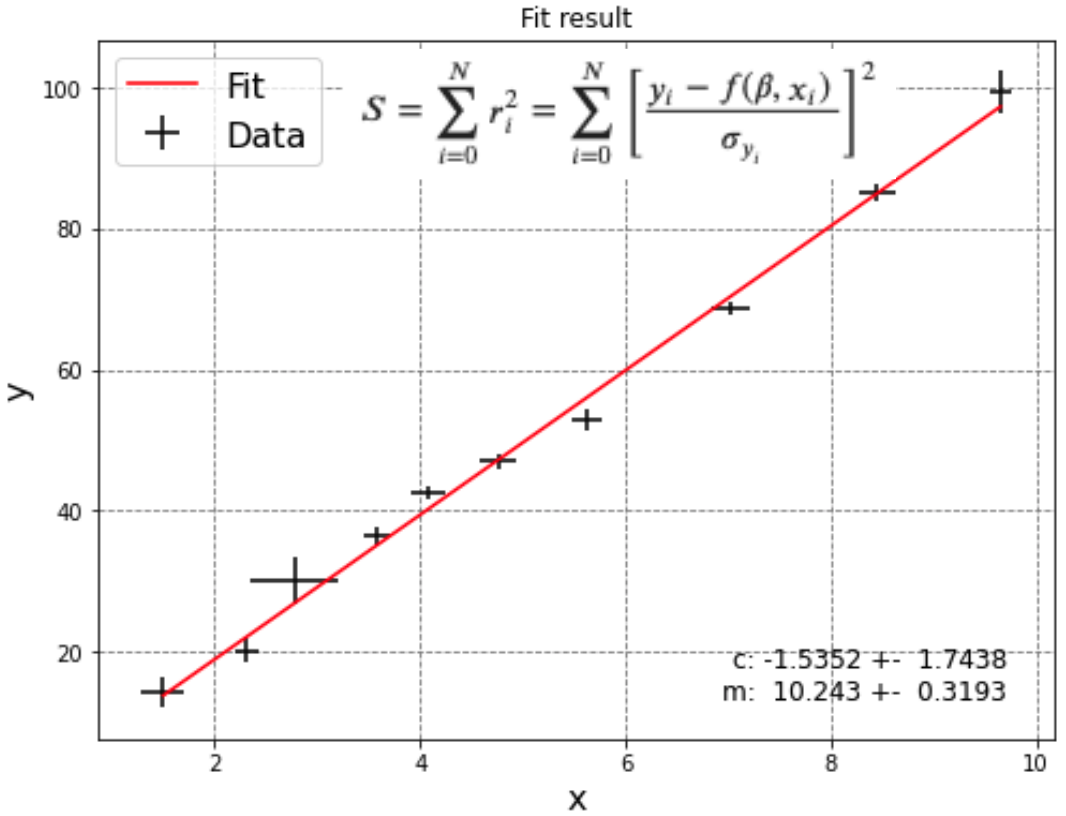
UNIVERSITY OF LIVERPOOL

Lecture 6: Recap

- Last week we started the second part of the course, using python for real physics problems!

- Started with fitting a functional form to data.
  - Read in data from files (using `np.loadtxt`)
    - Plot to check sensible
    - Find initial params
  - Define necessary functions
    - Functional form and its differential
    - Point-by-point residuals
  - Run `least_squares()` from scipy.optimize
    - Check fit succeeds and get fitted params
  - Calculate χ^2 test statistic from the residuals
    - Check fit is good e.g. $0.25 < \chi^2/\mathrm{NDF} < 4$
  - Calculate parameter errors
    - Print best-fit parameters and associated error
  - Plot data and fitted function
    - Passing best fit params to functional form

