

Computational Thinking

## Discrete Mathematics

Number Theory

Topic 00 : Module Introduction

Logic

Lecture 01 : Module Overview

Dr Kieran Murphy   

Computing and Mathematics, SETU (Waterford).  
(kieran.murphy@setu.ie)

Graphs and  
Networks

Autumn Semester, 2024

Collections

### Outline

- Motivation and aim of this module.
- Administration trivia — Contact hours, Assessment structure, ...
- Resources

Enumeration

Relations & Functions

# Outline

---

1. Module Introduction	2
1.1. <b>What</b> is the Aim of this Module?	3
1.2. <b>Why</b> Study Discrete Mathematics?	12
1.3. <b>How</b> will the Module be Delivered? Assessed?	14
1.4. <b>Who</b> is delivering this module?	18
1.5. <b>When</b> will the module be delivered?	19
2. Resources	20
2.1. Moodle	21
2.2. Github	22
2.3. Slack	23
2.4. Colab	24
2.5. PyTutor	25
3. Final Comments	27

# What?

(Aim)

## Aim, as per the Module Descriptor ...

This module provides a solid foundation of selected topics in discrete mathematics related to computing and information sciences. The topics are covered in an elementary manner in order to reinforce understanding of concepts and improving algebraic problem-solving skills so that the student can effectively proceed with their study of a degree programme in computing.

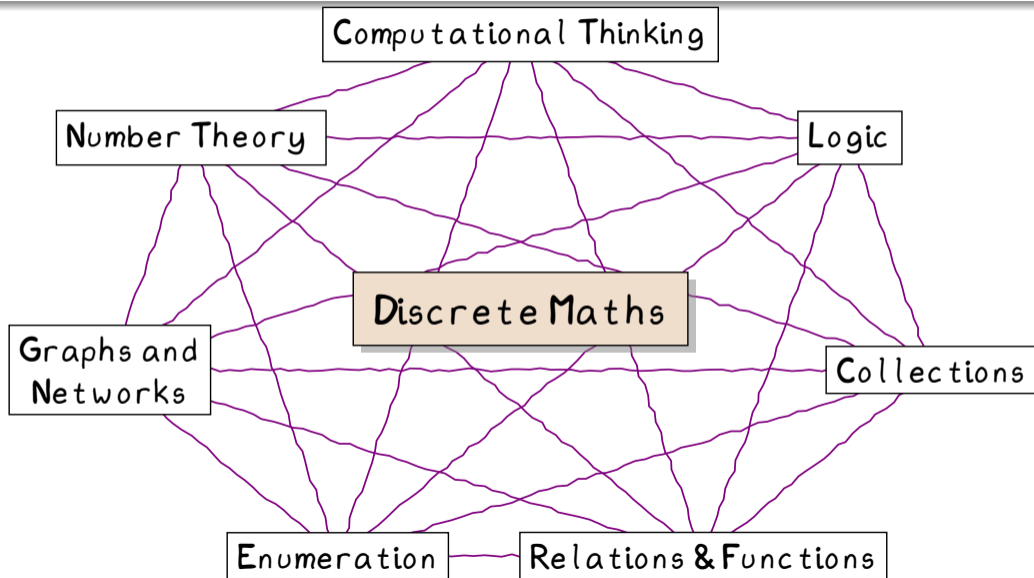
## Translation (Informal Aims)

- 1 Reason logically — aim for precision and correctness over speed.
- 2 Develop and manipulate theoretical models  
— a **set** is a collection of things, a **relation** is a collection of pairs of things, a **graph** is a collection of things with pairwise connections, etc..
- 3 Translate

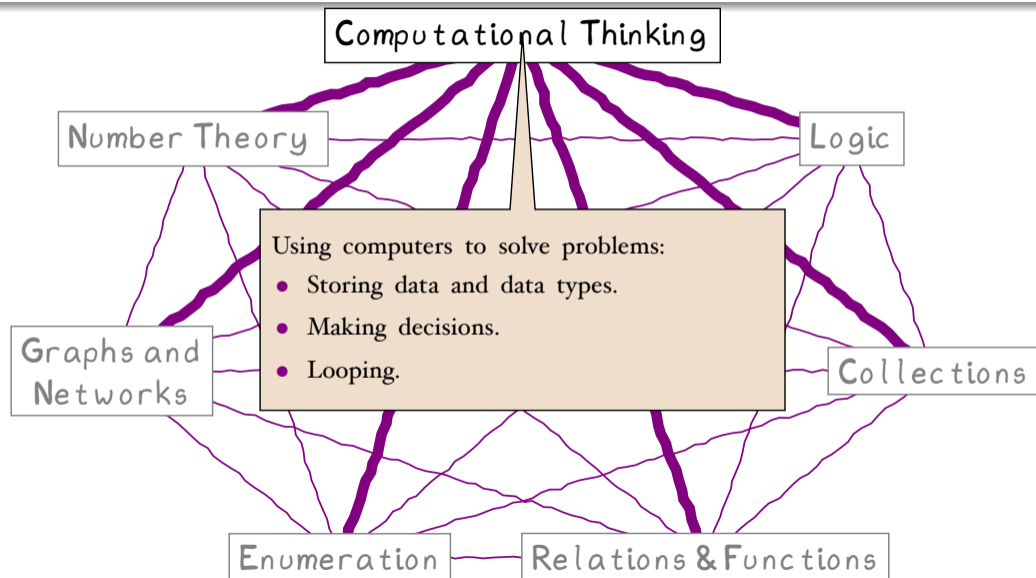
computing concepts/implementation (Python) ↔ theoretical models (mathematics).

