# Introduction to Scientific Computing

*Winter 2020*

In this assignment we focus on coding practices more common in science, like numerical methods, calculating statistics, and vectorization. You can use any Python libraries that are suitable for answering the question unless *excluded* in the question, but you are responsible for their correctness.

## 1 numpy operations

Consider the following definitions of variables

```
import numpy as np
a = 1
b = np.array([3.0,2.3,1.0])
c = np.array([3,.3,.03])
d = np.array([[2,4],[4,6],[7,8]])
f = [9,90,900]
```

Write Python code to **calculate and print** the following calculations

1. The sum of `a` and `b`
2. The sum of `c` and `b`
3. The sum of `c` and `d` (adding `c` to each column of `d`)
4. The sum of `b` and `f`
5. The type variable stored in `d`, and in `f`
6. The `len` of `d`, and the `len` of `f`.

## 2 numpy functions

Consider the following definitions of variables (the `array` word shows the type of these variables, a `numpy` array).

```
t= array([0.    , 0.1155, 0.2287, 0.3404, 0.4475,
       0.5546, 0.6607, 0.7753, 0.8871, 1.    ])
y= array([ 0.    ,  0.1655,  0.2009,  0.1124,
       -0.0873, -0.3996, -0.8197, -1.3977,
       -2.0856, -2.905 ])
```

Write Python code to **calculate and print** the following calculations

1. The average of all the values in `y`
2. The sample standard deviation of values in `y`
3. The differential of `y` with respect to `t`. i.e. $\frac{dy}{dt}$.
4. The integral of energy $\int E\,dt$, where $E = \frac{1}{2}y^2$.

One way of writing the differential $\frac{dy}{dt}$ numerically is to approximate the $dy$ with $\delta y_i = y_{i+1} - y_i$ where $y_i$ is the *ith* element of the array `y`, and similarly for $\delta t_i$. Then use $\frac{dt}{dy} \approx \frac{\delta y}{\delta t}$. You might need to use a `for` loop to calculate the value at each point in time and position.

One way of writing the integral is to calculate the value of $E(y_i)$, then multiply by $\delta t_i$ and sum all of the elements. *This method is known as the Riemann sum.*

In both cases, `numpy` provides functions that can assist you in the calculation. Consult the `numpy` documentation for help.

# 3 Implementing equations in Python

A common task in scientific computing is to implement an equation in Python code. Consider the equation for the period of a simple pendulum.

$$T = 2\pi\sqrt{\frac{l}{g}}$$

where $l$ is the pendulum length, $g$ is the acceleration due to gravity, and $T$ is the period of oscillation.

1. Write a function that **calculates and returns** a value for the acceleration due to gravity given a pendulum length and period of oscillation. Test your code for a pendulum of length 2.50m that oscillates with a period of 5.16s.
2. Real measurements have uncertainties. Write a second function to propagate uncertainties in pendulum length and oscillation period through to the uncertainty in the predicted gravitational accelerations. i.e. calculate $\delta g$ given $l \pm \delta l$ and $T \pm \delta T$. Test your code and print the uncertainty assuming $l = 2.40 \pm 0.01m$ and $T = 5.0 \pm 0.01s$. **Do not use the `uncertainties` package for this exercise.**

# 4 Conditionals

Conditionals are used to perform different calculations based on the input to the code. For example, a grading scheme that is Pass/Fail at 70% could be written as

```
if grade > 70 :
    mark = "pass"
else:
    mark = "fail"
```

According to the University of Toronto grading scheme the following table provides the conversion from numerical mark to letter grade and GPA:

| Numerical Scale of Mark | Letter Grade | Grade Point Value |
|---|---|---|
| 90-100 | A+ | 4.0 |
| 85-89 | A | 4.0 |
| 80-84 | A- | 3.7 |
| 77-79 | B+ | 3.3 |
| 73-76 | B | 3.0 |
| 70-72 | B- | 2.7 |
| 67-69 | C+ | 2.3 |
| 63-66 | C | 2.0 |
| 60-62 | C- | 1.7 |
| 57-59 | D+ | 1.3 |
| 53-56 | D | 1.0 |
| 50-52 | D- | 0.7 |
| 0-49 | F | 0 |

Write the function called `gpa` that takes a value corresponding to “Numerical Scale of Mark” and returns

"Grade Point Value".

# 5 Loops

Loops are used to perform the same calculation on difference elements of a collection (such as a list). The marks for forty (entirely fictional) students from PHY224 in 2019 are given in the `marks` list below. Calculate the average GPA for the class.

```
marks= array([72, 82, 72, 72, 79, 57, 59, 71, 66, 80,
       67, 62, 91, 74, 77, 62, 71, 78, 65, 80,
       70, 74, 70, 95, 76, 66, 85, 64, 79, 57,
       63, 78, 84, 78, 75, 73, 62, 69, 72, 87])
```

# 6 Data comparison

One common question when you make a new measurement of a *known* value is whether your measurement "agrees" with the *known* value. If we assume the known value has no uncertainty (or much smaller uncertainty) the question can be answered by comparing the difference between your measurement and the *known* value, relative the the uncertainty in your measurement.

For example, if you measure the mass of the electron to be $m_e \pm s_e$, and the known value is $m_{ke}$ you should compare the $|m_{ke} - m_e|$ to the uncertainty $s_e$. If the difference is smaller than your uncertainty you can reasonably state that the measurement agrees with the known value.

Write a function that takes your measurement with uncertainty, and the known value of a parameter, and returns `True` if they agree, and `False` if they don't.

Test your function on the following values, printing the result of each call to your function.

| Measurement | Uncertainty | Known value |
|---|---|---|
| 19.2 | 0.1 | 19.41 |
| 19.5 | 0.8 | 19.41 |
| 19.5 | 0.1 | 19.41 |