Welcome to COSC 123!

Introduction to COSC 123 – Computer Creativity

Okanagan

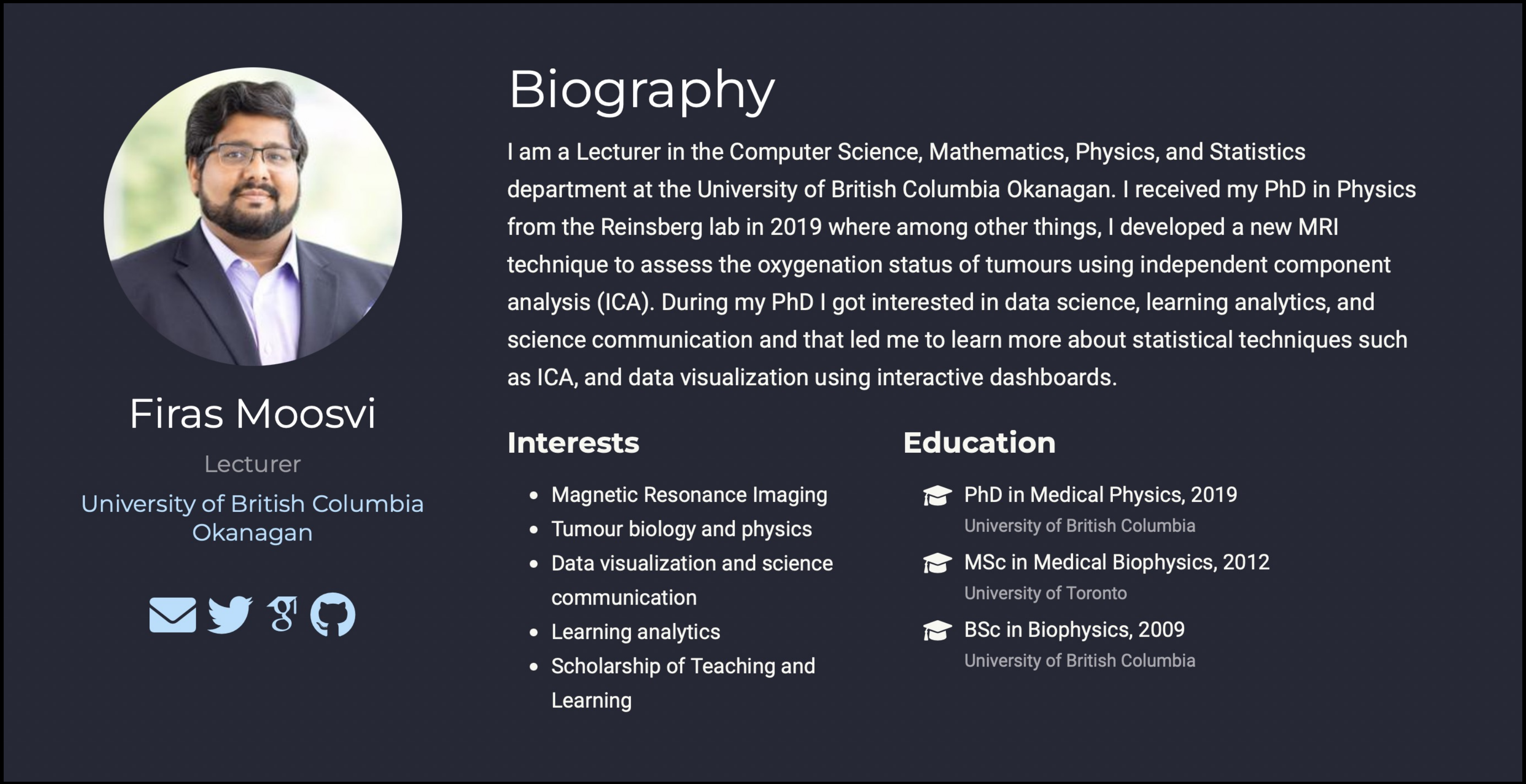
Slides courtesy of Dr. Abdallah Mohamed.

Credits and Acknowledgements

- Many thanks to Drs. Abdallah Mohammed for creating the course notes, which we will instead of a “textbook” in this course

- Thanks also to Dr. Ramon Lawrence for creating the original version of this course and creating a vision of what this course should be.

About me



Firas Moosvi

Lecturer

University of British Columbia Okanagan

Biography

I am a Lecturer in the Computer Science, Mathematics, Physics, and Statistics department at the University of British Columbia Okanagan. I received my PhD in Physics from the Reinsberg lab in 2019 where among other things, I developed a new MRI technique to assess the oxygenation status of tumours using independent component analysis (ICA). During my PhD I got interested in data science, learning analytics, and science communication and that led me to learn more about statistical techniques such as ICA, and data visualization using interactive dashboards.