## What?

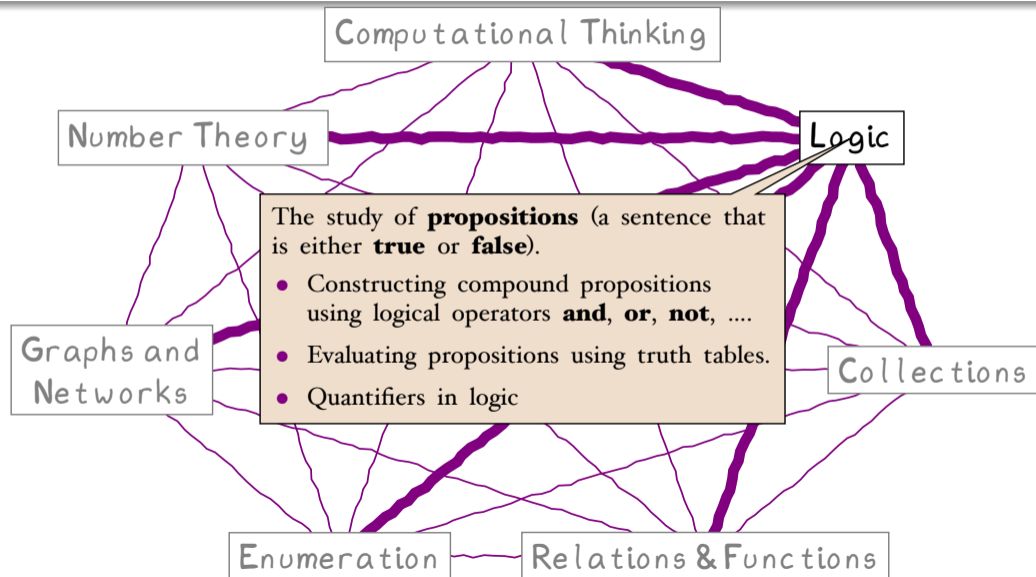
## (Discrete Mathematics)



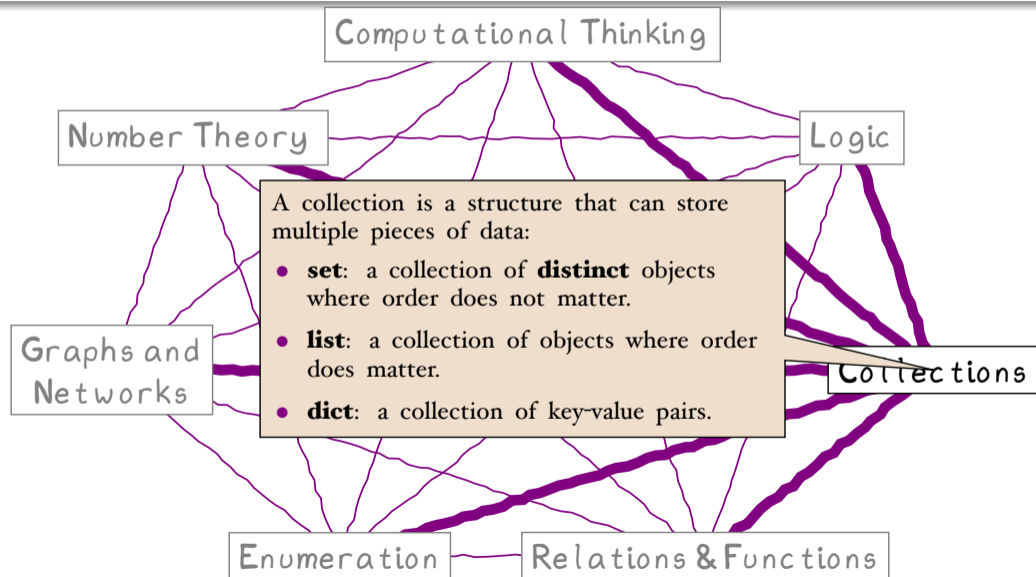
# What?



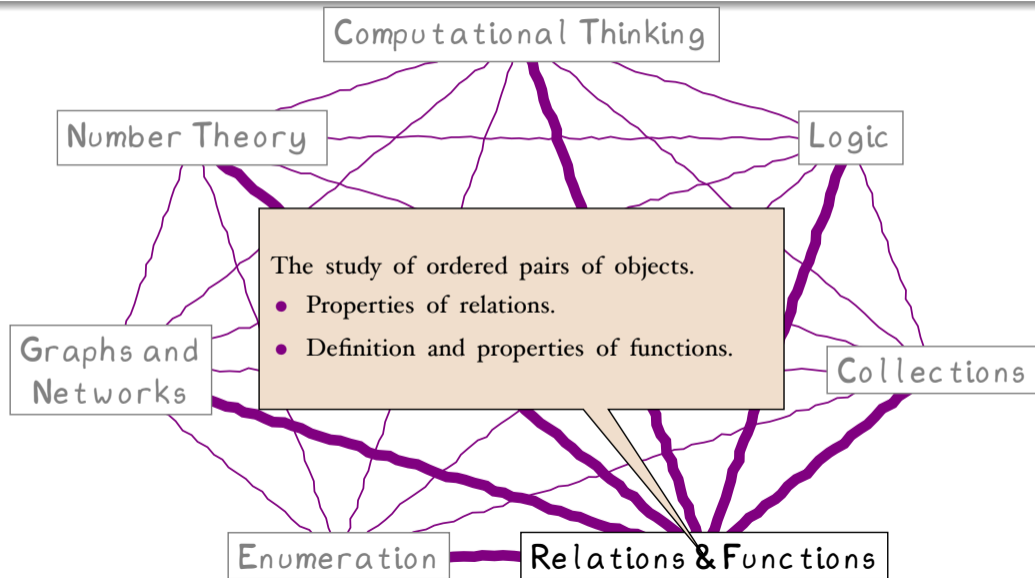
# What?