- Encapsulated all this in a reusable function



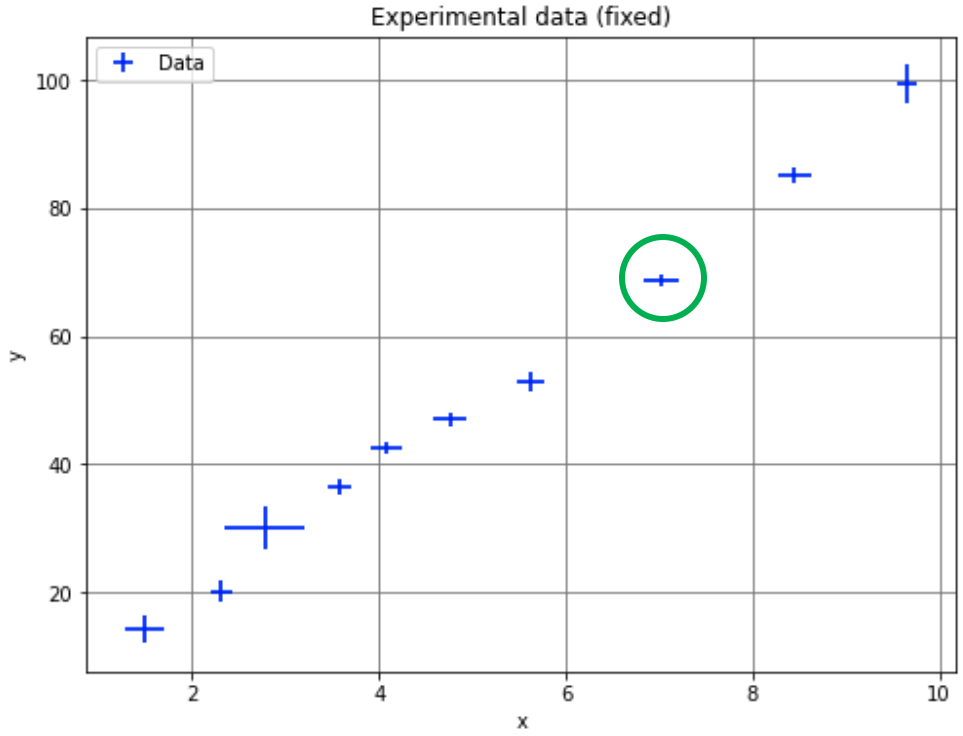
- You will use this over-and-over in labs
  - Will create your own python module for this in the last week of PHYS105

Lecture 6: Formative problem solutions

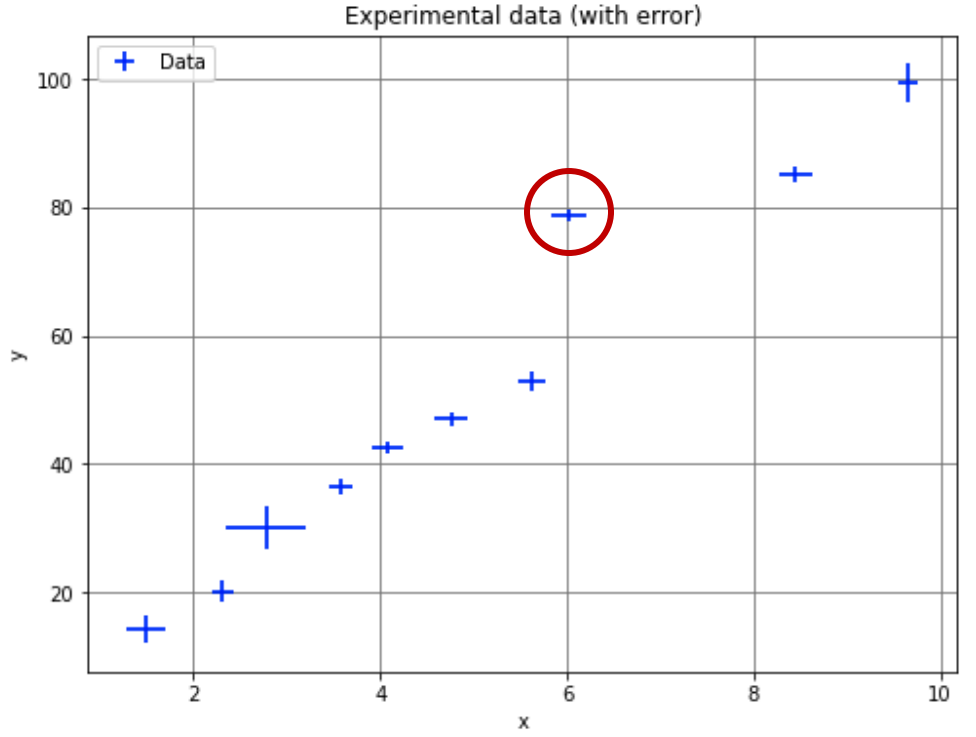
- Exercise 1: Plot the experimental data and fix any errors compared to the table above
  - Plot data read from csv file using `errorbar()`
  - Clear that 8th point is incorrect → fix by updating array value at **index** 7 and replot