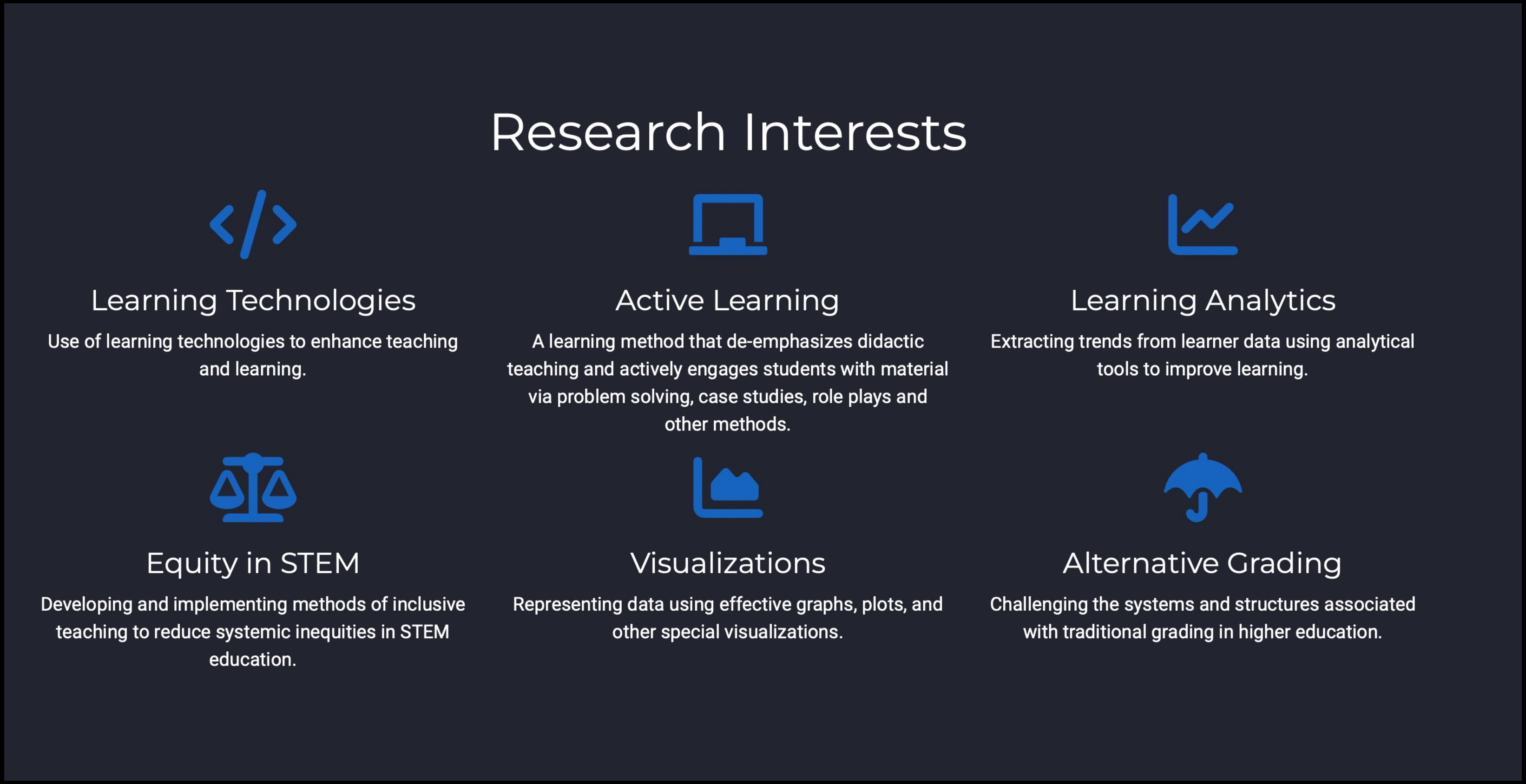
Interests

- Magnetic Resonance Imaging
- Tumour biology and physics
- Data visualization and science communication
- Learning analytics
- Scholarship of Teaching and Learning

Education

- PhD in Medical Physics, 2019
  University of British Columbia
- MSc in Medical Biophysics, 2012
  University of Toronto
- BSc in Biophysics, 2009
  University of British Columbia

Research Interests



Research Interests

Learning Technologies

Use of learning technologies to enhance teaching and learning.

Active Learning

A learning method that de-emphasizes didactic teaching and actively engages students with material via problem solving, case studies, role plays and other methods.

Learning Analytics

Extracting trends from learner data using analytical tools to improve learning.

Equity in STEM

Developing and implementing methods of inclusive teaching to reduce systemic inequities in STEM education.

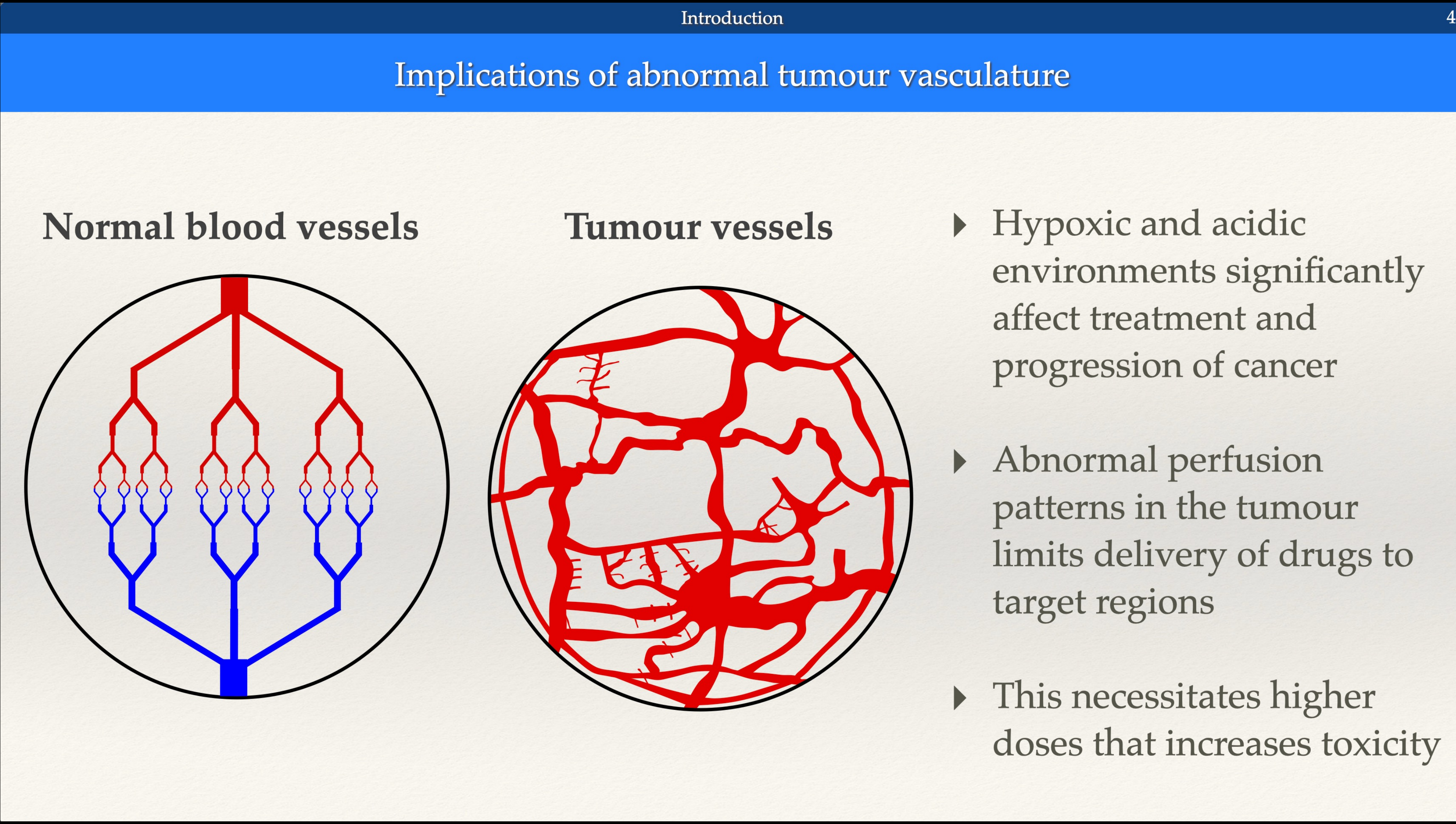
Visualizations

Representing data using effective graphs, plots, and other special visualizations.

Alternative Grading

Challenging the systems and structures associated with traditional grading in higher education.