# What?

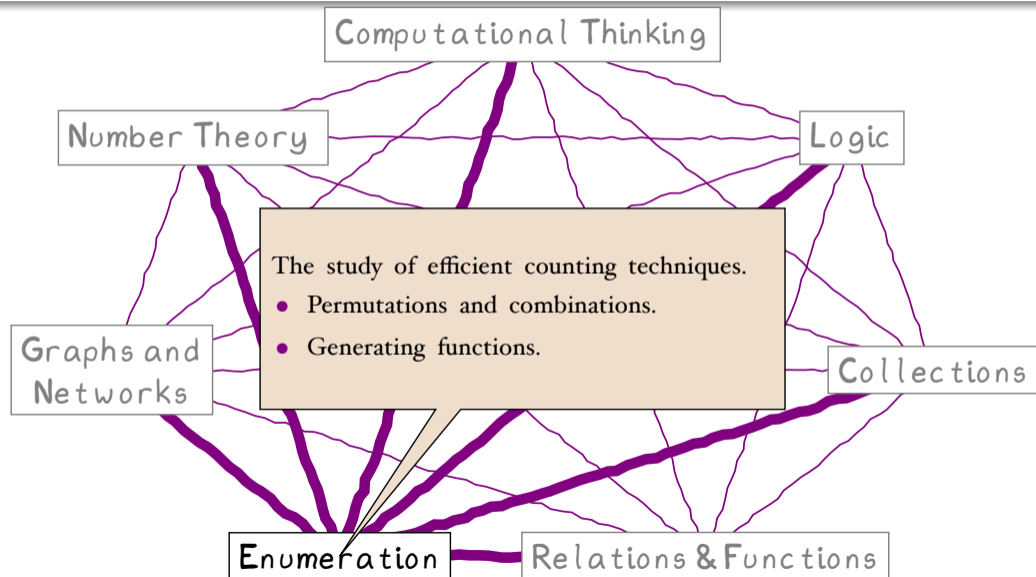


# What?

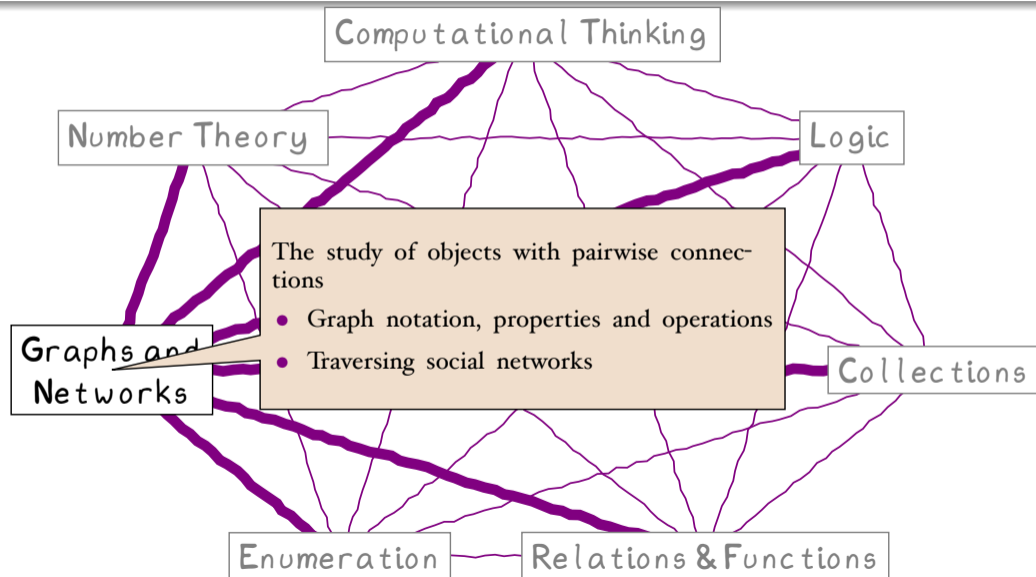




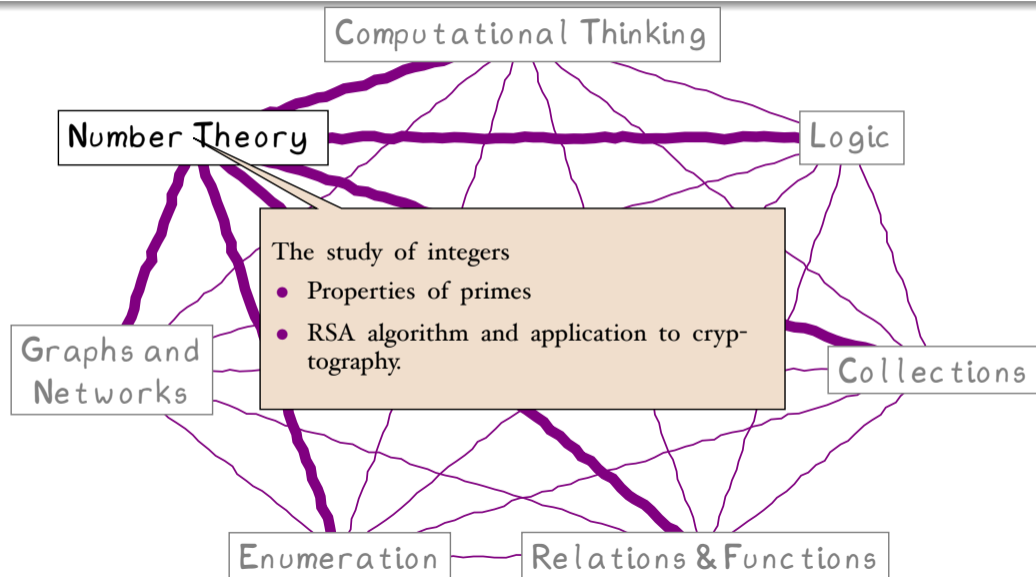
# What?