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize = (8, 6))
plt.title("Experimental data (with error)")
plt.xlabel("x")
plt.ylabel("y")
plt.errorbar(xdata, ydata, xerr=xerror, yerr=yerror,
             color = 'b', linestyle = '', label = 'Data')
plt.grid(color = 'grey')
plt.legend(loc = 2)
plt.show()
```




```
xdata[7] = 7.02
ydata[7] = 68.8
```



Lecture 6: Formative problem solutions (2)

- Exercise 2: Fit extended data

Read data

```
# Extend data
import numpy as np
xdata2 = np.append(xdata, [10.25, 11.85])
ydata2 = np.append(ydata, [110.5, 122.1])
xerror2 = np.append(xerror, [0.21, 0.41])
yerror2 = np.append(yerror, [3.1, 3.3])
```

Run fit and check it succeeds

```
import numpy as np
from scipy.optimize import least_squares
from scipy.stats import chi2 as stats_chi2
import matplotlib.pyplot as plt
%matplotlib inline

# Set initial params
init_params = [0.0, 10.0]

# Run fit
result = least_squares(minimise, init_params,
                       args=(xdata2, ydata2, xerror2, yerror2))

# Check fit succeeds
if not result.success or result.status < 1:
    print ("ERROR: Fit failed with message {}".format(result.message))
    print ("Please check the data and inital parameter estimates")
else:
    print ("Fit succeeded")
```

New data

Get fitted parameters

```
# Get fitted parameters
final_params = result.x
c = final_params[0]
m = final_params[1]
nparams = len(final_params)
```

Calculate χ^2 and check fit is good

```
# Calculate chi2
chi2_array = result.fun ** 2
chi2 = sum(chi2_array)
npoints = len(xdata)
reduced_chi2 = chi2 / (npoints - nparams)
chi2_prob = stats_chi2.sf(chi2, (npoints - nparams))

# Print chi2
np.set_printoptions(precision = 3)
print("\n=== Fit quality ===")
print("chisq per point = \n",chi2_array)
print("chisq = {:7.5g}, ndf = {}, chisq/NDF = {:7.5g}, chisq prob = {:7.5g}\n".
      format(chi2, npoints-nparams, reduced_chi2, chi2_prob))

if reduced_chi2 < 0.25 or reduced_chi2 > 4:
    print("WARNING: chi2/ndf suspiciously small or large."
          "Please check the data and initial parameter estimates")

if chi2_prob < 0.05:
    print("WARNING: chi2 probability for given degrees of freedom less than 0.05."
          "Please check the data and initial parameter estimates")
```

Calculate param errors and print

```
# Calculate errors
jacobian = result.jac
jacobian2 = np.dot(jacobian.T, jacobian)
determinant = np.linalg.det(jacobian2)

if determinant < 1E-32:
    print(f"Matrix singular (determinant = {determinant}, error calculation failed.")
    param_errors = np.zeros(nparams)
else:
    covariance = np.linalg.inv(jacobian2)
    param_errors = np.sqrt(covariance.diagonal())

print ("c = {:7.5g} +- {:7.5g}".format(final_params[0], param_errors[0]))
print ("m = {:7.5g} +- {:7.5g}".format(final_params[1], param_errors[1]))
```

Lecture 6: Formative problem solutions (3)

- Exercise 2 (cont): Fit extended data

```
# Calculate fitted function values
yfit = straight_line(final_params, xdata2)