Research Interests



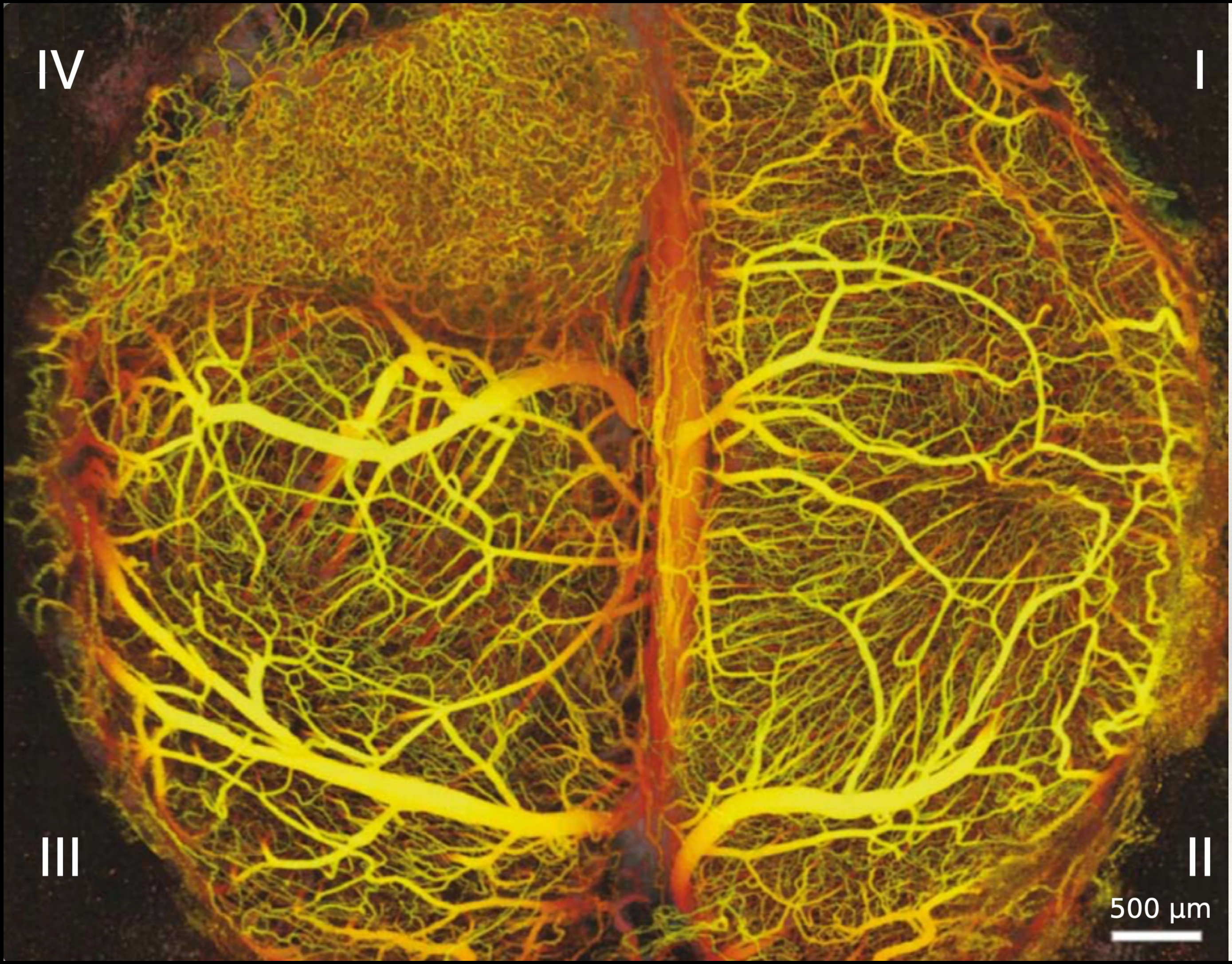
Implications of abnormal tumour vasculature

Normal blood vessels

Tumour vessels

- Hypoxic and acidic environments significantly affect treatment and progression of cancer
- Abnormal perfusion patterns in the tumour limits delivery of drugs to target regions
- This necessitates higher doses that increases toxicity

Research Interests



Optical Frequency Domain Imaging (OFDI)

- Anaesthetized mouse brain imaged through **cranial windows** using optical imaging techniques
- Vessel colour encodes **depth**; closer vessels are yellow and further vessels are red
- Note the normal brain vascular branching patterns in quadrants I, II, and II compared to the chaotic network of the **U87 tumour** in quadrant IV

Vakoc et al (2009). *Nature Medicine Technical Reports*

Processing

Slides courtesy of Dr. Abdallah Mohamed.

Course Objectives

1) To be creative with programming and write fun, interesting computer programs.
2) To master fundamental programming skills of data variables, decisions, iteration, methods, and the basics of object-oriented programming, and how to create larger programs
3) To design and develop strategies for solving basic programing problems.
4) To algorithmically create 2D graphics, animations, and simple games using Processing language.
5) To design interactive graphical user interfaces.
6) To learn how to switch from Processing to Java.

The Essence of the Course

- If you walk out of this course with nothing else you should:

Become a creative programmer with the ability to problem solve, perform critical thinking, and communicate precisely.

- This course is not only about learning a particular language (or even programming itself), it is about being a creative problem solver and critical thinker!

Programming using Processing

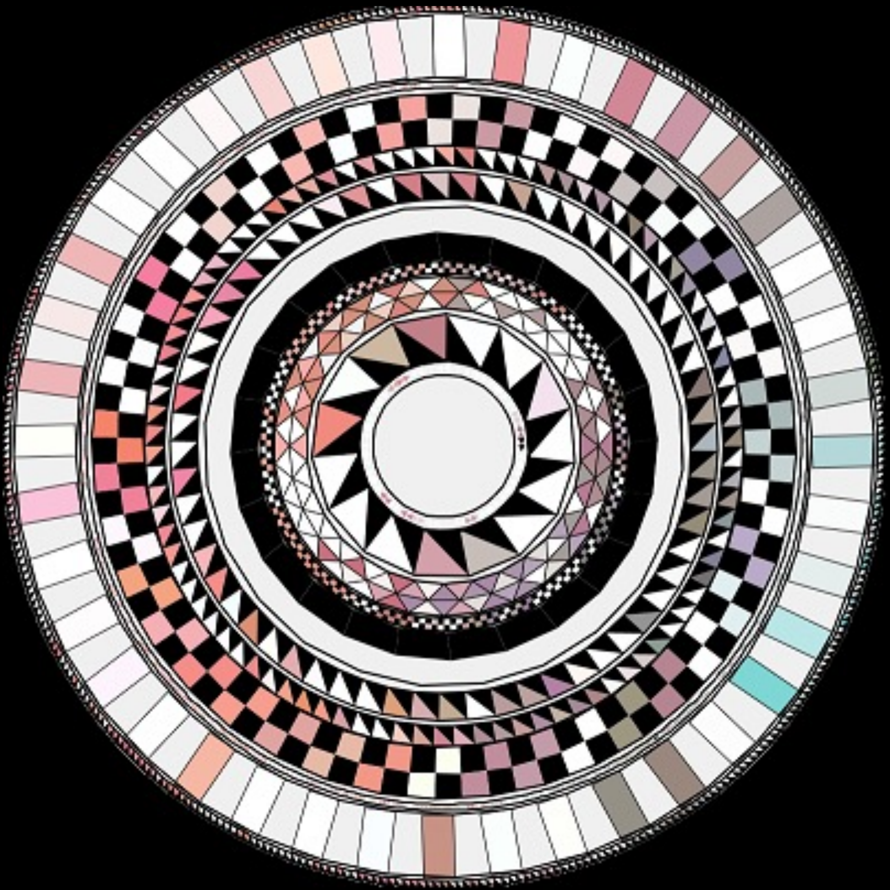
- You already learned algorithmic thinking using basic programming techniques in COSC111 and COSC122.

- In addition to being able to solving algorithmic problems (similar to what you did in COSC 111 or COSC 122), we will try to re-learn programming using graphical functions, especially to create user interfaces, animations, and simple 2D games.

- We will use basic programming techniques (e.g. conditionals, loops, arrays, objects, etc.) on *Sketches* to draw and interact with shapes and images.

