cfm

Vince Knight

Jan 08, 2024

CONTENTS:

The class notes are available here: https://vknight.org/cfm/

CHAPTER ONE

CLASS MEETINGS

1.1 Introduction week

1.1.1 First meeting

After this meeting students should:

- Know me a bit
- Understand how and why programming can be used to study mathematics.
- Know where all class resources are
- As a class agree on a schedule for office hours
- Know what they need to do to prepare for their first class

Introduce myself

Use this slide deck: `main`

- Pick 4 or 5 pictures of my family and I to introduce myself.
- Mathematician (where/what I studied)
- Trustee of the UK Python association https://uk.python.org
- Conference and workshop organiser
- Editor for JOSS
- Fellow of the sustainable software institute
- Maintainer of a number of open source scientific research packages

Programming

1. Class exercise: get a class word cloud association for the word "programming". (Using mentimeter.com).
2. Class exercise: how to make a cup of tea in 3 steps (using menti)
3. Class exercises: get a class word cloud association "What are uses for programming in Mathematics?"
    - Proving theorems;
    - Obtaining conjectures;
    - Implementing tools.

Location of class resources

- Show site: https://vknight.org/cfm/
- Show recordings
- Show learning central recordings
- Discuss discord

Office hours

As a class identify office hours.

Potential times are:

Tuesday: 1000-1200 Friday: 1100-1300 or Friday: 1300-1500

How the class works

Direct students at my teaching philosophy page: <https://vknight.org/tch-phi/>

Discuss the philosophy of the main text, specifically discussing this <https://vknight.org/pfm/overview/introduction/main.html#fig-knowledge-vs-technique>

Discuss structure of the book.

- Timetable:
  - Discuss the actual timetable.

  Discuss this diagram: `Week structure`

  Discuss this diagram: `Understanding`
- Assessment.
- For first lab session: work through chapter 1. Give a brief demo.

1.1.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class we discussed multiple things about the class itself:

- Course documents available here: https://vknight.org/cfm/
- We agreed that office hours will be: <>
- You can find my teaching philosophy here: https://vknight.org/tch-phi/

The goal of this week is to make sure you are able to install the software
necessary. Follow the instruction in this chapter of the course text:
https://vknight.org/pfm/tools-for-mathematics/01-using-notebooks/introduction/main.html

If you are unable to follow through the tutorial and/or the exercises please
```

(continues on next page)

(continued from previous page)

```
come by the optional drop in session on Wednesday.

Please get in touch if I can assist with anything,
Vince
```

1.1.3 Second meeting

- Give brief review of the contents of the chapter. Do this by working through the tutorial and the how to section of the relevant chapter of the class.
- Ask if anyone has any questions/comments/feedback.
- Remind everyone about the resources available to them (videos both on youtube and LC, chapters, plan, discord etc…).
- **Explain that the following things seemed to be the main ones to come up:**
    1. Finding files on computer.
    2. Difference between markdown and code cell.
    3. Using brackets when doing computations.
- Work through the exercises and solutions.

For each of those do a small walk-through showing the concepts again.

1.2 Algebra week

1.2.1 First meeting

After this meeting students should:

- Understand how to use the Sympy library to carry out basic Algebraic tasks.
- Know what they need to do to prepare for their second tutorial.

Problem

Explain to students that we will be solving the following problem:

1. Rationalise the following expression: $\frac{1}{\sqrt{3}+1}$.
2. Consider the quadratic $f(x) = -x^2 + 8x - 18$.
    1. Calculate the discriminant of the quadratic equation $f(x) = 0$. What does this tell us about the graph of $f(x)$.
    2. By completing the square, confirm that $(4, -2)$ is the maximum of point of $f(x)$.

Solution

Ask students to spend 5 minutes if they know/remember how to do this by hand. (This is just to get the students to think about it)

Now show how to get code to do this:

```
>>> import sympy
>>> expression = 1 / (sympy.sqrt(3) + 1)
>>> expression
1/(1 + sqrt(3))
>>> sympy.simplify(expression)
-1/2 + sqrt(3)/2
```

Discuss here how this differs if we used `math.sqrt`. Explain that `sympy.simplify` is essentially acting as a black box here.

Now to carry out the rest of the problem:

```
>>> x = sympy.Symbol("x")
>>> expression = - x ** 2 + 8 * x - 18
>>> expression
-x**2 + 8*x - 18
>>> sympy.discriminant(expression)
-8
```

Confirm results by hand.

Discuss what this implies:

- Quadratic equation has no real roots.
- Graph does not intersect the $y = 0$ line.
- Concave parabola (sign of leading coefficient of quadratic).

Confirm by solving the quadratic equation:

```
>>> equation = sympy.Eq(lhs=expression, rhs=0)
>>> equation
Eq(-x**2 + 8*x - 18, 0)

>>> sympy.solveset(expression, x)
FiniteSet(4 + sqrt(2)*I, 4 - sqrt(2)*I)
```

Now to move on to next part of the problem: completing the square:

```
>>> a, b, c = sympy.Symbol("a"), sympy.Symbol("b"), sympy.Symbol("c")
>>> completed_square = a * (x - b) ** 2 + c
>>> completed_square
a*(-b + x)**2 + c
```

Let us expand and compare the coefficients:

```
>>> sympy.expand(completed_square)
a*b**2 - 2*a*b*x + a*x**2 + c
```

We see that a is -1. Let us substitute this value in to the expression:

```
>>> completed_square.subs({a: -1})
c - (-b + x)**2
```

We can in fact overwrite the expression:

```
>>> completed_square = completed_square.subs({a: -1})
>>> completed_square
c - (-b + x)**2
```

If we now expand again and compare coefficients:

```
>>> sympy.expand(completed_square)
-b**2 + 2*b*x + c - x**2
```

We see that $2b = 8$. Despite the fact that this equation is relatively straightforward, let us solve it using `sympy`:

```
>>> equation = sympy.Eq(lhs=2 * b, rhs=8)
>>> sympy.solveset(equation, b)
FiniteSet(4)
```

We will substitute this value for b back in to the completed square, and expand again:

```
>>> completed_square = completed_square.subs({b: 4})
>>> completed_square
c - (x - 4)**2
>>> sympy.expand(completed_square)
c - x**2 + 8*x - 16
```

We see that $c - 16 = -18$. Let us again solve that equation using *sympy*:

```
>>> equation = sympy.Eq(lhs=c - 16, rhs= -18)
>>> sympy.solveset(equation, c)
FiniteSet(-2)
```

We will substitute this value back in:

```
>>> completed_square = completed_square.subs({c: -2})
>>> completed_square
-(x - 4)**2 - 2
>>> sympy.expand(completed_square)
-x**2 + 8*x - 18
```

Come back: with time take any questions.

Point at resources.

1.2.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over a demonstration of using Python to solve an
algebraic problem. I did the following mathematical techniques:

- Simplifying an exact numerical expression.
- Calculating the discriminant of a quadratic.
- Solving a symbolic equation.
- Substitute values in to a symbolic expression.

In preparation for your tutorial tomorrow please work through the second
chapter of the Python for mathematics book:
https://vknight.org/pfm/tools-for-mathematics/02-algebra/introduction/main.html

Please get in touch if I can assist with anything,
Vince
```

1.3 Calculus week

1.3.1 First meeting

After this meeting students should:

- Understand how to use the Sympy library to carry out basic Calculus tasks
- Know what they need to do to prepare for their third tutorial.

Problem

Explain to students that we will be solving the following problem:

Consider the function $f(x) = x^3 - ax^2 + bx - 5$

1. Given that $\frac{df}{dx}|_{x=0} = 0$, $\frac{d^2f}{dx^2}|_{x=0} = 5$ and that $b > 0$ find the values of a and b.
2. **For the specific values of a and b find:**
    1. $\lim_{x\to 0} f(x)$;
    2. $\lim_{x\to \infty} f(x)$;
    3. $\int f(x)dx$;
    4. $\int_5^{\pi} f(x)dx$.

Solution

Group exercise (breakout rooms of 3): ask students to spend 5 minutes writing a plan to tackle that problem (not necessarily carrying out each step).

Clearly write down these steps:

1. Differentiate and write system of equations and solve them for $a$ and b.
2. Take the limits and calculate the integrals.