# What?



# What?



# Why?

Many, many reasons ... pickinig one\* ...

Machine learning is the future of computing

Discrete mathematics is the core of machine learning.

# Why?

Many, many reasons ... pickinig one\* ...

Machine learning is the future of computing

Discrete mathematics is the core of machine learning.

# Why?

Many, many reasons ... pickinig one\* ...

Machine learning is the future of computing

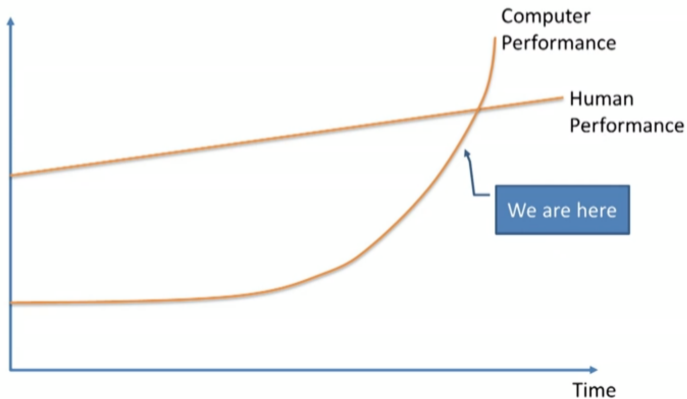
Discrete mathematics is the core of machine learning.

# Why?

Many, many reasons ... pickinig one\* ...

Machine learning is the future of computing

Discrete mathematics is the core of machine learning.



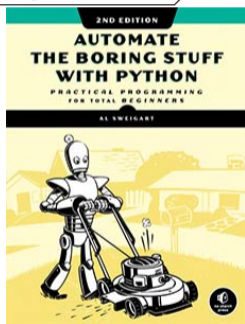
# Why?

(Aim)

Computers are faster and more accurate\* than us, they don't get board/distracted†

⇒ Get computers to do the hard (and/or boring) stuff

See Al Sweigart's books



Or our view on life



Why have a dog and bark?

\*Exception to this are the generative text models (chatGPT\*, Bard, ...) where accuracy is replaced by "most likely/probable".

†Well, if you don't count Teslas, in FSD mode, driving over pedestrians ...



# How?

(Contact Hours)

- ⚙️ Three lectures per week
  - ⚙️ Cover concepts, definitions, examples, etc.
  - ⚙️ BUT feel free to stop me and ask questions at any point.
  - ⚠️ **You should print out notes in advance of lecture, or have access to them during class.**
  - ⚙️ Ideally you skimmed over the notes in advance of lecture.
  - ⚠️ **Take notes during lectures.**
- ⚙️ One tutorial per week
  - ⚙️ Review of exercises based on the material covered in the lecturers.
  - ⚠️ **You need to have printout of tutorial sheets in advance of lecture.**
  - ⚙️ Ideally you have attempted/completed some/all questions in advance of tutorial and you are just attending the tutorials to show off.
  - ⚙️ Online quiz for self review at end of each topic.
- ⚙️ One practical per week
  - ⚙️ Using Python (via *colab notebooks*) to demonstrate implementation details of discrete mathematics concepts.
  - ⚙️ Introduce programming in Python — never have too much programming.
  - ⚠️ **You need to upload notebook by end of week (Saturday 11:00pm).**

# How?

(Contact Hours)

## ⚙️ Three lectures per week

- ⚙️ Cover concepts, definitions, examples, etc.

- ⚙️ BUT feel free to stop me and ask questions at any point.

- ⚠️ **You should print out notes in advance of lecture, or have access to them during class.**

- ⚙️ Ideally you skimmed over the notes in advance of lecture.

- ⚠️ **Take notes during lectures.**

## ⚙️ One tutorial per week

- ⚙️ Review of exercises based on the material covered in the lectures.

- ⚠️ **You need to have printout of tutorial sheets in advance of lecture.**

- ⚙️ Ideally you have attempted/completed some/all questions in advance of tutorial and you are just attending the tutorials to show off.

- ⚙️ Online quiz for self review at end of each topic.

## ⚙️ One practical per week

- ⚙️ Using Python (via *colab notebooks*) to demonstrate implementation details of discrete mathematics concepts.

- ⚙️ Introduce programming in Python — never have too much programming.