# Visualise result
fig = plt.figure(figsize = (8, 6))
plt.title('Fit result')
plt.xlabel('x', fontsize=16)
plt.ylabel('y', fontsize=16)
plt.grid(color = 'grey', linestyle="--")

plt.errorbar(xdata2, ydata2, xerr = xerror2, yerr = yerror2, fmt='k',
             linestyle = '', label = "Data")
plt.plot(xdata2, yfit, color = 'r', linestyle = '-', label = "Fit")

plt.legend(loc = 2, fontsize=16)

text = "c: {:7.5g} +- {:7.5g}\n".format(final_params[0], param_errors[0])
text += "m: {:7.5g} +- {:7.5g}\n".format(final_params[1], param_errors[1])
plt.text(0.95, 0.0, text, transform = fig.axes[0].transAxes, ha = "right",
         va = "bottom", fontsize=12)

plt.show()
```

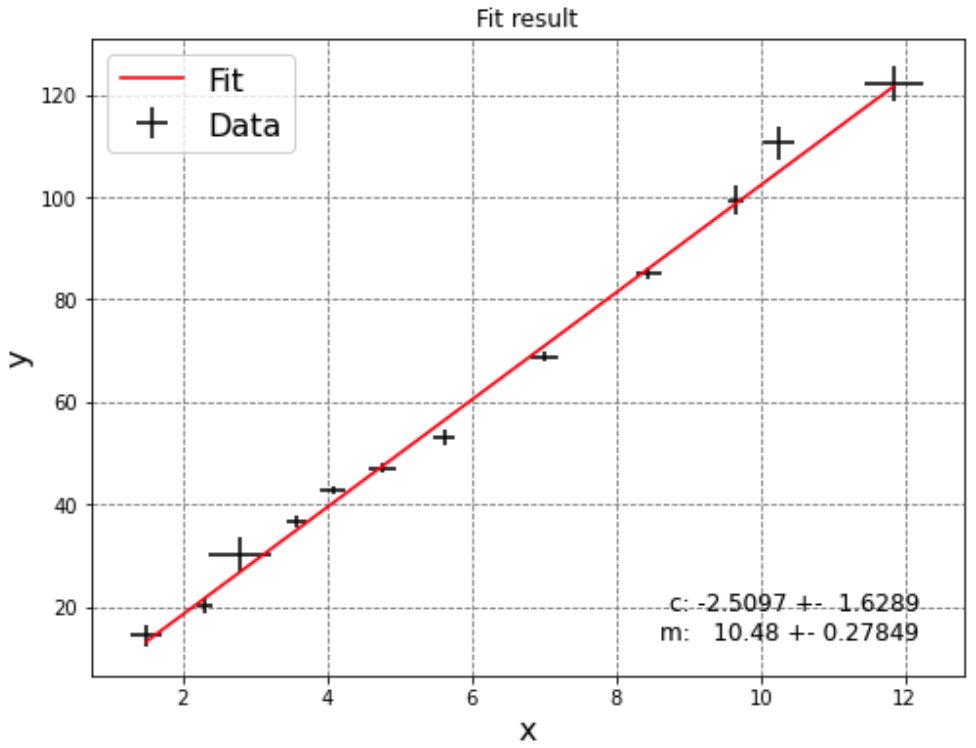
Evaluate best fit y and plot

New data!

```
Fit succeeded

=== Fit quality ===
chisq per point =
 [0.128 0.533 0.387 0.724 1.507 0.016 2.579 1.072 0.156 0.062 2.159 0.006]
chisq =   9.329, ndf = 10, chisq/NDF =  0.9329, chisq prob = 0.50119

c = -2.5097 +-  1.6289
m =   10.48 +- 0.27849
```



Lecture 6: Formative problem solutions (4)

- Exercise 3: Does straight line still fit the data
  - Yes, as shown by χ^2:
    - χ^2/ndf between ¼ and 4 `chisq/NDF = 1.1661`
    - χ^2 prob greater than 5% `chisq prob = 0.3153`

- Have the gradient or intercept changed significantly?
  - No, values are still compatible within errors
  - Quantify by "Consistency Check" (as in labs)

$$\text{Significance} = \frac{|x_1 - x_2|}{\sqrt{\sigma_{x1}^2 + \sigma_{x_2}^2}} < 3$$

Total error