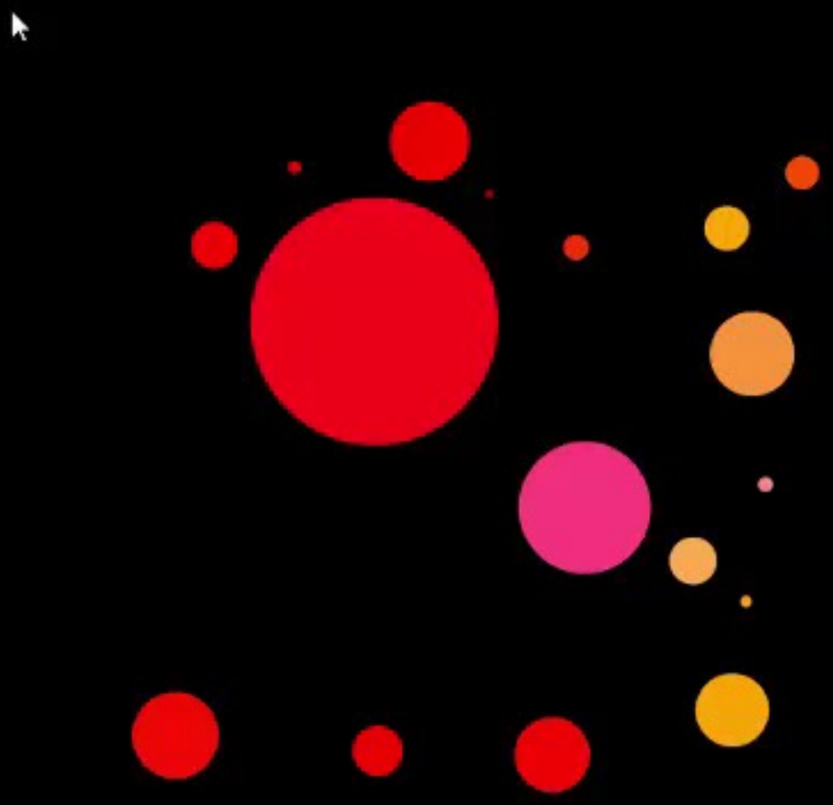
Processing Examples!

- Algorithmic Drawing
  - Example: Artistic Designs



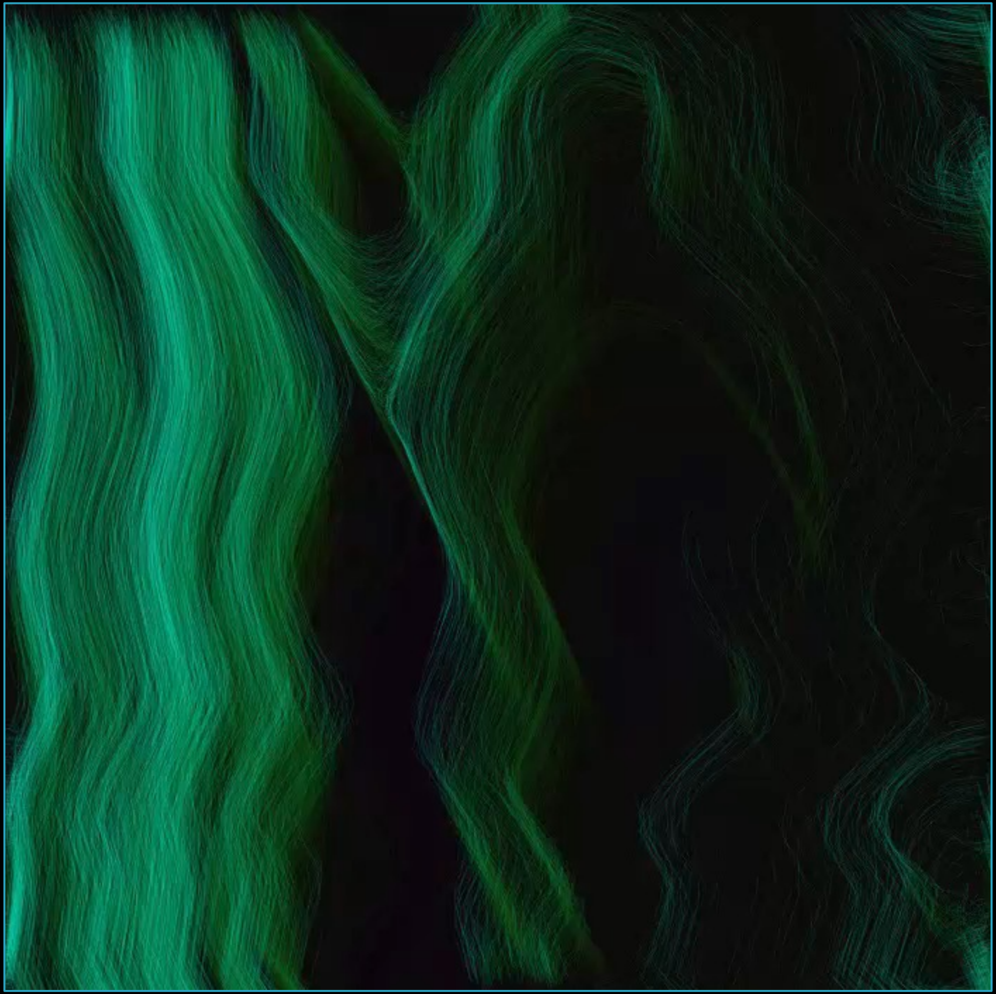
Processing Examples!

- Artistic Animations
  - Example:



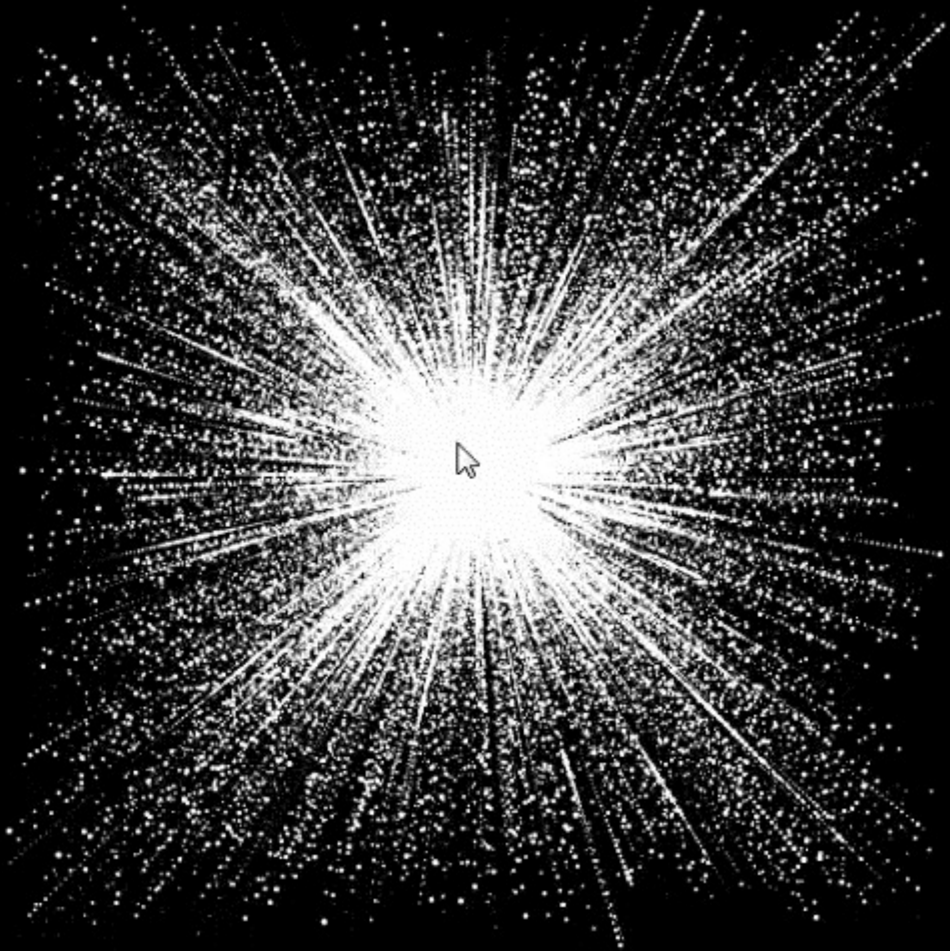
Processing Examples!