- ⚠️ **You need to upload notebook by end of week (Saturday 11:00pm).**

# How?

(Contact Hours)

- ⚙️ Three lectures per week
  - ⚙️ Cover concepts, definitions, examples, etc.
  - ⚙️ BUT feel free to stop me and ask questions at any point.
  - ⚠️ **You should print out notes in advance of lecture, or have access to them during class.**
  - ⚙️ Ideally you skimmed over the notes in advance of lecture.
  - ⚠️ **Take notes during lectures.**
- ⚙️ One tutorial per week
  - ⚙️ Review of exercises based on the material covered in the lecturers.
  - ⚠️ **You need to have printout of tutorial sheets in advance of lecture.**
  - ⚙️ Ideally you have attempted/completed some/all questions in advance of tutorial and you are just attending the tutorials to show off.
  - ⚙️ Online quiz for self review at end of each topic.
- ⚙️ One practical per week
  - ⚙️ Using Python (via *colab notebooks*) to demonstrate implementation details of discrete mathematics concepts.
  - ⚙️ Introduce programming in Python — never have too much programming.
  - ⚠️ **You need to upload notebook by end of week (Saturday 11:00pm).**

# How?

# (Assessment Structure)

## 75% End of Semester Exam

Current plan (this is subject to change so ask about this in week 10!)

- 4 questions (typically 3–5 parts per question. Answer all questions (i.e. no choice).
- Tend not to have question per topic.
- 80% same material as last year — see pre-Covid exam papers, but there may still be some differences in format/style of questions as will the relative emphasis/weighting of the different topics.

## 25% Continuous Assessment

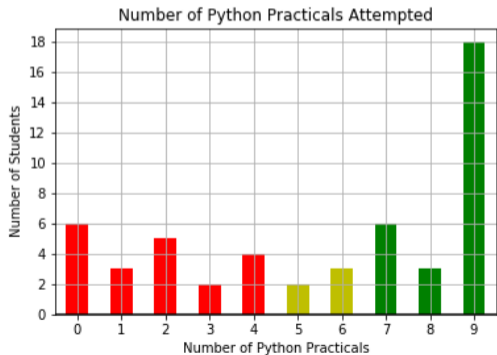
- Practical work based on 10 python practicals and 2 online class tests<sup>‡</sup>.
- In theory<sup>§</sup>, weekly assignments, are graded in advance of next week.

---

<sup>§</sup>In practice, I ~~may~~ will fall behind a bit.

## A Brief Look at 2019/20 Results

- 52 students enrolled, but only 28 passed!  $\Rightarrow$  pass rate of 53.8%.
- Of the 32 students who attempted at least 5 practicals, 25 passed  $\Rightarrow$  pass rate of 78.1%.
- Of the 27 students who attempted at least 7 practicals, 24 passed  $\Rightarrow$  pass rate of 88.8%.



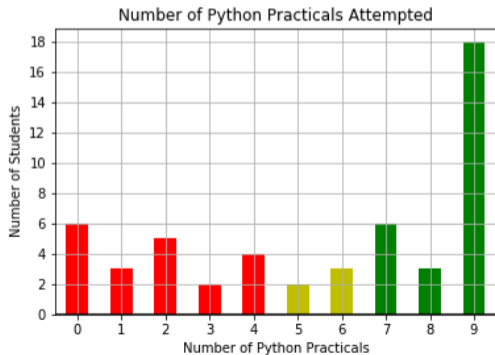
Keep up with the material:

- Read notes before & after lectures
- Attend practicals and upload practical work on time.
- Attempt tutorial questions.

Note "Attempted"  $\neq$  "Completed correctly"

## A Brief Look at 2019/20 Results

- 52 students enrolled, but only 28 passed!  $\Rightarrow$  pass rate of 53.8%.
- Of the 32 students who attempted at least 5 practicals, 25 passed  $\Rightarrow$  pass rate of 78.1%.
- Of the 27 students who attempted at least 7 practicals, 24 passed  $\Rightarrow$  pass rate of 88.8%.



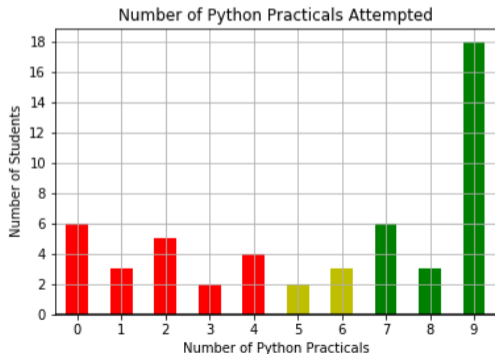
Keep up with the material:

- Read notes before & after lectures
- Attend practicals and upload practical work on time.
- Attempt tutorial questions.

Note "Attempted"  $\neq$  "Completed correctly"

## A Brief Look at 2019/20 Results

- 52 students enrolled, but only 28 passed!  $\Rightarrow$  pass rate of 53.8%.
- Of the 32 students who attempted at least 5 practicals, 25 passed  $\Rightarrow$  pass rate of 78.1%.
- Of the 27 students who attempted at least 7 practicals, 24 passed  $\Rightarrow$  pass rate of 88.8%.



Keep up with the material:

- Read notes before & after lectures
- Attend practicals and upload practical work on time.
- Attempt tutorial questions.

Note “Attempted”  $\neq$  “Completed correctly”

## A Brief Look at Last Year's Results (2023/24)

- 88 students, pass/compensation rate of 64%. (better but can still improve)
- Comparison of Practical Work (CA) vs Final Exam:

- One student passed while failing the CA

It is possible to pass this module without doing the practical work but the odds are against you.

- Average grade on CA was 66.1% while on Final Exam was 47.5%

The CA is graded easier (a carrot to help keep you motivated during the semester).

- Average number of practicals attempted last year was 6.6.

While (like in 2019) the more practicals a student attempted the more likely they passed the module, but if you miss a week or two it is not "end of the world".