Now show how to get code to do this:

```
>>> import sympy as sym  # Note that we are using an alias
>>> x = sym.Symbol("x")
>>> a = sym.Symbol("a")
>>> b = sym.Symbol("b")
>>> expression = x ** 3 - a * x ** 2 + b * x - 5
>>> expression
-a*x**2 + b*x + x**3 - 5
```

So far we have not done anything different to what we saw in the Algebra chapter. Now to differentiate:

```
>>> derivative = sym.diff(expression, x)
>>> derivative
-2*a*x + b + 3*x**2
```

This can now be used to give us our first equation:

```
>>> first_equation = sym.Eq(derivative.subs({x: 0}), 0)
>>> first_equation
Eq(b, 0)
```

Now let us get the second derivative:

```
>>> second_derivative = sym.diff(expression, x, 2)
>>> second_derivative
2*(-a + 3*x)
```

and get the second equation:

```
>>> second_equation = sym.Eq(second_derivative.subs({x: 0}), 5)
>>> second_equation
Eq(-2*a, 5)
```

Now to solve the first equation to obtain a value for b:

```
>>> sym.solveset(first_equation, b)
FiniteSet(0)
```

Now to substitute that value for $a$ (note that this is not necessary) and solve the second equation for $a$:

```
>>> second_equation = second_equation.subs({b: 0})
>>> second_equation
Eq(-2*a, 5)
```

We can solve this:

```
>>> sym.solveset(second_equation, a)
FiniteSet(-5/2)
```

Substituting these back:

```
>>> expression = expression.subs({b: 0, a: - sym.S(5) / 2})
>>> expression
x**3 + 5*x**2/2 - 5
```

We can confirm our findings:

```
>>> sym.diff(expression, x).subs({x: 0})
0
>>> sym.diff(expression, x, 2).subs({x: 0})
5
```

Now we can compute the limits and integrals:

```
>>> sym.limit(expression, x, 0)
-5
>>> sym.limit(expression, x, sym.oo)
oo
>>> sym.integrate(expression, x)
x**4/4 + 5*x**3/6 - 5*x
>>> sym.factor(sym.integrate(expression, (x, 5, sym.pi)))
(-5 + pi)*(3*pi**3 + 25*pi**2 + 125*pi + 565)/12
```

Come back: with time take any questions.

Point at resources.

1.3.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over a demonstration of using Python to solve a
calculus problem. I carried out the following mathematical techniques:

- Differentiation
- Limits
- Integrations

In preparation for your tutorial tomorrow please work through the third
chapter of the Python for mathematics book:
https://vknight.org/pfm/tools-for-mathematics/03-calculus/introduction/main.html

Please get in touch if I can assist with anything,
Vince
```

1.3.3 Second meeting

- Give brief review of the contents of the chapter. Do this by browsing through: https://vknight.org/pfm/tools-for-mathematics/03-calculus/how/main.html
- Mention that no specific query arose so I will do a coursework type question but also demonstrate some difficulties that might arise with the Kernel.

I will work through the following problem:

Consider the functions $f(x) = x^3 + 3x - 3$ and $g(x) = \cos(x) \sin(x)$.`

1. Create a variable *turning_points_of_f* which has value the turning points of $f(X)$.
2. Create a variable *turning_points_of_g* which has value the turning points of $g(X)$.
3. Create a variable *max_of_f_on_unit_circle* which has the maximum value of f for $x \in [0, 2\pi]$.
4. Create a variable *max_of_g_on_unit_circle* which has the maximum value of g for $x \in [0, 2\pi]$.
5. Which function has the maximum value?

The solution approach:

```
>>> import sympy as sym
>>> x = sym.Symbol("x")
>>> f = x ** 3 + 3 * x - 3
>>> g = sym.cos(x) * sym.sin(x)
>>> turning_points_of_f = sym.solveset(sym.diff(f, x), x)
>>> turning_points_of_f
FiniteSet(I, -I)
>>> turning_points_of_g = sym.solveset(sym.diff(g, x), x)
>>> turning_points_of_g
Union(ImageSet(Lambda(_n, _n*pi + pi/4), Integers), ImageSet(Lambda(_n, _n*pi + 3*pi/4),␣
↪Integers))
>>> max_of_f_on_unit_circle = max(f.subs({x: 0}), f.subs({x: 2 * sym.pi}))
>>> max_of_f_on_unit_circle
-3 + 6*pi + 8*pi**3
>>> max_of_g_on_unit_circle = max(g.subs({x: 5 * sym.pi / 4}), g.subs({x: 3 * sym.pi / 4}
↪), g.subs({x: 7 * sym.pi / 4}), g.subs({x: sym.pi / 4}))
>>> max_of_g_on_unit_circle
1/2
>>> float(max_of_f_on_unit_circle)
263.8997...
```

Highlight that the answer to the final question is thus f.

Note that *max* is not a function that has been specifically been seen before but that's not unexpected.

Now explain what the kernel is. Draw a picture showing the notebook separated from the terminal. Analogy of a brain.

Restart the kernel, show that the last command does not work.

Rerun the cells but include a mistake:

```
sym.solveset = (sym.diff(f, x), x)
```

Note the type of error we then get. And show that here we want the kernel to forget everything.

If there is time demonstrate plotting.

Discuss that sympy has some basic plotting but that I do not recommend it.

Point at matplotlib and numpy chapters and also specify that we will be using other things that are seen in future chapters (list comprehensions).:

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> domain = np.linspace(0, 2 * np.pi, 100)
>>> f_image = [f.subs({x: x_value}) for x_value in domain]
>>> g_image = [g.subs({x: x_value}) for x_value in domain]
>>> plt.figure()
>>> plt.plot(domain, f_image, label="$f(x)$")
>>> plt.plot(domain, g_image, label="$g(x)$")
>>> plt.legend()
```

1.4 Matrices week

1.4.1 First meeting

After this meeting students should:

- Understand how to use the Sympy library to carry out basic Matrix related tasks
- Know what they need to do to prepare for their fourth tutorial.

Problem

Explain to students that we will be solving the following problem:

The matrix A is given by $A = \begin{pmatrix}1 & 1 & a\\ 2 & a & 1\\ a & 1 & 2a\end{pmatrix}$.

1. Find the determinant of A
2. Hence find the values of a for which A is singular.
3. For the following values of a, when possible obtain A^{-1} and confirm the result by computing AA^{-1}:
    1. $a = 0$;
    2. $a = 1$;
    3. $a = 2$;
    4. $a = 3$.

Solution

Group exercise (breakout rooms of 3): ask students to spend 5 minutes writing a plan to tackle that problem (not necessarily carrying out each step). Some students might not be familiar with matrices so this is an opportunity

Now show how to get code to do this:

```
>>> import sympy as sym
>>> a = sym.Symbol("a")
>>> A = sym.Matrix([[1, 1, 3], [2, a, 1], [a, 1, 0]])
```

(continues on next page)

(continued from previous page)

```
>>> A
Matrix([
[1, 1, 3],
[2, a, 1],
[a, 1, 0]])
```

Now to calculate the determinant:

```
>>> determinant = A.det()
>>> determinant
-3*a**2 + a + 5
```

This can now be used to give us our first equation:

```
>>> sym.solveset(determinant, a)
FiniteSet(1/6 - sqrt(61)/6, 1/6 + sqrt(61)/6)
```

Indeed we can substitute those values in to the matrix and compute the inverse:

```
>>> A.subs({a: sym.S(1) / 6 - sym.sqrt(61) / 6})
Matrix([
[               1,                1, 3],
[               2, 1/6 - sqrt(61)/6, 1],
[1/6 - sqrt(61)/6,                1, 0]])
>>> A.subs({a: sym.S(1) / 6 - sym.sqrt(61) / 6}).inv()
Traceback (most recent call last):
...
sympy.matrices.common.NonInvertibleMatrixError: Matrix det == 0; not invertible.
```

Now for each value of a we can compute the inverse and confirm that inverse acts as required:

```
>>> A.subs({a: 0}).inv()
Matrix([
[-1/5,  3/5,  1/5],
[   0,    0,    1],
[ 2/5, -1/5, -2/5]])
>>> A.subs({a: 0}).inv() @ A.subs({a: 0})
Matrix([
[1, 0, 0],
[0, 1, 0],
[0, 0, 1]])
>>> A.subs({a: 1}).inv()
Matrix([
[-1/3,  1, -2/3],
[ 1/3, -1,  5/3],
[ 1/3,  0, -1/3]])
>>> A.subs({a: 1}).inv() @ A.subs({a: 1})
Matrix([
[1, 0, 0],
[0, 1, 0],
[0, 0, 1]])
>>> A.subs({a: 2}).inv()
Matrix([
```

(continues on next page)

(continued from previous page)

```
[ 1/5, -3/5,  1],
[-2/5,  6/5, -1],
[ 2/5, -1/5,  0]])
>>> A.subs({a: 2}).inv() @ A.subs({a: 2})
Matrix([
[1, 0, 0],
[0, 1, 0],
[0, 0, 1]])
>>> A.subs({a: 3}).inv()
Matrix([
[ 1/19, -3/19,  8/19],
[-3/19,  9/19, -5/19],
[ 7/19, -2/19, -1/19]])
>>> A.subs({a: 3}).inv() @ A.subs({a: 3})
Matrix([
[1, 0, 0],
[0, 1, 0],
[0, 0, 1]])
```

Come back: with time take any questions.

Point at resources.