```
from math import sqrt

def sigma(x1, x2, e1, e2):
    "Calculate number of standard deviations"
    return abs(x1-x2)/sqrt(e1**2 + e2**2)

final_params_10pts = [-1.535, 10.243]
param_errors_10pts = [1.744, 0.319]

csig = sigma(final_params[0], final_params_10pts[0],
             param_errors[0], param_errors_10pts[0])
msig = sigma(final_params[1], final_params_10pts[1],
             param_errors[1], param_errors_10pts[1])

print (f"Significance for c = {csig:.2f} and m = {msig:.2f}")
if csig < 3 and msig < 3:
    print("Values are consistent")
```

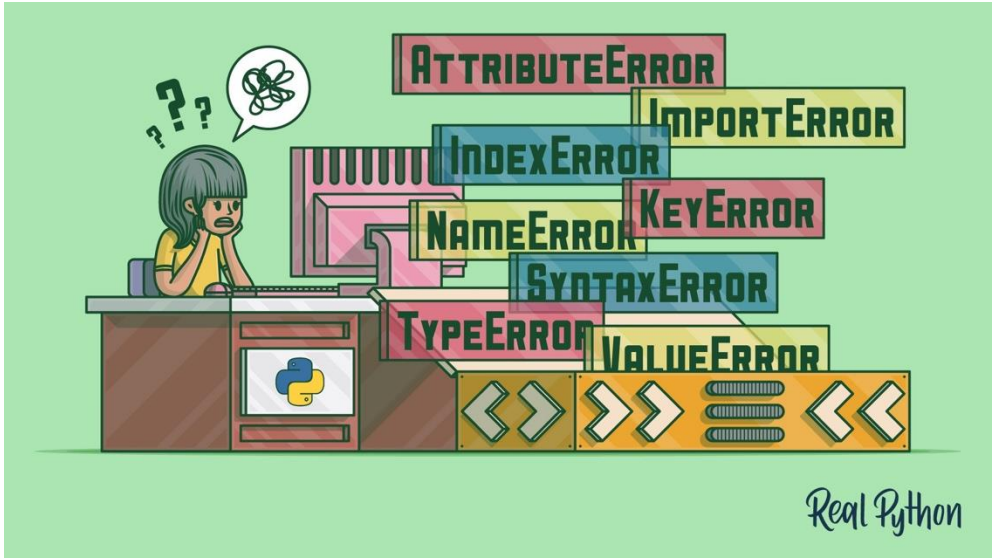
```
Significance for c = 0.41 and m = 0.56
Values are consistent
```

- Now that we have developed our first real physics program, it is time to take a step back …

Python errors

- At many points during the course you will have come across issues with your code
  - Where python has given you a big fat error message or you get the wrong result

- We can try to avoid these via good coding practice
  - Using sensible variable names,
  - Writing useful comments,
  - Checking inputs/values, etc



- But no matter how good we are, as we write more complex programs we'll encounter them more often

- There are several different types of error we may encounter
  - Code that is syntactically incorrect and therefore won't parse at all
  - Programs that start but give a run-time error and don't complete
  - Programs that run but produce incorrect results

Harder to debug

- This week, we are going to look in detail at how to understand these errors and debug them

Understanding and debugging error messages

- Real example saw during week 4 PC class

```
1  def rectPrismParams(width, length, height):
2      '''
3      Return volume, surface area and length of edges of
4      rectangular prism given its width,length and height
5      '''
6      vol = width * length * height
7      area = 2 (width * length + width * height + length * height)
8      edge = 4 (width + length + height)
9      return vol, area, edge
10
11 w,l,h = 0.3,0.17,0.25
12 V, A, s = rectPrismParams(w, l, h)
13
14 print(f"Volume = {V:.2f}")
15 print(f"Area   = {A:.2f}")
16 print(f"Edges  = {s:.2f}")
```

- Can turn on line numbers by
  View → Line Numbers → Show

Understanding and debugging error messages

- Real example saw during week 4 PC class