## A Brief Look at Last Year's Results (2023/24)

- 88 students, pass/compensation rate of 64%. (better but can still improve)
- Comparison of Practical Work (CA) vs Final Exam:

- One student passed while failing the CA

It is possible to pass this module without doing the practical work but the odds are against you.

- Average grade on CA was 66.1% while on Final Exam was 47.5%

The CA is graded easier (a carrot to help keep you motivated during the semester).

- Average number of practicals attempted last year was 6.6.

While (like in 2019) the more practicals a student attempted the more likely they passed the module, but if you miss a week or two it is not “end of the world”.

# Who?

## Dr Denis Flynn

- 📠 Room 327
- ☎ 051-30 2068
- ✉ Denis.Flynn@setu.ie



### Background

- PhD in Applied Mathematics and Civil Engineering.
- BSc (H) Physics.

### Academic Interests

- Dynamical systems, in particular systems with hysteresis
- Game development
- Languages: C/C++, Python, Java

## Dr Kieran Murphy

- 📠 Room 313
- ☎ 051-30 2055
- ✉ Kieran.Murphy@setu.ie



### Background

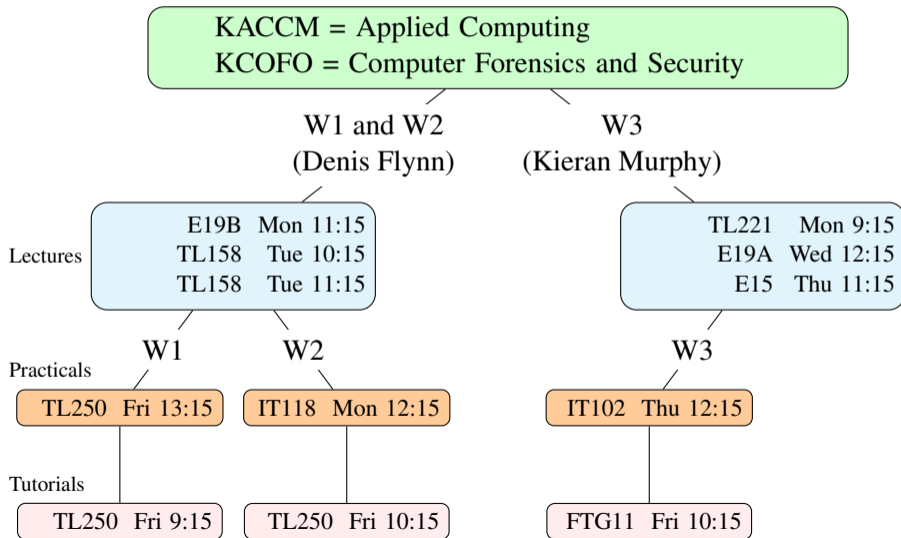
- PhD in Applied Mathematical Sciences
- BSc (H) Applied Mathematics.

### Academic Interests

- Dynamical systems, in particular numerical analysis
- Game development
- Languages: C/C++, Python, Java

## When?

(Correct as of 11 Sep 2024)



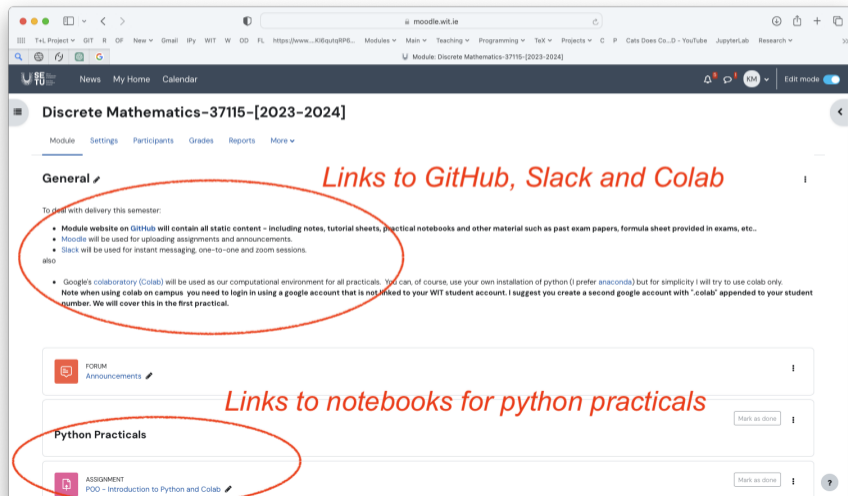
# Outline

---

1. Module Introduction	2
1.1. <b>What</b> is the Aim of this Module?	3
1.2. <b>Why</b> Study Discrete Mathematics?	12
1.3. <b>How</b> will the Module be Delivered? Assessed?	14
1.4. <b>Who</b> is delivering this module?	18
1.5. <b>When</b> will the module be delivered?	19
2. Resources	20
2.1. Moodle	21
2.2. Github	22
2.3. Slack	23
2.4. Colab	24
2.5. PyTutor	25
3. Final Comments	27

# Resources — Moodle

- URL: [moodle.wit.ie/course/view.php?id=201785](https://moodle.wit.ie/course/view.php?id=201785)
- Used for all notices, assignment and practical work submissions.