1.4.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over a demonstration of using Python to solve a
matrix problem. I carried out the following mathematical techniques:

- Calculating a determinant
- Computing a matrix inverse
- Doing matrix multiplication

One thing I did not cover explicitly is solving a linear system. However
this is implicitely covered by calculating a matrix inverse. You can see
this here:
https://vknight.org/pfm/tools-for-mathematics/04-matrices/how/main.html#how-to-solve-a-
↪linear-system

In preparation for your tutorial tomorrow please work through the fourth
chapter of the Python for mathematics book:
https://vknight.org/pfm/tools-for-mathematics/04-matrices/introduction/main.html

Please get in touch if I can assist with anything,
Vince
```

1.4.3 Post meeting

Here is a video recording of a short review given in the 2020/2021 academic year: https://www.youtube.com/watch?v=rq_2ZYKq904

1.5 Combinatorics week

1.5.1 First meeting

After this meeting students should:

- Understand how to use the itertools library to solve combinatorial problems.
- Know what they need to do to prepare for their fifth tutorial.

Problem

Explain to students that we will be solving the following problem:

The digits 1, 2, 3 and 4 are arranged in random order, for form a four-digit number.

1. How many different four-digit numbers can be formed?
2. **How many different four-digit numbers:**
    1. Are even.
    2. Are less than 4000.

Solution

Group exercise (breakout rooms of 3): ask students to spend 5 minutes writing a plan to tackle that problem (not necessarily carrying out each step).

Now show how to get code to do this:

```
>>> import itertools
>>> digits = range(1, 5)
>>> digits
range(1, 5)
```

Explain what that is and how it can be useful to encapsulate the instructions for a set of objects as opposed to the objects themselves.:

```
>>> tuple(digits)
(1, 2, 3, 4)
```

Explain how we are mainly going to be using `itertools`:

```
>>> permutations = itertools.permutations(digits)
>>> permutations
<itertools.permutations object at 0x7fe267f1b5e0>
>>> permutations = tuple(itertools.permutations(digits))
>>> permutations
```

(continues on next page)

(continued from previous page)

```
((1, 2, 3, 4), (1, 2, 4, 3), (1, 3, 2, 4), (1, 3, 4, 2), (1, 4, 2, 3), (1, 4, 3, 2), (2,␣
↪1, 3, 4), (2, 1, 4, 3), (2, 3, 1, 4), (2, 3, 4, 1), (2, 4, 1, 3), (2, 4, 3, 1), (3, 1,␣
↪2, 4), (3, 1, 4, 2), (3, 2, 1, 4), (3, 2, 4, 1), (3, 4, 1, 2), (3, 4, 2, 1), (4, 1, 2,␣
↪3), (4, 1, 3, 2), (4, 2, 1, 3), (4, 2, 3, 1), (4, 3, 1, 2), (4, 3, 2, 1))
>>> len(permutations)
24
```

Note that we can confirm this theoretically:

```
>>> import math
>>> math.factorial(4)
24
```

For the next part of the question we want to consider our permutations and only count the ones that have an even last digit:

```
>>> sum((permutation[3] + 1) % 2 for permutation in permutations)
12
```

Be sure to explain:

- The indexing.
- The summation mod 2: $\sum_{\pi\in\Pi}\pi_4 + 1 \mod 2$

Note that this can be calculated theoretically. The last digit can either be 2 or 4. For each of those there are 3! possibilities for the other digits:

```
>>> 2 * math.factorial(3)
12
```

For the last question show that we will compute the following:

$$\sum_{\pi\in\Pi \text{ if } \pi_1 10^3+\pi_2 10^2+\pi_3 10+\pi_4\leq 4000} 1$$

Here is the code that corresponds to that:

```
>>> sum(
...     1
...     for permutation in permutations
...     if permutation[0] * 10 ** 3
...     + permutation[1] * 10 ** 2
...     + permutation[2] * 10
...     + permutation[3]
...     <= 4000
... )
18
```

Again confirm this theoretically. For a number to be less than 4000 it needs to start with either 1, 2 or 3. For each of those possibilities there are 3! permutations of the remaining digits:

```
>>> 3 * math.factorial(3)
18
```

Come back: with time take any questions.

Point at resources.

1.5.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over a demonstration of using Python to solve a
combinatorial problem. We used a computer to directly generate permutations
of objects.

- We mainly used the itertools library
- The range tool allowed us to access specific ranges of numbers.
- We saw how to sum over sets as a way of counting specific objects.

One thing I did not cover explicitly is using the itertools library to
generate combinations (where order does not matter).
You can see this here:
https://vknight.org/pfm/tools-for-mathematics/05-combinations-permutations/how/main.html
↪#creating-combinations-of-a-given-set-of-elements

We also did not cover how to compute directly binomial and permutation
coefficients which is also covered in the how to section of the book.

In preparation for your tutorial tomorrow please work through the fifth
chapter of the Python for mathematics book:
https://vknight.org/pfm/tools-for-mathematics/05-combinations-permutations/introduction/
↪main.html

Please get in touch if I can assist with anything,
Vince
```

1.5.3 Second meeting

We will cover two points of dicciulty:

- Understanding the summation notation
- Understanding the ordering of a combination specifically by revisiting the COMPUTER problem.

Consider:

$$\sum_{i=0}^{100} i^2$$

Discuss the images at https://github.com/drvinceknight/pfm/issues/120

Then discuss the COMPUTER solution.

Highlight what happens when we add another P to the end:

```
>>> letters = ("C", "O", "M", "P", "U", "T", "E", "R", "P")
>>> selections = tuple(itertools.combinations(letters, 3))
>>> sum(1 for selection in selections if selection == ("O", "P", "T"))
1
```

However:

```
>>> sum(1 for selection in selections if selection == ("O", "P", "T") or selection == ("O
↪", "T", "P"))
2
```

1.6 Probability week

1.6.1 First meeting

After this meeting students should:

- Understand how to use the random library to simulate random events
- Know what they need to do to prepare for their sixth tutorial.

Problem

Explain to students that we will be solving the following problem:

A delivery company delivers fragile items. If a delivery is on time it is usually because it was rushed. The probability that an item is delivered on time is 0.75. The probability that an item is broken given that it arrived on time is 0.3 and if it is late 0.2.

1. What is the probability that an item is late?
2. Given that an item is broken what is the probability that it was on time?

Solution

Group exercise (breakout rooms of 3): ask students to spend 5 minutes writing a plan to tackle that problem (not necessarily carrying out each step).

Now explain that we are going get a computer to simulate the described events from which we can measure the probabilities (and theoretically compare to the expected results).

Now show how to get code to first write a function to simulate a delivery:

```
>>> import random
>>> def is_delivery_late():
...     """
...     A function to randomly simulate if a delivery is late or not.
...     """
...     return random.random() > 0.75
```

Spend some time explaining what is happening here, including:

- The random library

- Functions (importance of white space, importance of the docstring, the help statement…)
- The return statement

Now we will use that function to create a number of experiments:

```
>>> number_of_repetitions = 10000
>>> samples = [is_delivery_late() for repetition in range(number_of_repetitions)]
>>> samples
[True, False, True, ..., False, True, False]
```

We can confirm the number of samples:

```
>>> len(samples)
10000
```

We can now confirm the probability:

```
>>> sum(sample for sample in samples) / number_of_repetitions
0.2459
```

Now explain that we will cover the entire question by writing a function to simulate both the delivery and whether or not the item is broken:

```
>>> def sample_experiment():
...     """
...     This samples a delivery and depending on whether or not it is late
...     selects whether or not the item is broken.
...     """
...     is_late = is_delivery_late()
...
...     if is_late is True:
...         probability_of_broken = 0.2
...     else:
...         probability_of_broken = 0.3
...
...     is_broken = random.random() < probability_of_broken
...     return is_late, is_broken
```

We can use this to confirm the previous result:

```
>>> samples = [sample_experiment() for repetition in range(number_of_repetitions)]
>>> sum(is_broken for is_late, is_broken in samples) / number_of_repetitions
0.2699
```

Now we can compute the probability of an item being on time if it is broken:

```
>>> samples_with_broken = [(is_late, is_broken) for is_late, is_broken in samples if is_
↪broken is True]
>>> sum(is_late for is_late, is_broken in samples_with_broken) / len(samples_with_broken)
0.18114406
```

Note that we can use Bayes' theorem to confirm this theoretically:

$$P(\text{On time}|\text{Broken}) = \frac{P(\text{Broken}|\text{On time})P(\text{On time})}{P(\text{Broken})}$$

We have:

$$P(\text{Broken}|\text{On time}) = 0.3$$

$$P(\text{On time}) = 0.75$$

$$P(\text{Broken}) = 0.3 \times 0.75 + 0.2 \times 0.25$$

We can compute this:

```
>>> probability_of_on_time_if_broken = 0.3 * 0.75 / (0.3 * 0.75 + 0.2 * 0.25)
>>> probability_of_on_time_if_broken
0.818181...
```

Thus the probability for the question is:

```
>>> 1 - probability_of_on_time_if_broken
0.181818...
```

Come back: with time take any questions.

Point at resources.

1.6.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over a demonstration of using Python to solve a
probabilitistic problem. I demontrated how to simulate random events and
measure probabilities directly. We did this using the following Python
tools:

- Writing functions.
- List comprehensions.