- Artistic Animations
  - Example: particle systems



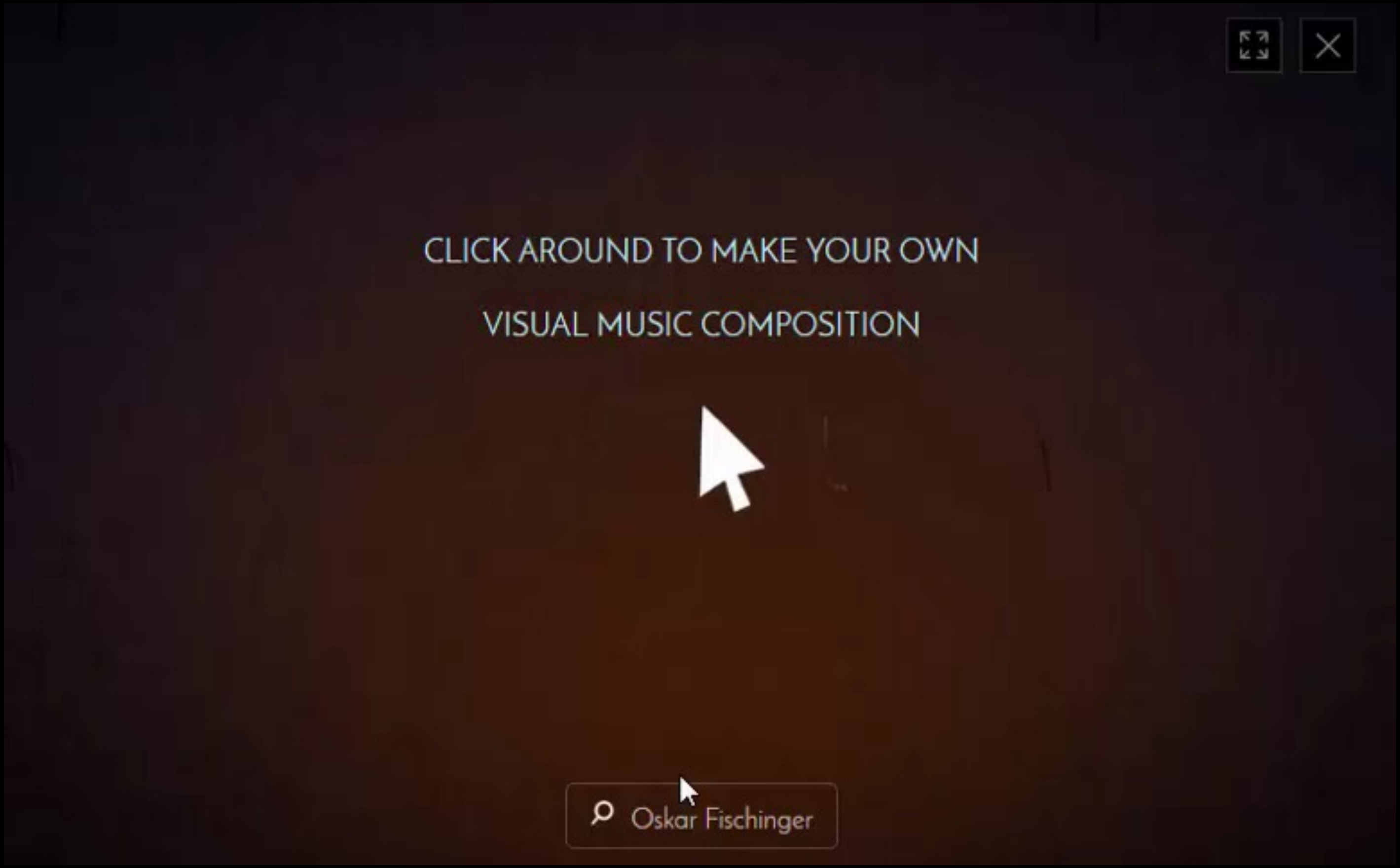
Processing Examples!

- Interactive Animations
  - Example: controlled particle system



Processing Examples!

- Interactive Animations
  - Example: Google Doodle (June 2017)



CLICK AROUND TO MAKE YOUR OWN

VISUAL MUSIC COMPOSITION

Oskar Fischinger

https://www.google.com/doodles/oskar-fischingers-117th-birthday

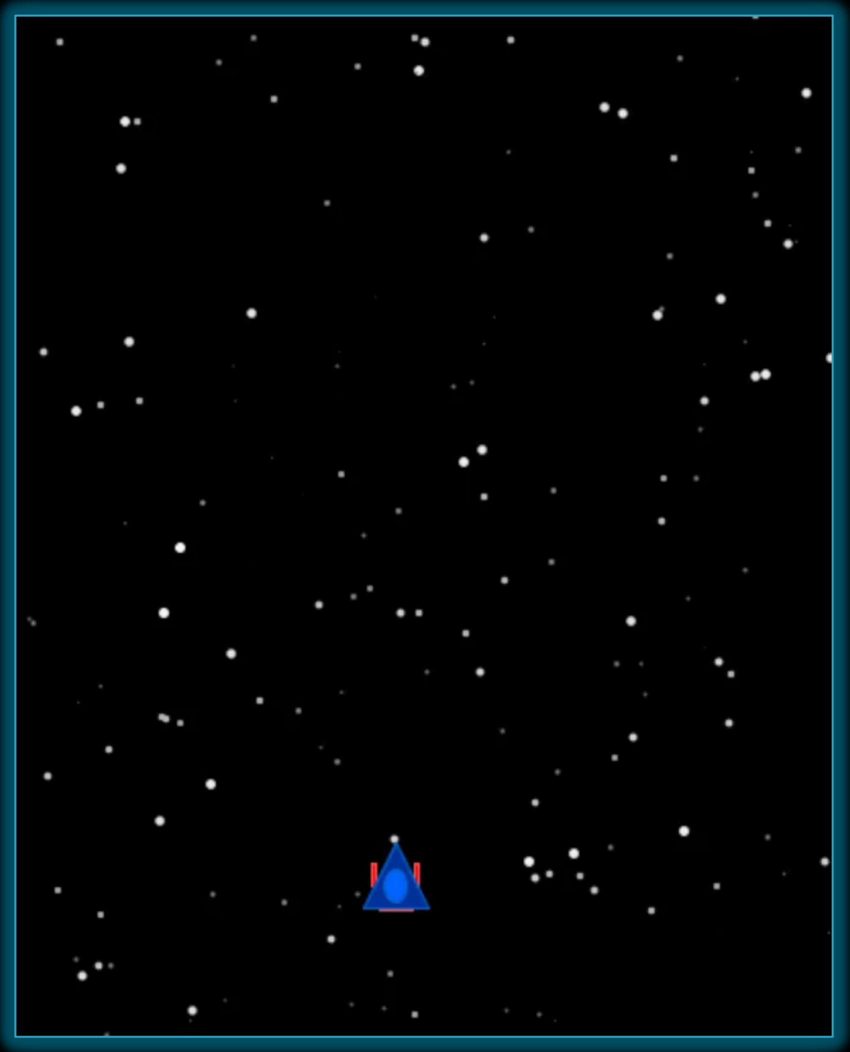
Processing Examples!