Discrete Mathematics-37115-[2023-2024]

Module Settings Participants Grades Reports More

**General**

To deal with delivery this semester:

- **Module website on GitHub** will contain all static content – including notes, tutorial sheets, practical notebooks and other material such as past exam papers, formula sheet provided in exams, etc..
- Moodle will be used for uploading assignments and announcements.
- Slack will be used for instant messaging, one-to-one and zoom sessions.

also

- Google's **colaboratory (Colab)** will be used as our computational environment for all practicals. You can, of course, use your own installation of python (I prefer *anaconda*) but for simplicity I will try to use colab only.  
**Note when using colab on campus you need to login in using a google account that is not linked to your WIT student account. I suggest you create a second google account with ".colab" appended to your student number. We will cover this in the first practical.**

**FORUM**  
Announcements

**Python Practicals**

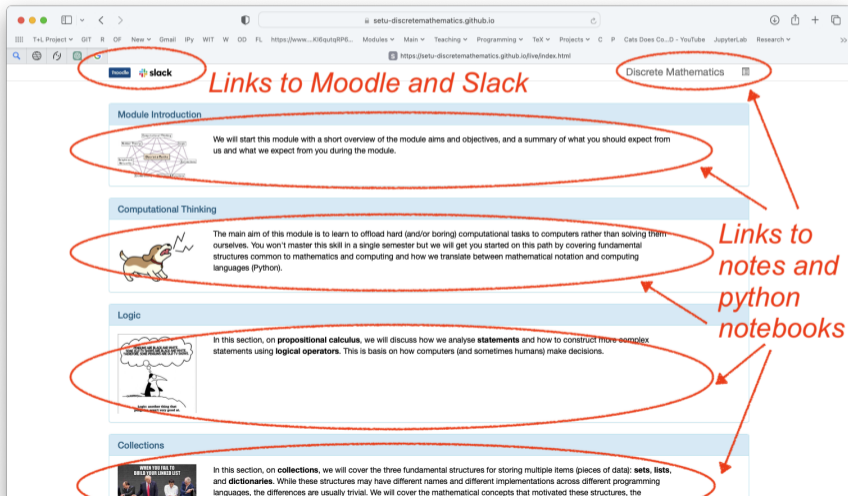
**ASSIGNMENT**  
POO - Introduction to Python and Colab

*Links to GitHub, Slack and Colab*

*Links to notebooks for python practicals*

# Resources — Github

- URL: [SETU-DiscreteMathematics.github.io/live](https://setu-discretemathematics.github.io/live)
- Used for all content (slides, notebooks, tutorial sheets).



The screenshot shows a web browser displaying the website <https://setu-discretemathematics.github.io/live>. The page has a navigation bar with a search icon, a Slack icon, and a "Discrete Mathematics" dropdown menu. The main content is organized into sections: "Module Introduction", "Computational Thinking", "Logic", and "Collections". Each section contains a title, an image, and a paragraph of text. Red circles and arrows highlight specific elements: the Slack icon, the "Discrete Mathematics" menu, the "Module Introduction" section, the "Computational Thinking" section, the "Logic" section, and the "Collections" section. Red text annotations are placed over the page: "Links to Moodle and Slack" is written in red over the Slack icon; "Links to notes and python notebooks" is written in red over the "Computational Thinking" and "Logic" sections.

**Links to Moodle and Slack**

**Links to notes and python notebooks**

**Module Introduction**

We will start this module with a short overview of the module aims and objectives, and a summary of what you should expect from us and what we expect from you during the module.

**Computational Thinking**

The main aim of this module is to learn to offload hard (and/or boring) computational tasks to computers rather than solving them ourselves. You won't master this skill in a single semester but we will get you started on this path by covering fundamental structures common to mathematics and computing and how we translate between mathematical notation and computing languages (Python).

**Logic**

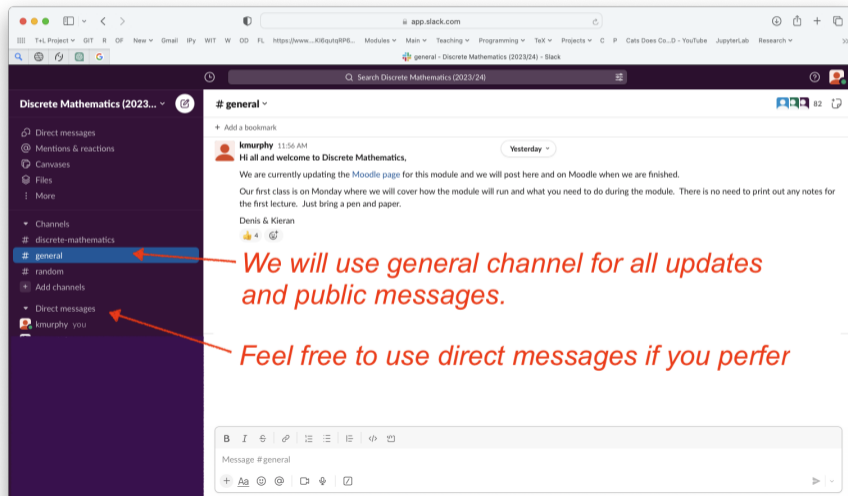
In this section, on **propositional calculus**, we will discuss how we analyse **statements** and how to construct more complex statements using **logical operators**. This is basis on how computers (and sometimes humans) make decisions.

**Collections**

In this section, on **collections**, we will cover the three fundamental structures for storing multiple items (pieces of data): **sets**, **lists**, and **dictionaries**. While these structures may have different names and different implementations across different programming languages, the differences are usually trivial. We will cover the mathematical concepts that motivated these structures, the

# Resources — Slack

- URL: [discretemathe-7co3349.slack.com](https://discretemathe-7co3349.slack.com)
- Used for instant messaging, one-on-one sessions, etc.