In preparation for your tutorial tomorrow please work through the sixth
chapter of the Python for mathematics book:
https://vknight.org/pfm/tools-for-mathematics/06-probability/introduction/main.html

Please get in touch if I can assist with anything,
Vince
```

1.6.3 Post meeting

Here is a video recording of a short review given in the 2020/2021 academic year: https://www.youtube.com/watch?v=u-ii1TeLHrM

1.7 Sequences week

1.7.1 First meeting

After this meeting students should:

- Understand how to use define function recursively
- Know what they need to do to prepare for their seventh tutorial.

Problem

Explain to students that we will be solving the following problem:

A sequence $a_1, a_2, a_3, \dots$ is defined by:

$$\begin{cases} a_1 = k, \\ a_{n+1} = 3a_n \breve{}4, n \geq 1, \end{cases}$$

where k is a constant.

1. Write down an expression for a_2 in terms of k.
2. Show that $a_3 = 9k - 16$
3. Given that $\sum_{r=1}^{12} a_r = 50$ find the value of k.

Solution

Group exercise (breakout rooms of 3): ask students to spend 5 minutes writing a plan to tackle that problem (not necessarily carrying out each step).

Now explain that we essentially need to add to our tools the ability to define a function recursively which we can do:

```
>>> def generate_a(k_value, n):
...     """
...     Uses recursion to return a_n for a given value of k:
...     a_1 = k
...     a_n = 3a_n - 4
...     """
...     if n == 1:
...         return k_value
...     return 3 * generate_a(k_value, n - 1) - 4
```

Spend some time explaining what is happening here. Specifically highlighting that the programming recursion matches the mathematical definition.

Show how we can make some calls of the function:

```
>>> k = 5
>>> generate_a(k_value=k, n=1)
5
>>> generate_a(k_value=k, n=2)
11
>>> generate_a(k_value=k, n=3)
29
>>> generate_a(k_value=k, n=4)
83
>>> generate_a(k_value=k, n=5)
245
```

To be able to answer the questions we need k as a symbol:

```
>>> import sympy as sym
>>> k = sym.Symbol("k")
>>> generate_a(k_value=k, n=2)
3*k - 4
```

To find a_3:

```
>>> generate_a(k_value=k, n=3)
9*k - 16
```

We now compute the sum of the first 12 terms:

```
>>> sum_of_first_12_terms = sum(generate_a(k_value=k, n=r) for r in range(1, 13))
>>> sum_of_first_12_terms
265720*k - 531416
```

Our equation is thus:

```
>>> equation = sym.Eq(sum_of_first_12_terms, 50)
>>> equation
Eq(265720*k - 531416, 50)
>>> sym.solveset(equation, k)
FiniteSet(20441/10220)
```

Come back: with time take any questions.

Point at resources.

1.7.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over a demonstration of using recursion in Python to be
able to define sequences recursively.

```

(continues on next page)

(continued from previous page)

```
In preparation for your tutorial tomorrow please work through the seventh
chapter of the Python for mathematics book:
https://vknight.org/pfm/tools-for-mathematics/07-sequences/introduction/main.html

Please get in touch if I can assist with anything,
Vince
```

1.8 Example coursework

1.8.1 First meeting

After this meeting students should:

- Understand how the individual coursework will look.

In class

Download and show students this notebook `assignment.ipynb`

Could be beneficial to show students how to download and open it (from the this page of notes). This is a task that is often difficult. Point students at: https://vknight.org/cfm/#how-to-download-the-coursework and here is a video showing how to do it as well: https://www.youtube.com/watch?v=LpDtq589P8w

Work through the example coursework.

1.8.2 Second meeting

Ask students to download the mock (from the class site) and offer to demo how to do it.

Then take any questions.

1.8.3 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over the example coursework.

In preparation for your tutorial please feel free to work through the mock
coursework or revise any topics seen so far.

Please get in touch if I can assist with anything,
Vince
```

1.9 Variables, conditionals and loops

1.9.1 Pre class email

Send the following email before class:

```
Hi all,

I hope you are all well.

This is a brief email to make you aware of a few things:

- The class timetable and weekly schedule for this Semester is available
  here: https://vknight.org/cfm/
- This week we have class at 1600 (via Zoom) and we
  will be covering this chapter
  https://vknight.org/pfm/building-tools/01-variables-conditionals-loops/introduction/
↪main.html.
- This Semester your assessment is a group exercise. I am letting you self
  select groups (groups of 4) but will assist anyone unable to find a group.
  We will discuss this more today in class.

I am about to send you your draft individual coursework feedback.
**Important** This will come from `vincent.knight@gmail.com` and the subject
will be `DO NOTE REPLY - Coursework mark and feedback`. Note that I consider
this first feedback I send you to be a *draft*. As it is automatically marked
it is not unlikely that a mistake has crept in. Feel free to get in touch
with me if anything is unclear in the feedback.

Please get in touch if I can assist with anything,
Vince
```

1.9.2 First meeting

After this meeting students should:

- Understand the coursework exercise and the group aspect
- Understand the how to use a for loop and a while loop
- Know what they need to do to prepare for their seventh tutorial and how to make groups.

Coursework

Explain goal of coursework:

> To build and present a Python library to solve a mathematical problem of your
choice.

How it will be assessed (15 minute presentation and 2 page (max) paper).

Groups of 4. Self select and see instructions in email.

Agree on deadline for formation of group.

I recommend you meet regularly.

Have an agenda for every meeting.

Single point of contact.

Problem

Explain to students that we will be solving the following problem:

A perfect number is a positive integer whose divisors (excluding itself) sum to itself. For example, 6 is a perfect number because $6 = 1 + 2 + 3$.

1. How many perfect numbers are there less than 20?
2. What is the next perfect number?

Solution

Group exercise (breakout rooms of 3): ask students to spend 5 minutes writing a plan to tackle that problem (not necessarily carrying out each step).

We start by writing a function that checks if a number is perfect:

```
>>> def is_perfect(n):
...     """
...     Checks if a number n is perfect.
...     """
...     return sum(i for i in range(1, n) if (n % i == 0)) == n
>>> is_perfect(5)
False
>>> is_perfect(6)
True
```

Now we can check all numbers up until 20:

```
>>> checks = [is_perfect(n) for n in range(1, 21)]
```

Note that we can combine all booleans using `all`:

```
>>> all(checks)
False
```

Or `any`:

```
>>> any(checks)
True
```

We cam also count them (as before) using `sum`:

```
>>> sum(checks)
1
```

We are there using a list comprehension (as before) **but** we can also use a *for* loop:

```
>>> count = 0
>>> for n in range(1, 21):
...     if is_perfect(n):
...         count += 1
>>> count
1
```

Discuss how the *for* loop works.

Now to answer the other question, **we do not know** how many times we need to iterate the code so we use a *while* loop:

```
>>> n = 7
>>> while not is_perfect(n):
...     n += 1
>>> n
28
```

Indeed the prime divisors of 28 are:

```
>>> import sympy as sym
>>> sym.primefactors(28)
[2, 7]
```

And:

```
>>> 1 + 2 + 4 + 7 + 14
28
```

We can repeat the above to find the 3rd perfect number:

```
>>> n = 29
>>> while not is_perfect(n):
...     n += 1
>>> n
496
```

Indeed the divisors of 496 are:

```
>>> import sympy as sym
>>> sym.primefactors(496)
[2, 31]
```

And:

```
>>> 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248
496
```

1.9.3 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over 2 separate things:

1. The group coursework (please read the end of this email where action is
   required).
2. Variables, loops and conditionals.

I recommend working through the following chapter of the class text:
https://vknight.org/pfm/building-tools/01-variables-conditionals-loops/introduction/main.
↪html

If you have any questions I will be in 2.35 on Wednesday from 1000 until
1200 for the first 4 weeks of the course.

IMPORTANT ACTION REQUIRED

For your group coursework you have until <DEADLINE> to form groups with 4
people. I am letting you self select groups. If you do not have a group by
<DEADLINE> I will create a group for you.

Once you have created a group, 1 member of your group must email me (CC'ing
in all other members) with subject: "Group formed"

In the email please tell me the name of your group (you can be imaginative)
and also which member of your group is the point of contact.

Once you have created a group and started working, please use https://forms.office.com/e/
↪qnVdB1KUG7
to record weekly progress.

---

Please get in touch if I can assist with anything,
Vince
```

1.9.4 Post meeting

The main difficulties raised were:

- Sets
- Modulo
- Doing more involved problems

Create a set:

```
>>> set_1 = set((-1, 1, 1, 2, 3, 5))
>>> set_1
{1, 2, 3, 5, -1}
>>> set_2 = set(range(10))
>>> set_2
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Union:

```
>>> set_1 | set_2
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -1}
```

Intersection:

```
>>> set_1 & set_2
{1, 2, 3, 5}
```

Set subtraction:

```
>>> set_1 - set_2
{-1}
```

Set inclusion:

```
>>> set_1 <= set_2
False
```

Reasons for using sets:

- The ability to to the above operations.
- Immediate removal of duplicates.
- Checking inclusion.

Run the following timing examples:

```
N = 10 ** 4
numbers_as_range = range(N)
numbers_as_list = list(numbers_as_range)
numbers_as_tuple = tuple(numbers_as_range)
numbers_as_set = set(numbers_as_range)
```

Testing the list:

```
%%timeit
5000 in numbers_as_list
```

Testing the tuple:

```
%%timeit
5000 in numbers_as_tuple
```

Testing the set:

```
%%timeit
5000 in numbers_as_set
```

Testing the range itself:

```
%%timeit
5000 in numbers_as_range
```

Give some examples of modula arithmetic:

```
>>> 5 % 2
1
>>> 17 % 9
8
>>> for i in range(24 * 5):
...     print(i % 24)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

(continues on next page)

(continued from previous page)

```
17
18
19
20
21
22
23
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

(continues on next page)

(continued from previous page)

```
21
22
23
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Discuss how important it is, when building tools to spend time away from the computer. Make the analogy of writing down a recipe before starting to cook.