- Interactive Animations
  - Example: 2D Games

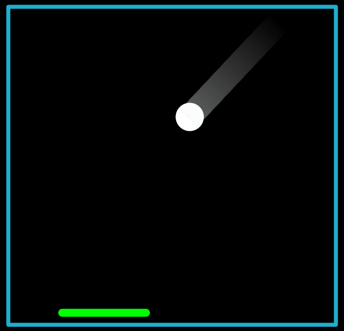
Game name: Toon Shooters 2

Example of things you will do!

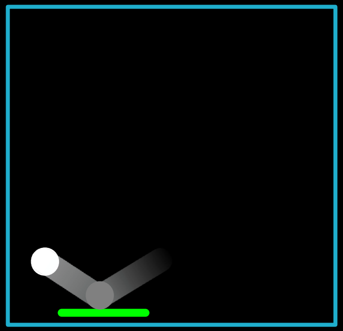
Example of things you will do!



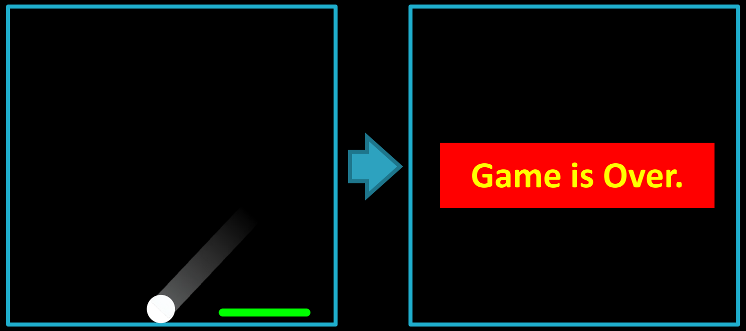
Example of things you will do!



Player needs to
move the paddle to
hit the ball up



When the ball is hit, score
is incremented and ball
speed increases



If ball touches bottom
edge, game is over,
and animation stops.

Why this Course is Important

- This course will make programming fun and relevant.
  - Our economy, health, and entertainment is dependent on software written by programmers.
  - We will learn to be creative programmers, so that we may create great software to be used by others.

- Important results:
  - ***Storyboarding*** – We will sketch our stories before programing them.
  - ***Algorithmic Thinking*** – We will learn how to solve problems by specifying precise sequences of actions.
  - ***Collaboration*** – We will program in teams of two to build interpersonal skills and increase our knowledge.
  - ***Processing and Java Languages*** – We will use Processing which is based on Java programming language – Java can be used in many areas including future computer science courses.

Ed Discussion (Demo)

Interface

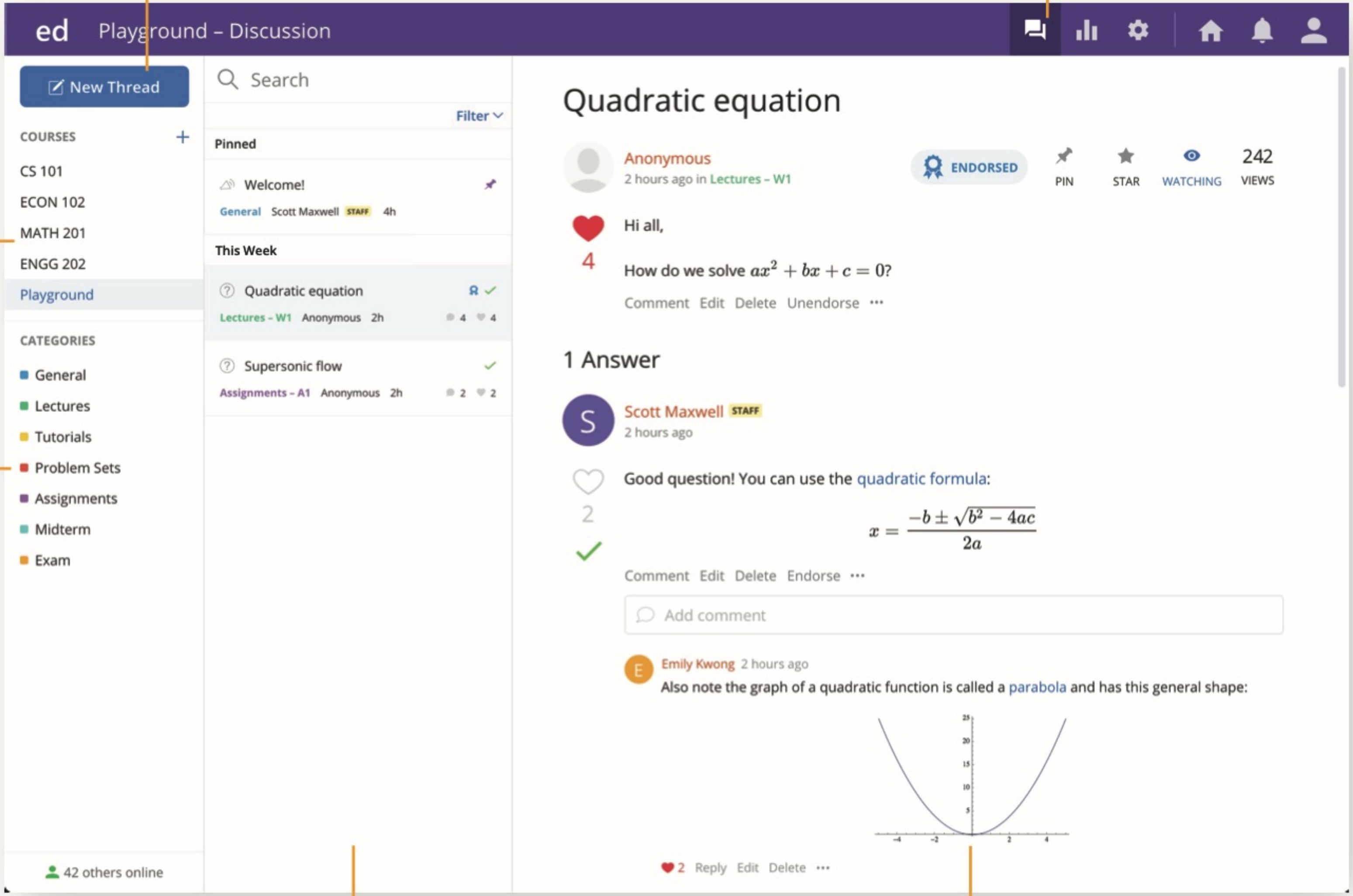
Clean and intuitive.

Start a **new thread**

Open **Ed Discussion**

Toggle between **courses**

Toggle between **categories**



Quadratic equation

Anonymous
2 hours ago in Lectures – W1

4

Hi all,

How do we solve $ax^2 + bx + c = 0$?