*We will use general channel for all updates and public messages.*

*Feel free to use direct messages if you prefer*

## Resources — Colab

- We will use the online Google Colab<sup>¶</sup> environment to code in python for all of our practical work.
- You can open a notebook from these slides by clicking the "OPEN in COLAB" icon or clicking/scanning the QR code



Since using Colab is such a large part of our module we have a separate set of slides covering using Colab in more detail.

A screenshot of a web browser displaying a Google Colab notebook. The browser's address bar shows "colab.research.google.com". The notebook interface includes a toolbar with icons for search, play, and a file explorer. A code cell is visible with the following Python code:

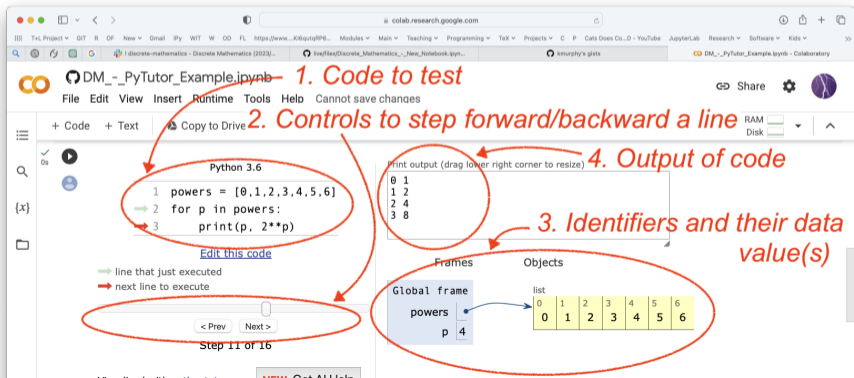
```
1 # Setup magic, do not edit this cell! Just press Shift+Enter or click on arrow at top-left
2
3 import urllib.request
4 content = urllib.request.urlretrieve ("https://setu-discretemathematics.github.io/live/files/setup_
5 exec(open(content[0]).read())
6 setup_practical(locale())
```

<sup>¶</sup>Alternatively, if you want to install python on your laptop you could use the anaconda distribution from [www.anaconda.com](http://www.anaconda.com) (just install the latest 64-bit, version 3.+).



# Resources — PyTutor

- PyTutor ([pythontutor.com](http://pythontutor.com)) is a website that helps programmers to learn Python, Javascript, C/C++, and Java by visualising code execution (shows what happens to data as code runs, line by line).
- We will run PyTutor from within Colab so will demo PyTutor when we cover Colab (or click/scan the QR code).

The screenshot shows the PyTutor interface within a Google Colab notebook. The code being executed is:

```

Python 3.6
1 powers = [0,1,2,3,4,5,6]
2 for p in powers:
3     print(p, 2**p)

```

The interface includes a code editor, a runtime control panel, and a visual output area. Red annotations highlight key features:

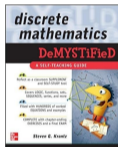
- 1. Code to test:** Points to the code in the editor.
- 2. Controls to step forward/backward a line:** Points to the '< Prev' and 'Next >' buttons at the bottom.
- 3. Identifiers and their data value(s):** Points to the 'Frames' and 'Objects' panels. The 'Global frame' shows 'powers' as a list and 'p' as 4. The 'Objects' panel shows a list with values [0, 1, 2, 3, 4, 5, 6].
- 4. Output of code:** Points to the 'Print output' area showing the results of the code execution:
 

0	1
1	2
2	4
3	8

Additional annotations include a legend for line execution (green arrow for 'line that just executed', red arrow for 'next line to execute') and a 'Print output (drag lower right corner to resize)' label.

# Text Books

I like the following textbooks on discrete mathematics and expect that my notes will overlap significantly with these books. I do encourage you to read\* them<sup>†</sup>, however, be aware they may use different notation or cover different topics.



## Discrete Mathematics Demystified

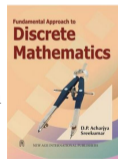
by Steven Krantz

Touches on nearly all of the topics that we hope to cover. We will probably go into greater depth in places, but a very nice and short read.

## Fundamental Approach to Discrete Mathematics

by D. P. Acharjya Sreekumar

I also liked this book, however, due to time constraints, this module only focuses on material in chapter 1–4, 8, and 10.




---

\*or skim them over a coffee or two.

<sup>†</sup>I also like *Applied Discrete Structures* by Alan Doerr and Kenneth Levasseur — it is a good source of exercises. (and is free (legally))

# Outline

---

1. Module Introduction	2
1.1. <b>What</b> is the Aim of this Module?	3
1.2. <b>Why</b> Study Discrete Mathematics?	12
1.3. <b>How</b> will the Module be Delivered? Assessed?	14
1.4. <b>Who</b> is delivering this module?	18
1.5. <b>When</b> will the module be delivered?	19
2. Resources	20
2.1. Moodle	21
2.2. Github	22
2.3. Slack	23
2.4. Colab	24
2.5. PyTutor	25
3. Final Comments	27

## Final Comments on Module

- Discrete Mathematics concepts appear either directly or indirectly in approximately 22 of the 30 modules on your degree.
  - ⇒ *Knowing Discrete Mathematics concepts greatly simplifies rest of the course.*
- The module is intended to be an introduction to a large number of topics, so treatment is broad rather than deep.
  - ✓ Most of material is at an introductory level.
  - ⚠ Keeping in sync with material, practicals and tutorials is important.
- The continuous assessment (the practicals) is intended to reinforce the connections between programming and discrete mathematics.

The CA is a “carrot not a stick” — we want you to enjoy the module and keep up to date with the material.