1.10 Functions and data structures

1.10.1 First meeting

After this meeting students should:

- Have an understanding of functions.
- Have an understanding of lists.

Problem

Explain to students that we will be solving the following problem:

Verify the following identity for all integer values of $0 \leq a \leq 100$, $0 \leq b \leq 100$ and $1 \leq n \leq 10$:

$$(a+b)^n = \sum_{i=0}^{n}\binom{n}{i}a^ib^{n-i}$$

Solution

Ask students to discuss how they would do this.

Now show:

```
>>> def get_lhs(a, b, n):
...     """
...     Compute (a + b) ^ n directly
...     """
...     return (a + b) ** n
>>> get_lhs(a=5, b=10, n=5)
759375
>>> import scipy.special
>>> def get_rhs(a, b, n):
...     """
...     Compute the right hand side of the identity which aims to calculate
...     a summation of powers a^ib^(n - i) with a binomial coefficient.
...     """
...     return sum(scipy.special.binom(n, i) * a ** i * b ** (n - i) for i in range(n +␣
↪1))
>>> get_rhs(a=5, b=10, n=5)
759375.0
```

Ask if this confirms the identity?

Now create the following function to check the identify:

```
>>> def check_identity(a, b, n):
...     """
...     Check the relationship
...     """
...     return get_lhs(a=a, b=b, n=n) == get_rhs(a=a, b=b, n=n)
```

Now create all checks:

```
>>> checks = [
...    check_identity(a=a_val, b=b_val, n=n_val)
...    for a_val in range(101) for b_val in range(101) for n_val in range(1, 11)
... ]
```

Note that we can check all of them:

```
>>> all(checks)
False
```

Ask why this is the case. Have a discussion leading to the fix:

```
>>> def get_rhs(a, b, n):
...     """
...     Compute the right hand side of the identity which aims to calculate
...     a summation of powers a^ib^(n - i) with a binomial coefficient.
...     """
...     return sum(int(scipy.special.binom(n, i)) * a ** i * b ** (n - i) for i in␣
↪range(n + 1))
```

Re define `check_identity`:

```
>>> def check_identity(a, b, n):
...     """
...     Check the relationship
...     """
...     return get_lhs(a=a, b=b, n=n) == get_rhs(a=a, b=b, n=n)
```

Now create all checks:

```
>>> checks = [
...    check_identity(a=a_val, b=b_val, n=n_val)
...    for a_val in range(101) for b_val in range(101) for n_val in range(1, 11)
... ]
```

Note that we can check all of them:

```
>>> all(checks)
True
```

Come back: with time take any questions.

Point at resources.

1.10.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I we used functions and list comprehensions. The chapter we
are covering this week also includes sections on dictionaries and sets.

In preparation for your tutorial please work through the following chapter:
https://vknight.org/pfm/building-tools/02-functions-and-data-structures/introduction/
↪main.html

Please get in touch if I can assist with anything,
Vince
```

1.11 Object oriented programming

1.11.1 First meeting

After this meeting students should:

- Understand the motivation for using objects
- Understand the basic syntax of how to create classes and instances
- Know what they need to do to prepare for their tutorial

Problem

Tell students we are going to investigate some group theory.

A group G is defined as:

- Closure: if $a, b \in G$ then $a \cdot b \in G$.
- Identity: there exists $e \in G: \ e \cdot a = a \cdot e = a$ for all $a \in G$
- Inverse: for all $a \in G$ there exists $a^{-1} \in G: \ a \cdot a^{-1} = e$
- Associativity: for all $a, b, c \in G: \ (a \cdot b) \cdot c = a \cdot (b \cdot c)$

Consider the permutation group: https://en.wikipedia.org/wiki/Permutation_group. On two elements: $\{0, 1\}$ there are two permutations:

- $\sigma_{01}(0) = 0$ and $\sigma_{01}(1) = 1$ (identity)
- $\sigma_{10}(0) = 1$ and $\sigma_{10}(1) = 0$ (flip)

Obtain the multiplication table for this group.

Solution

$$\begin{pmatrix} 01 & 10 \\ 10 & 01 \end{pmatrix}$$

Explain that we can create an **abstract** object in Python that allows us to manipulate these elements in the same way that we can manipulate integers and/or floats. Or in the same way that `<insert popular video game>` manipulates characters.

Here is how we create a very basic class that allows us to create an element corresponding to a given σ:

```
>>> class Permutation():
...     """A class that corresponds to an element of a permutation group"""
...     def __init__(self, sigma):
...         """When creating an instance set attributes."""
...         self.sigma = sigma
...         self.N = len(sigma)
...     def permute(self, vector):
...         """Given a vector of integers permute them."""
...         return [self.sigma[i] for i in vector]
```

We can use this to create specific elements of the Permutation group:

```
>>> pi_01 = Permutation([0, 1])
>>> pi_10 = Permutation([1, 0])
>>> pi_01.sigma, pi_01.N
([0, 1], 2)
>>> pi_10.sigma, pi_10.N
([1, 0], 2)
```

The `permute` method allows us to permute a given vector: for example what does σ_{10} do to $(0, 1)$:

```
>>> pi_10.permute([0, 1])
[1, 0]
```

We now have all we need to define the group operation on the permutation class:

```
>>> class Permutation():
...     """A class that corresponds to an element of a permutation group"""
...     def __init__(self, sigma):
...         """When creating an instance set attributes."""
...         self.sigma = sigma
...         self.N = len(sigma)
...     def permute(self, vector):
...         """Given a vector of integers permute them."""
...         return [self.sigma[i] for i in vector]
...     def operate(self, other):
...         """Define the group operation on self and other"""
...         return Permutation(self.permute(other.permute(range(self.N))))
```

Redefining our new instances:

```
>>> pi_01 = Permutation([0, 1])
>>> pi_10 = Permutation([1, 0])
>>> pi_10.operate(pi_01)
<__main__.Permutation ...>
```

We see that a new instance of the *Permutation* class has been produced (which is expected) but we cannot really tell what it is. Let us implement another magic method to do so:

```
>>> class Permutation():
...     """A class that corresponds to an element of a permutation group"""
...     def __init__(self, sigma):
...         """When creating an instance set attributes."""
...         self.sigma = sigma
...         self.N = len(sigma)
...     def permute(self, vector):
...         """Given a vector of integers permute them."""
...         return [self.sigma[i] for i in vector]
...     def __repr__(self):
...         return str(self.sigma)
...     def operate(self, other):
...         """Define the group operation on self and other"""
...         return Permutation(self.permute(other.permute(range(self.N))))
```

Redefining our new instances:

```
>>> pi_01 = Permutation([0, 1])
>>> pi_10 = Permutation([1, 0])
>>> pi_10.operate(pi_01)
[1, 0]
```

We see that when σ_{01} operates on *sigma_{10}* we get *sigma_{10}* back. A nice way to be able to check this using Python's == operator is to include a new special method:

```
>>> class Permutation():
...     """A class that corresponds to an element of a permutation group"""
...     def __init__(self, sigma):
...         """When creating an instance set attributes."""
...         self.sigma = sigma
```

(continues on next page)

(continued from previous page)

```
...         self.N = len(sigma)
...     def permute(self, vector):
...         """Given a vector of integers permute them."""
...         return [self.sigma[i] for i in vector]
...     def __repr__(self):
...         return str(self.sigma)
...     def __eq__(self, other):
...         return self.sigma == other.sigma
...     def operate(self, other):
...         """Define the group operation on self and other"""
...         return Permutation(self.permute(other.permute(range(self.N))))
```

Let us confirm this now:

```
>>> pi_01 = Permutation([0, 1])
>>> pi_10 = Permutation([1, 0])
>>> pi_10.operate(pi_01) == pi_10
True
```

One final change we're going to make is replace the `operate` method to use a magic python method:

```
>>> class Permutation():
...     """A class that corresponds to an element of a permutation group"""
...     def __init__(self, sigma):
...         """When creating an instance set attributes."""
...         self.sigma = sigma
...         self.N = len(sigma)
...     def permute(self, vector):
...         """Given a vector of integers permute them."""
...         return [self.sigma[i] for i in vector]
...     def __repr__(self):
...         return str(self.sigma)
...     def __eq__(self, other):
...         return self.sigma == other.sigma
...     def __mul__(self, other):
...         """Define the group operation on self and other"""
...         return Permutation(self.permute(other.permute(range(self.N))))
```

We can now use the * operator:

```
>>> pi_01 = Permutation([0, 1])
>>> pi_10 = Permutation([1, 0])
>>> pi_10 * pi_01 == pi_10
True
```

Ask student to write code that uses this class to obtain the multiplication table for our group:

```
>>> def display_multiplication_table(elements):
...     for first in elements:
...         products = []
...         for second in elements:
...             products.append(first * second)
...         print(products)
```

We can now use this:

```
>>> permutations = [pi_01, pi_10]
>>> display_multiplication_table(elements=permutations)
[[0, 1], [1, 0]]
[[1, 0], [0, 1]]
```

Let us modify this to look at permutations of size $N = 3$. Explain that we will make use of a very handy Python library for creating permutations of things:

```
>>> import itertools
>>> N = 3
>>> permutations = [Permutation(list(perm)) for perm in itertools.permutations(range(N))]
>>> permutations
[[0, 1, 2], [0, 2, 1], [1, 0, 2], [1, 2, 0], [2, 0, 1], [2, 1, 0]]
```

Let us take a look at the multiplication table:

```
>>> display_multiplication_table(elements=permutations)
[[0, 1, 2], [0, 2, 1], [1, 0, 2], [1, 2, 0], [2, 0, 1], [2, 1, 0]]
[[0, 2, 1], [0, 1, 2], [2, 0, 1], [2, 1, 0], [1, 0, 2], [1, 2, 0]]
[[1, 0, 2], [1, 2, 0], [0, 1, 2], [0, 2, 1], [2, 1, 0], [2, 0, 1]]
[[1, 2, 0], [1, 0, 2], [2, 1, 0], [2, 0, 1], [0, 1, 2], [0, 2, 1]]
[[2, 0, 1], [2, 1, 0], [0, 2, 1], [0, 1, 2], [1, 2, 0], [1, 0, 2]]
[[2, 1, 0], [2, 0, 1], [1, 2, 0], [1, 0, 2], [0, 2, 1], [0, 1, 2]]
```

Can students see the various properties closure, associativity, inverse and identity?

If there is any remaining time, invite students to write code that checks these conditions.

Walk and discuss with them.

Closure:

```
>>> def test_closure(elements):
...     return all(first * second in elements
...                for first in elements
...                for second in elements)
>>> test_closure(elements=permutations)
True
```

Identity:

```
>>> def test_specific_identity_element(elements, identity):
...     return all(first * identity == first for first in elements)
>>> def test_identity_element(elements):
...     return any(test_specific_identity_element(elements=elements, identity=identity)
...                for identity in permutations)
>>> test_identity_element(elements=permutations)
True
```

Inverse:

```
>>> def test_inverse_element_for_given_identity(elements, identity):
...     has_inverse = []
...     for first in elements:
```

(continues on next page)

(continued from previous page)

```
...         products = []
...         for second in elements:
...             products.append(first * second)
...         has_inverse.append(identity in products)
...     return all(has_inverse)
>>> def test_inverse_element(elements):
...     return any(test_inverse_element_for_given_identity(elements=permutations,␣
↪identity=identity)
...                for identity in elements)
>>> test_inverse_element(elements=permutations)
True
```

Associativity:

```
>>> def test_associativity(elements):
...     return all((first * second) * third == first * (second * third)
...                for first, second, third in itertools.product(elements, repeat=3))
>>> test_associativity(elements=permutations)
True
```

These can all be brought together:

```
>>> def test_group(elements):
...     return (test_closure(elements=elements) and
...             test_identity_element(elements=elements) and
...             test_inverse_element(elements=elements) and
...             test_associativity(elements=elements))
>>> test_group(elements=permutations)
True
```

Not all subsets of a group are a group:

```
>>> test_group(elements=permutations[:-1])
False
```

We can also use this to check for larger group sizes:

```
>>> N = 4
>>> permutations = [Permutation(list(perm)) for perm in itertools.permutations(range(N))]
>>> test_group(elements=permutations)
True
>>> N = 5  # This takes a little while
>>> permutations = [Permutation(list(perm)) for perm in itertools.permutations(range(N))]
>>> test_group(elements=permutations)
True
```

1.11.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over 1 main thing: Object Oriented Programming.

In preparation for your tutorial tomorrow please work through the tenth
chapter of the Python for mathematics book:
https://vknight.org/pfm/building-tools/03-objects/introduction/main.html

Please get in touch if I can assist with anything,
Vince
```

1.12 Using the cli and an editor

1.12.1 First meeting

After this meeting students should:

- Understand the basic idea of using a cli and an editor

Explain that there is no specific class problem today as there is no new python concept to cover.

- Open a terminal an macOS and explain the use of Anaconda Prompt on Windows.
- Navigate to *cfm* describing the difference in commands when appropriate on Windows.
- Open a Python REPL.

Now open VScode (show how to install), create a file and put it in the folder. Run it.

1.12.2 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over using the command line and an editor.

In preparation for your tutorial tomorrow please work through the tenth
chapter of the Python for mathematics book:
https://vknight.org/pfm/building-tools/04-editor-and-cli/introduction/main.html

A reminder that tomorrow is your last tutorial.

Please get in touch if I can assist with anything,
Vince
```

1.13 Modularisation

1.13.1 First meeting

After this meeting students should:

- Understand the idea of modularising code itself.
- Understand how to import a file from a file.

Show students the following code and ask them to go in breakout rooms and discuss what it does.:

```
>>> def run_2_opt_algorithm(
...     number_of_stops,
...     distance_matrix,
...     iterations,
...     seed=None,
... ):
...
...     internal_stops = list(range(1, number_of_stops))
...     if seed is not None:
...         np.random.seed(seed)
...         np.random.shuffle(internal_stops)
...     tour = [0] + internal_stops + [0]
...     best_cost = sum(
...         distance_matrix[current_stop, next_stop]
...         for current_stop, next_stop in
...         zip(tour[:-1], tour[1:])
...         )
...     for _ in range(iterations):
...
...         two_indices = np.random.choice(range(1, number_of_stops), 2)
...         i, j = sorted(two_indices)
...
...         candidate_tour = tour[:i] + tour[i:j + 1][::-1] + tour[j + 1:]
...
...         candidate_cost = sum(
...             distance_matrix[current_stop, next_stop]
...             for current_stop, next_stop in
...             zip(candidate_tour[:-1], candidate_tour[1:])
...             )
...
...         if (candidate_cost) <= best_cost:
...             best_cost = candidate_cost
...             tour = candidate_tour
...
...     return tour
```

1.13.2 The mathematical problem considered

Ask students what the travelling sales agent problem is.

Have a discussion about it.

Consider the following 25 stops with coordinates as follows:

```
>>> import numpy as np
>>> x = np.array([13, 16, 22,  1,  4, 28,  4,  8, 10, 20, 22, 19,  5, 24,  7, 25, 25, 13,
↪ 27,  2,  7,  8, 24, 15, 25])
>>> y = np.array([18,  6, 26, 14,  9, 10, 21, 20, 17, 20,  6, 16, 16,  1, 19,  4, 25, 18,
↪ 20, 20, 20, 15,  8,  1,  2])
```

We can visualise this:

```
>>> import matplotlib.pyplot as plt
>>> plt.scatter(x, y)
<matplotlib...
```

A python library called *sklearn* has a lot of functionality that can be used to look at data, for example we can get the distances here:

```
>>> import sklearn.metrics.pairwise
>>> distance_matrix = sklearn.metrics.pairwise.euclidean_distances(tuple(zip(x, y)))
>>> distance_matrix
array([[ 0.        , 12.36931688, ...
```

Show how the problem is indeed solved (copy the code above and put it in a notebook).

We can use the above code to find a tour:

```
>>> tour = run_2_opt_algorithm(number_of_stops=25, distance_matrix=distance_matrix,␣
↪iterations=500, seed=0)
>>> tour
[0, 17, 14, 20, 7, 8, 21, 12, 6, 19, 3, 4, 11, 23, 1, 10, 13, 15, 24, 22, 5, 18, 16, 2,␣
↪9, 0]
```

We want to plot this tour so we need to recover the coordinates:

```
>>> def plot_tour(x, y, tour):
...     ordered_x = x[tour]
...     ordered_y = y[tour]
...     plt.figure()
...     plt.scatter(x, y)
...     plt.plot(ordered_x, ordered_y)
>>> plot_tour(x=x, y=y, tour=tour)
```

This is great but it's code that works and it is not straightforward to see how or why it works. Let us fix that:

```
>>> def get_tour(number_of_stops, seed=None):
...     internal_stops = list(range(1, number_of_stops))
...     if seed is not None:
...         np.random.seed(seed)
...         np.random.shuffle(internal_stops)
...     return [0] + internal_stops + [0]
```

(continues on next page)

(continued from previous page)

```
>>> def swap_cities(tour, steps):
...     i, j = sorted(steps)
...     new_tour = tour[:i] + tour[i:j + 1][::-1] + tour[j + 1:]
...     return new_tour
>>> def get_cost(tour, distance_matrix):
...     return sum(
...         distance_matrix[current_stop, next_stop]
...         for current_stop, next_stop in
...         zip(tour[:-1], tour[1:])
...     )
```

We can use this to for example get the cost of our tour:

```
>>> get_cost(tour=tour, distance_matrix=distance_matrix)
133.40828432465426
```

Show how the code is much cleaner now:

```
>>> def run_2_opt_algorithm(
...     number_of_stops,
...     distance_matrix,
...     iterations,
...     filename=None,
...     seed=None,
... ):
...     tour = get_tour(number_of_stops=number_of_stops, seed=seed)
...     best_cost = get_cost(tour=tour, distance_matrix=distance_matrix)
...     for _ in range(iterations):
...         two_indices = np.random.choice(range(1, number_of_stops), 2)
...         candidate_tour = swap_cities(tour=tour, steps=two_indices)
...         if (cost:=get_cost(tour=candidate_tour, distance_matrix=distance_matrix)) <=␣
↪best_cost:
...             best_cost = cost
...             tour = candidate_tour
...     return tour
>>> tour = run_2_opt_algorithm(number_of_stops=25, distance_matrix=distance_matrix,␣
↪iterations=500, seed=0)
>>> tour
[0, 17, 14, 20, 7, 8, 21, 12, 6, 19, 3, 4, 11, 23, 1, 10, 13, 15, 24, 22, 5, 18, 16, 2,␣
↪9, 0]
```

Discuss the need for docstrings.

Then put the code in `tsp.py` and show how it can be imported.

1.13.3 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over modularising code which is an important foundation
of software development.

In class we used an example of solving the travelling salesagent problem and
you can find a different example (studying snakes and ladders) here:
https://vknight.org/pfm/building-tools/05-modularisation/tutorial/main.html

Please get in touch if I can assist with anything,
Vince
```

1.14 Documentation

1.14.1 First meeting

After this meeting students should:

- Understand the general structure required for documentation
- Understand how to write further markdown

1.14.2 Writing documentation

Discuss how we will write documentation for the *tsp* library we wrote in the first section.