1 Answer

Scott Maxwell STAFF
2 hours ago

2

Good question! You can use the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Emily Kwong 2 hours ago
Also note the graph of a quadratic function is called a parabola and has this general shape:

2

Open a **thread**

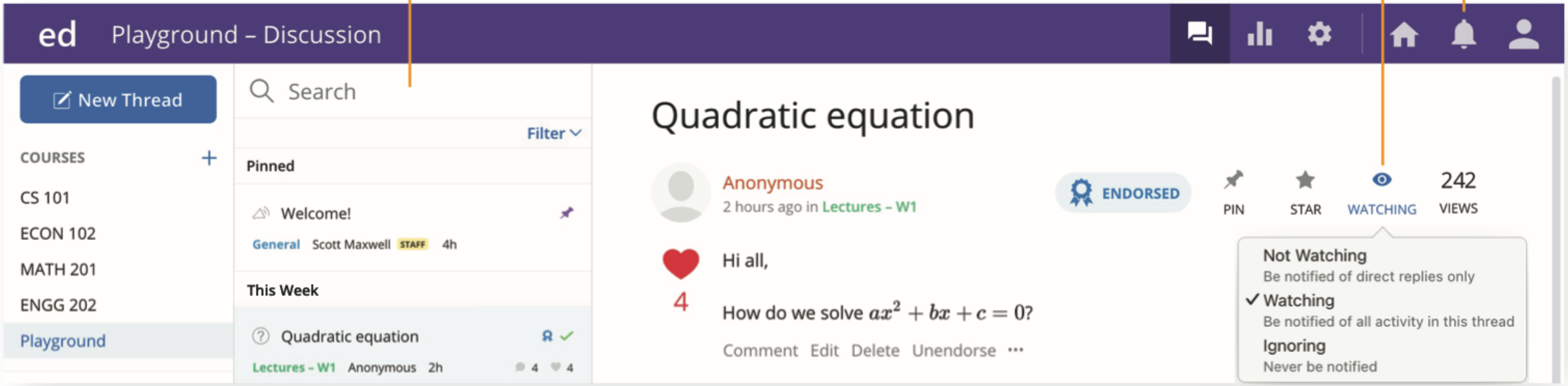
Read and **respond** to threads

Tips and tricks

Search and stay notified about threads.

Search for relevant **threads**

Stay **notified** about threads



Quadratic equation

Hi all,

How do we solve $ax^2 + bx + c = 0$?

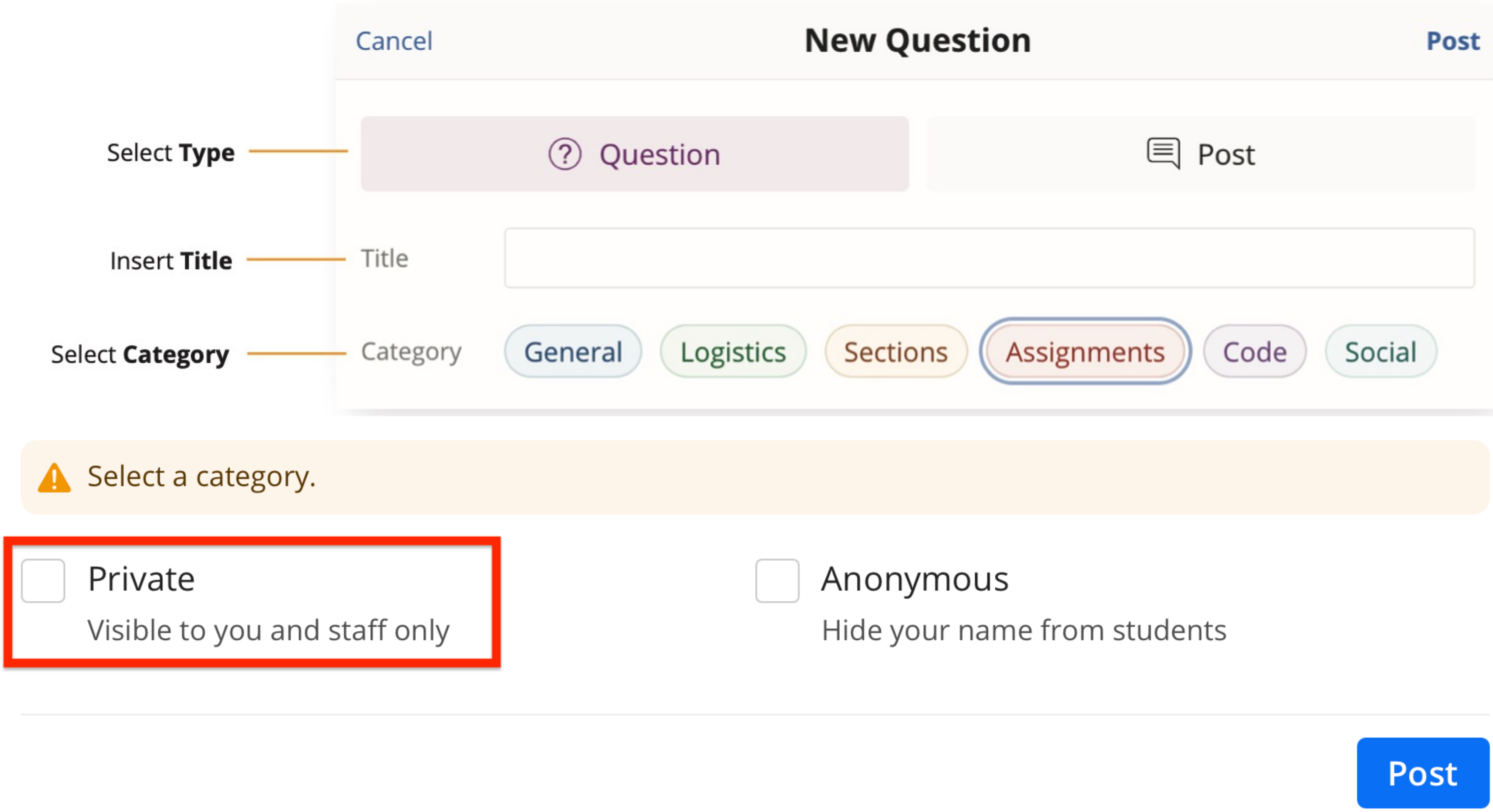
Not Watching
Be notified of direct replies only

✓ Watching
Be notified of all activity in this thread

Ignoring
Never be notified

Post a question

Ask, with confidence.