```
1  def rectPrismParams(width, length, height):
2      '''
3      Return volume, surface area and length of edges of
4      rectangular prism given its width,length and height
5      '''
6      vol = width * length * height
7      area = 2 (width * length + width * height + length * height)
8      edge = 4 (width + length + height)
9      return vol, area, edge
10
11 w,l,h = 0.3,0.17,0.25
12 V, A, s = rectPrismParams(w, l, h)
13
14 print(f"Volume = {V:.2f}")
15 print(f"Area   = {A:.2f}")
16 print(f"Edges  = {s:.2f}")
```

- Can turn on line numbers by View → Line Numbers → Show

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-2185f7667564> in <module>
     10
     11 w,l,h = 0.3,0.17,0.25
---> 12 V, A, s = rectPrismParams(w, l, h)
     13
     14 print(f"Volume = {V:.2f}")

<ipython-input-39-2185f7667564> in rectPrismParams(width, length, height)
      5     '''
      6     vol = width * length * height
----> 7     area = 2 (width * length + width * height + length * height)
      8     edge = 4 (width + length + height)
      9     return vol, area, edge

TypeError: 'int' object is not callable
```

Understanding and debugging error messages

- Real example saw during week 4 PC class

```
1  def rectPrismParams(width, length, height):
2      '''
3      Return volume, surface area and length of edges of
4      rectangular prism given its width,length and height
5      '''
6      vol = width * length * height
7      area = 2 (width * length + width * height + length * height)
8      edge = 4 (width + length + height)
9      return vol, area, edge
10
11 w,l,h = 0.3,0.17,0.25
12 V, A, s = rectPrismParams(w, l, h)
13
14 print(f"Volume = {V:.2f}")
15 print(f"Area   = {A:.2f}")
16 print(f"Edges  = {s:.2f}")
```

- Can turn on line numbers by View → Line Numbers → Show

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-2185f7667564> in <module>
     10
     11 w,l,h = 0.3,0.17,0.25
---> 12 V, A, s = rectPrismParams(w, l, h)
     13
     14 print(f"Volume = {V:.2f}")

<ipython-input-39-2185f7667564> in rectPrismParams(width, length, height)
      5     '''
      6     vol = width * length * height
----> 7     area = 2 (width * length + width * height + length * height)
      8     edge = 4 (width + length + height)
      9     return vol, area, edge

TypeError: 'int' object is not callable
```

Read from bottom to top

Understanding and debugging error messages

- Real example saw during week 4 PC class

```
1  def rectPrismParams(width, length, height):
2      '''
3      Return volume, surface area and length of edges of
4      rectangular prism given its width,length and height
5      '''
6      vol = width * length * height
7      area = 2 (width * length + width * height + length * height)
8      edge = 4 (width + length + height)
9      return vol, area, edge
10
11 w,l,h = 0.3,0.17,0.25
12 V, A, s = rectPrismParams(w, l, h)
13
14 print(f"Volume = {V:.2f}")
15 print(f"Area   = {A:.2f}")
16 print(f"Edges  = {s:.2f}")
```

- Can turn on line numbers by View → Line Numbers → Show

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-2185f7667564> in <module>
     10
     11 w,l,h = 0.3,0.17,0.25
---> 12 V, A, s = rectPrismParams(w, l, h)
     13
     14 print(f"Volume = {V:.2f}")

<ipython-input-39-2185f7667564> in rectPrismParams(width, length, height)
      5     '''
      6     vol = width * length * height
----> 7     area = 2 (width * length + width * height + length * height)
      8     edge = 4 (width + length + height)
      9     return vol, area, edge

TypeError: 'int' object is not callable
```

Read from bottom to top

- The specific type of error that was encountered
  - In this case a "TypeError"
  - Python is telling us the type of the object is incorrect

Understanding and debugging error messages

- Real example saw during week 4 PC class

```
1  def rectPrismParams(width, length, height):
2      '''
3      Return volume, surface area and length of edges of
4      rectangular prism given its width,length and height
5      '''
6      vol = width * length * height
7      area = 2 (width * length + width * height + length * height)
8      edge = 4 (width + length + height)
9      return vol, area, edge
10
11 w,l,h = 0.3,0.17,0.25
12 V, A, s = rectPrismParams(w, l, h)
13
14 print(f"Volume = {V:.2f}")
15 print(f"Area   = {A:.2f}")
16 print(f"Edges  = {s:.2f}")
```

- Can turn on line numbers by View → Line Numbers → Show

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-2185f7667564> in <module>
     10
     11 w,l,h = 0.3,0.17,0.25
---> 12 V, A, s = rectPrismParams(w, l, h)
     13
     14 print(f"Volume = {V:.2f}")