There are 4 components for documentation:

1. Tutorial
2. How to guide
3. Reference
4. Discussion

Ask students what they think the purpose of each of these are. Ask them to discuss amongst themselves.

Say that we will address this after writing the documentation.

Writing the tutorial

Open a markdown file (in the same location as the `tsp.py` file) and call it `README.md`.

Write the following:

~~~
# TSP

A library for solving instances of the travelling sales agent problem

## Tutorial

In this tutorial we will see how to use `tsp` to solve instances of the
[Travelling Salesman Problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem)

Assuming we have the following distance matrix:

```python
import numpy as np

distance_matrix = np.array(((0, 5, 2, 9), (5, 0, 3, 1), (2, 3, 0, 7), (9, 1, 7, 0)))
```

We can obtain a tour using the following:

```python
import tsp

tour = tsp.run_2_opt_algorithm(number_of_stops=4, distance_matrix=distance_matrix,␣
↪iterations=1000, seed=0)
```

We can see the tour here:

```python
tour
```

This gives:

```
[0, 3, 1, 2, 0]
```

The `tsp` library includes further functionality which you can read in the
How To guides.

## How to guides

### How to obtain a basic tour

To obtain a basic tour, we use the `tsp.get_tour` function:

```python
~~~

(continues on next page)

(continued from previous page)

~~~
import tsp

tsp.get_tour(number_of_stops=5)
```

This gives

```
[0, 1, 2, 3, 4, 0]
```

If we pass a random seed this will also shuffle the interior stops (in a
reproducible manner):

```python
tsp.get_tour(number_of_stops=5, seed=0)
```

This gives:

```
[0, 3, 4, 2, 1, 0]
```

### How to swap two spots

To swap two cities for a given tour, we use the `tsp.swap_cities` function:

```python
tour = [0, 1, 2, 3, 4, 5, 0]
tsp_swap_cities(tour=tour, indices=(2, 4))
```

This gives:

```
[0, 1, 4, 3, 2, 5, 0]
```

### How to get the cost of a tour

To calculate the cost of a given tour, we use the `tsp.get_cost` function:

```python
distance_matrix = np.array(((0, 5, 2, 9), (5, 0, 3, 1), (2, 3, 0, 7), (9, 1, 7, 0)))
tour = [0, 1, 2, 3, 0]
tsp.get_cost(tour=tour, distance_matrix=distance_matrix)
```

which gives:

```
~~~

(continues on next page)

(continued from previous page)

~~~
24
```

### How to plot a tour

To plot a tour we use the `tsp.plot_tour` function:

```python
xs = (0, 1, 1, 2.5)
ys = (0, 5, 1, 3)
tour = [0, 1, 3, 2, 0]
tsp.plot_tour(x=xs, y=ys, tour=tour)
```

This gives the following image:

![](./how-to.svg)

### How to use the 2-opt algorithm

To run the full algorithm, we use the
`tsp.run_2_opt_algorithm` function:

```python
distance_matrix = np.array(((0, 5, 2, 9), (5, 0, 3, 1), (2, 3, 0, 7), (9, 1, 7, 0)))
tour = tsp.run_2_opt_algorithm(number_of_stops=4, distance_matrix=distance_matrix,␣
↪iterations=1000, seed=0)
tour
```

This gives:

```
[0, 3, 1, 2, 0]
```

## Explanations

This software implements the 2-opt algorithm for the travelling sales agent
problem.

### The TSP

As an example, if we consider three cities with the following matrix
defining their distances between them:

$$
    \begin{pmatrix}
        0 & 4 & 1\\
        4 & 0 & 2\\
        9 & 2 & 0
    \end{pmatrix}
~~~

(continues on next page)

(continued from previous page)

```
$$

Note that the distance matrix is not symmetric, it is a lot further to go
from the 3rd to the 1st city (9) than to go from the 1st to the 3rd (1)

If a tour starts **and** finishes at the first city there are in fact 2
possibilities:

$$T \in \{(0, 1, 2, 0), (0, 2, 1, 0)\}$$

The cost $c(t)$ for $t\in T$ of these tour tours is taken to be the total
distance travelled:

1. For $t=(0, 1, 2, 0)$ we have $c(t)=4 + 2 + 9=15$
2. For $t=(0, 2, 1, 0)$ we have $c(t)=1 + 2 + 2=5$

As the size of our problem grows the complexity of finding the optimal tour
grows in complexity. In fact this problem is NP-hard, which puts it in a
class of problems for which a general solution cannot be obtained
efficiently for any given sized problem.

## The 2-opt algorithm

One solution approach of this is the 2-opt algorithm which is what is
implemented in this software.

The 2-opt algorithm is an example of a neighbourhood search algorithm which
means that it iteratively improves a given solution by looking in
at other candidates near it.

The 2-opt algorithm does this by randomly choosing two stops in a tour, and
swapping the order between them. Essentially picking stop $n$ and stop $n +
k$ and reversing the order. Thus the new candidate would visit the same
stops as the original tour, until it got to stop $n$, when it would instead
go to stop $n + k$ and reverse its way back to stop $n$.

Once this candidate tour is obtained the cost is evaluated and if it is good
it is accepted as the new solution.

This has the effect of essentially untangling a given tour.

## Reference

### List of functionality

This software implements 5 functions:

1. `plot_tour`
2. `get_tour`
3. `swap_cities`
4. `get_cost`
5. `run_2_opt_algorithm`
```

(continues on next page)

(continued from previous page)

```
### Bibliography

The wikipedia page on the TSP offers good background reading:
https://en.wikipedia.org/wiki/Travelling_salesman_problem

The following text is a recommended text on neighbourhood search algorithms:

> Aarts, Emile, Emile HL Aarts, and Jan Karel Lenstra, eds. Local search in
> combinatorial optimization. Princeton University Press, 2003.
```

1.14.3 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over documenting code.

In class we documented the travelling salesagent problem code
you can find a different example (studying snakes and ladders) here:
https://vknight.org/pfm/building-tools/06-documentation/tutorial/main.html

Please get in touch if I can assist with anything,
Vince
```

1.15 Testing

1.15.1 First meeting

After this meeting students should:

- Understand how assertion statements work
- Understand how to write unit tests using assert statements
- Understand how to write tests for documentation

1.15.2 Testing software

Discuss how we will write automated tests for the *tsp* library we wrote in the first section.

Writing the unit tests

Write the following in a *test_tsp.py* file:

```python
import numpy as np
import matplotlib.pyplot as plt

import tsp

def test_get_tour_with_no_seed():
    number_of_stops = 4
    tour = tsp.get_tour(number_of_stops=number_of_stops)
    assert np.array_equal(tour, np.array([0, 1, 2, 3, 0]))

def test_get_tour_with_seed_0():
    number_of_stops = 4
    seed = 0
    tour = tsp.get_tour(number_of_stops=number_of_stops, seed=seed)
    assert np.array_equal(tour, np.array([0, 3, 2, 1, 0])), f"Obtained output is {tour}"

def test_get_tour_with_seed_1():
    number_of_stops = 4
    seed = 1
    tour = tsp.get_tour(number_of_stops=number_of_stops, seed=seed)
    assert np.array_equal(tour, np.array([0, 1, 3, 2, 0])), f"Obtained output is {tour}"


def test_swap_cities():
    tour = [0, 1, 3, 2, 4, 0]
    steps = (4, 1)
    new_tour = tsp.swap_cities(tour=tour, steps=steps)
    assert new_tour == [0, 4, 2, 3, 1, 0], f"Obtained output is {new_tour}"


def test_get_cost():
    distance_matrix = np.array(((0, 5, 2, 9), (5, 0, 3, 1), (2, 3, 0, 7), (9, 1, 7, 0)))
    tour = [0, 1, 2, 3, 0]
    cost = tsp.get_cost(tour=tour, distance_matrix=distance_matrix)
    assert cost == 24


def test_run_2_opt_algorithm_with_seed_0():
    distance_matrix = np.array(((0, 5, 2, 9), (5, 0, 3, 1), (2, 3, 0, 7), (9, 1, 7, 0)))
    number_of_stops = 4
    seed = 0
    iterations = 50
    tour = tsp.run_2_opt_algorithm(number_of_stops=number_of_stops,
            distance_matrix=distance_matrix, seed=seed, iterations=iterations)
    assert tour == [0, 2, 3, 1, 0], f"Obtained output is {tour}"

def test_run_2_opt_algorithm_with_seed_1():
    distance_matrix = np.array(((0, 5, 2, 9), (5, 0, 3, 1), (2, 3, 0, 7), (9, 1, 7, 0)))
    number_of_stops = 4
```

(continues on next page)

(continued from previous page)

```
    seed = 1
    iterations = 50
    tour = tsp.run_2_opt_algorithm(number_of_stops=number_of_stops,
            distance_matrix=distance_matrix, seed=seed, iterations=iterations)
    assert tour == [0, 2, 1, 3, 0], f"Obtained output is {tour}"

def test_plot_tour():
    x = np.array([2, 3])
    y = np.array([2, 3])
    tour = np.array([0, 1, 0])
    plot = tsp.plot_tour(x=x, y=y, tour=tour)
    assert plot is None

test_get_tour_with_no_seed()
test_get_tour_with_seed_0()
test_get_tour_with_seed_1()
test_swap_cities()
test_get_cost()
test_run_2_opt_algorithm_with_seed_0()
test_run_2_opt_algorithm_with_seed_1()
test_plot_tour()
```

Run the above tests by running:

```
$ python test_tsp.py
```

Depending on time now discuss refactoring some of the code. For example:

- Modify *swap_cities* to be called *swap_steps* as this name is more generic.
- Possibly modify *swap_cities/swap_steps* so that it can handle numpy array outputs.

At each step modify the source code and then modify the tests.

Discuss importance of tests as a debugging tool.

Testing the documentation

Modify the documentation in the *README.md* file to use doctests:

~~~
# TSP

A library for solving instances of the travelling sales agent problem

## Tutorial

In this tutorial we will see how to use `tsp` to solve instances of the
[Travelling Salesman Problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem)

Assuming we have the following distance matrix:

```python
>>> import numpy as np
>>> distance_matrix = np.array(((0, 5, 2, 9), (5, 0, 3, 1), (2, 3, 0, 7), (9, 1, 7, 0)))
~~~

(continues on next page)

(continued from previous page)

~~~
```

We can obtain a tour using the following:

```python
>>> import tsp  # doctest: +SKIP
>>> tour = tsp.run_2_opt_algorithm(number_of_stops=4, distance_matrix=distance_matrix,␣
↪iterations=1000, seed=0)  # doctest: +SKIP
>>> tour  # doctest: +SKIP
[0, 3, 1, 2, 0]

```

The `tsp` library includes further functionality which you can read in the
How To guides.
~~~

Note that the doctests are skipped when checked for this documentation.

Run the doc tests by typing:

```
$ python -m doctest README.md
```

Discussion around 3 pillars of software development

Specifically look at diagram in further information section: https://vknight.org/pfm/building-tools/07-testing/why/main.html#how-are-modularisation-documentation-and-testing-related

1.15.3 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over automated testing.

In class we wrote unit and doctests for the Travelling Salesagent Problem.
You can find a different example (studying snakes and ladders) here:
https://vknight.org/pfm/building-tools/07-testing/tutorial/main.html

Please get in touch if I can assist with anything,
Vince
```

1.16 Writing

1.16.1 First meeting

After this meeting students should:

- Understand how to use LaTeX
- Have access to guidance on writing

1.16.2 In class discussion

Discuss how they already have seen an example a markup language: LaTeX.

In fact: they have already seen some LaTeX.

Open up a local editor (do not feel obliged to use VS Code) and show to a LaTeX document can be compiled.

Then show that this is how they are expected to write their paper.

Specifically type the following:

```latex
\documentclass[a4paper]{article}

% Set up page size
\usepackage[margin=1.5cm, includefoot, footskip=30pt]{geometry}

% Nice way to display code
\usepackage{minted}

% Import images

\title{The TSP Package}
\author{Vince Knight}
\date{}


\begin{document}
\maketitle

\section{Introduction:The TSP}

\end{document}
```

Open overleaf and show how this can be used to write LaTeX.

Show them the following two guidelines for writing:

- https://vknight.org/cfm/assets/pdf/writing-mathematics-guidelines/main.pdf
- https://vknight.org/cfm/assets/pdf/a-guide-to-writing-mathematics/main.pdf

Use breakout rooms/groups and ask students to spend 5/10 minutes finding something specifically useful in there.

After breakout rooms ask all students to write down those things in a shared medium (for example the class discord).

1.16.3 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I went over automated writing mathematics. We discuss this
from a technical point of view (how to use LaTeX) and I refer you to my
notes on that: https://vknight.org/tex/

We also discussed guidelines for better writing:

- https://vknight.org/cfm/assets/pdf/writing-mathematics-guidelines/main.pdf
- https://vknight.org/cfm/assets/pdf/a-guide-to-writing-mathematics/main.pdf

On Friday the mathematics subject librarian will take the class to discuss
finding good sources for your paper (which is specifically a point in the
marking criteria).

Please get in touch if I can assist with anything,
Vince
```

1.17 Presenting

1.17.1 Before first meeting

Send the following email before class:

```
Hi all,

In tomorrow's class I will give an example 15 minute presentation.

After the presentation and any question you might have we will discuss the
following principles of presetionat: https://vknight.org/pop/

Note that as part of my presentation I will give you another example of an
entire project (the other examples having been discussed in class and in the
notes).

Please get in touch if I can assist with anything,
Vince
```

1.17.2 In class meeting

Give a presentation.

Specifically present *ertai* version *0.0.2* (a library specifically built for this purpose).

After the presentation, if there are any questions about the tpic answer them.

After that, give an overview of the principles of presentation: https://vknight.org/pop/

Use breakout rooms of about 4 students to ask them to discuss the presentation and the principles.

Bring back for a group discussion.

1.17.3 After class email

Send the following email after class:

```
Hi all,

A recording of today's class is available at <>.

In this class I gave you a presentation of a similar format to what you will
be doing during your coursework assessment.

We also discussed principles of presentation: https://vknight.org/pop/

You can find the paper that corresponds to the presentation here:
https://vknight.org/cfm/assets/pdf/ertai_paper/main.pdf (the tex source code is
available here: https://vknight.org/cfm/assets/pdf/ertai_paper/main.tex).

Please get in touch if I can assist with anything,
Vince
```

1.18 Marking criteria

The following marking criteria should be sent to the students once all groups have been formed:

```
# 2 Page Paper

- **Summary**: Has a clear description of the high-level functionality and
  purpose of the software for a diverse, non-specialist audience been
  provided?  [10%]
- **A statement of need**: Do the authors clearly state what problems the
  software is designed to solve and who the target audience is?
- **State of the field**: Do the authors describe how this software compares
  to other commonly-used packages? [10%]
- **Quality of writing**: Is the paper well written (i.e., it does not
  require editing for structure, language, or writing quality)? [10%]
- **References**: Is the list of references complete, and is everything
  cited appropriately that should be cited (e.g., papers, datasets,
  software)? Do references in the text use the proper citation syntax? [10%]
```

(continues on next page)

(continued from previous page)

```
# 15 Presentation

- **Functionality**: Have the functional claims of the software been
  confirmed? [10%]
- **Documentation**: Does the documentation have a Tutorial, How to section,
  Reference and Explanation section? Is it clear? Is the source code clear?
  [10%]
- **Modularity**: Is the code written in a modular way? [10%]
- **Testing**: Is all code tested? [10%]
- **Presentation**: Was the presentation format used appropriately? Were the
  visual aids appropriate? [10%]


Note that this marking criteria has some overlap with the review criteria
for the Journal of Open Source Software
<https://joss.readthedocs.io/en/latest/review_checklist.html>. Some examples
of papers written for that journal that can be helpful are:

- Matching: A Python library for solving matching games <https://joss.theoj.org/papers/
↪10.21105/joss.02169>
- Nashpy: A Python library for the computation of Nash equilibria <https://joss.theoj.
↪org/papers/10.21105/joss.00904>

Examples of presentations are available at: https://vknight.org/pop/
```

1.19 Matplotlib

Ask students to consider the 4 data sets:

```
set_1_x = (10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0)
set_1_y = (8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68)
set_2_x = (10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0)
set_2_y = (9.14, 8.14, 8.74, 8.77, 9.26, 8.1, 6.13, 3.1, 9.13, 7.26, 4.74)
set_3_x = (10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0)
set_3_y = (7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73)
set_4_x = (8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 19.0, 8.0, 8.0, 8.0)
set_4_y = (6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.5, 5.56, 7.91, 6.89)
```

Then have a discussions:

1. Are the 4 datasets the same?
2. Why?
3. How would we show that mathematically?

Say that we're going to plot them and then follow the steps from the matplotlib tutorial in pfm.

CHAPTER

TWO

OTHER

2.1 List of coursework topics

Exercise	Q1	Q2	Q3	Q4
Example	Calculus	Calculus	Comb.	Seq.
Mock	Arithmetic	Algebra	Matrices	Prob.
2020/2021	Prob.	Comb.	Calculus	Seq.
2020/2021R	Calculus	Comb.	Seq.	Matrices
2021/2022	Matrices	Seq.	Matrices	Calculus
2022/2023	Seq.	Prob.	Algebra	Matrices