<ipython-input-39-2185f7667564> in rectPrismParams(width, length, height)
      5     '''
      6     vol = width * length * height
----> 7     area = 2 (width * length + width * height + length * height)
      8     edge = 4 (width + length + height)
      9     return vol, area, edge

TypeError: 'int' object is not callable
```

Read from bottom to top

- The specific type of error that was encountered
  - In this case a "TypeError"
  - Python is telling us the type of the object is incorrect
- The error message itself
  - In this case python thinks we are trying to call an integer (like we would do a function)

Understanding and debugging error messages

- Real example saw during week 4 PC class

```
1  def rectPrismParams(width, length, height):
2      '''
3      Return volume, surface area and length of edges of
4      rectangular prism given its width,length and height
5      '''
6      vol = width * length * height
7      area = 2 (width * length + width * height + length * height)
8      edge = 4 (width + length + height)
9      return vol, area, edge
10
11 w,l,h = 0.3,0.17,0.25
12 V, A, s = rectPrismParams(w, l, h)
13
14 print(f"Volume = {V:.2f}")
15 print(f"Area   = {A:.2f}")
16 print(f"Edges  = {s:.2f}")
```

- Can turn on line numbers by View → Line Numbers → Show

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-2185f7667564> in <module>
     10
     11 w,l,h = 0.3,0.17,0.25
---> 12 V, A, s = rectPrismParams(w, l, h)
     13
     14 print(f"Volume = {V:.2f}")

<ipython-input-39-2185f7667564> in rectPrismParams(width, length, height)
      5     '''
      6     vol = width * length * height
----> 7     area = 2 (width * length + width * height + length * height)
      8     edge = 4 (width + length + height)
      9     return vol, area, edge

TypeError: 'int' object is not callable
```

- The "traceback", showing how the code got to the point of the error through various function calls, going from most recent at the bottom to least recent at the top.
  - The lines indicated by the '--->' are the ones called; the others are for context.
  - In this case it calls `rectPrimsPamas(…)` on L12 & then gives an error in calculating the area on L7

Read from bottom to top

- The specific type of error that was encountered
  - In this case a "TypeError"
  - Python is telling us the type of the object is incorrect

- The error message itself
  - In this case python thinks we are tying to call an integer (like we would do a function)

Understanding and debugging error messages

- Real example saw during week 4 PC class

```
1  def rectPrismParams(width, length, height):
2      '''
3      Return volume, surface area and length of edges of
4      rectangular prism given its width,length and height
5      '''
6      vol = width * length * height
7      area = 2 (width * length + width * height + length * height)
8      edge = 4 (width + length + height)
9      return vol, area, edge
10
11 w,l,h = 0.3,0.17,0.25
12 V, A, s = rectPrismParams(w, l, h)
13
14 print(f"Volume = {V:.2f}")
15 print(f"Area   = {A:.2f}")
16 print(f"Edges  = {s:.2f}")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-2185f7667564> in <module>
     10
     11 w,l,h = 0.3,0.17,0.25
---> 12 V, A, s = rectPrismParams(w, l, h)
     13
     14 print(f"Volume = {V:.2f}")

<ipython-input-39-2185f7667564> in rectPrismParams(width, length, height)
      5     '''
      6     vol = width * length * height
----> 7     area = 2 (width * length + width * height + length * height)
      8     edge = 4 (width + length + height)
      9     return vol, area, edge

TypeError: 'int' object is not callable
```

- So, what's the mistake in the code here?
  - We are missing a multiplication sign * between the integer and the brackets!
  - Python sees this as trying to call a function: `2 ()`
  - But you can't call an integer, hence the error

- The best way to get familiar with this is practice → let's look at some in a notebook now …

Summary

- This week we looked at various python errors and how to understand and debug them

- In particular, in the lecture we looked at:
  - Errors that prevent the program running at all
  - Programs that start but don't complete

- Learnt more about how to read python error messages
  - And practiced this with several different error types

- In this week's notebook you will explore errors further:
  - How to reduce the scope for errors with good coding practice
  - Debugging more of the above errors yourself
  - Finally, some programs that run but give the incorrect result!

- You will make coding mistakes (I still make many!)
  - But with practice you can find and fix them quicker!