Select **Type**

Insert **Title**

Select **Category**

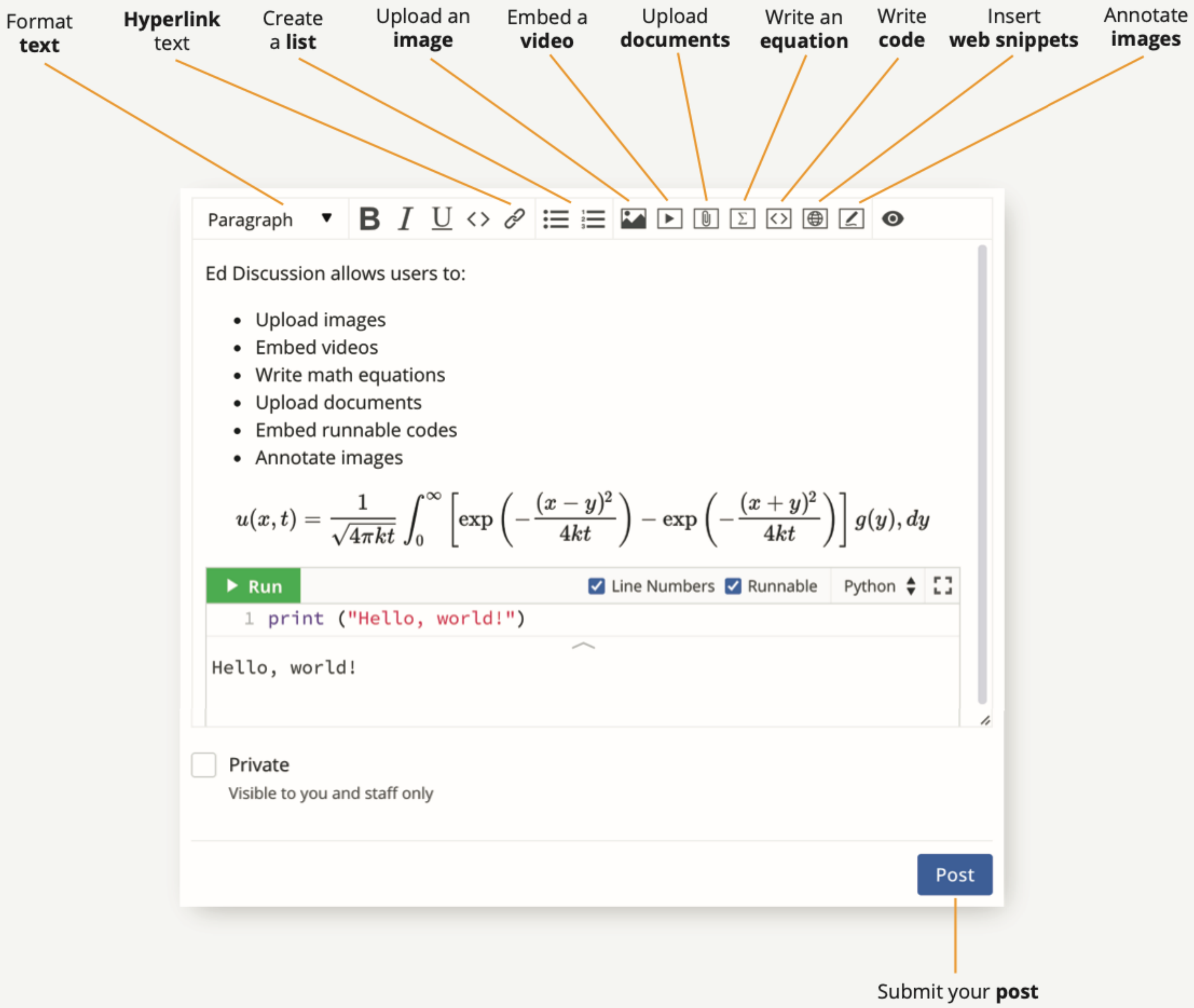
⚠ Select a category.

□ Private
Visible to you and staff only

□ Anonymous
Hide your name from students

Express yourself in any way

Superb all-in-one editor to better communicate your ideas.



Format **text**

Hyperlink text

Create a **list**

Upload an **image**

Embed a **video**

Upload **documents**

Write an **equation**

Write **code**

Insert **web snippets**

Annotate **images**

Ed Discussion allows users to:

- Upload images
- Embed videos
- Write math equations
- Upload documents
- Embed runnable codes
- Annotate images

$$u(x,t) = \frac{1}{\sqrt{4\pi kt}} \int_0^\infty \left[\exp\left(-\frac{(x-y)^2}{4kt}\right) - \exp\left(-\frac{(x+y)^2}{4kt}\right)\right] g(y), dy$$

☑ Line Numbers ☑ Runnable Python

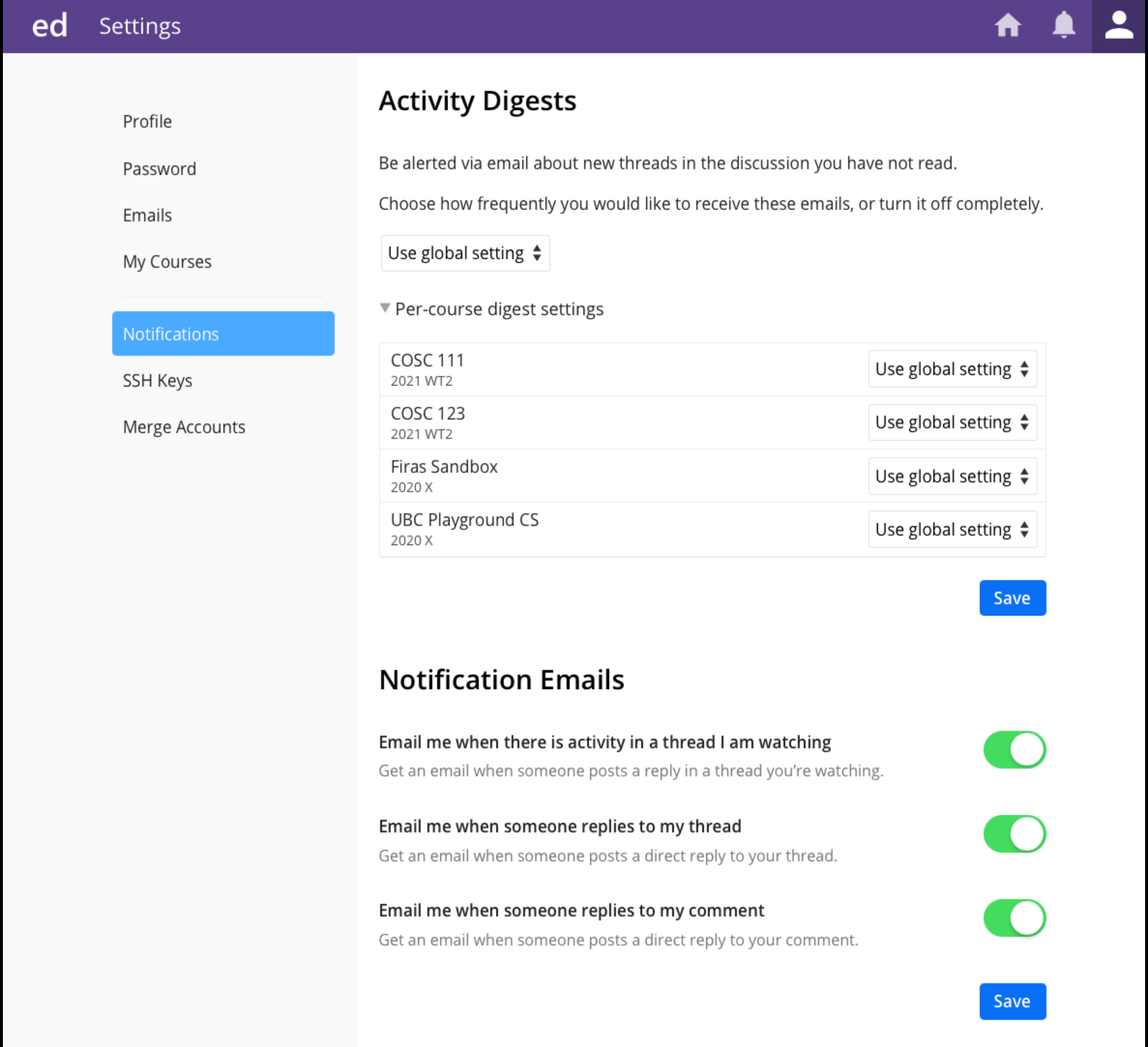
```python
print ("Hello, world!")
```

```
Hello, world!
```

□ Private

Visible to you and staff only

Submit your **post**



Activity Digests

Be alerted via email about new threads in the discussion you have not read.

Choose how frequently you would like to receive these emails, or turn it off completely.

Use global setting ▲▼

▼ Per-course digest settings

COSC 111 2021 WT2	Use global setting ▲▼
COSC 123 2021 WT2	Use global setting ▲▼
Firas Sandbox 2020 X	Use global setting ▲▼
UBC Playground CS 2020 X	Use global setting ▲▼

Notification Emails

Email me when there is activity in a thread I am watching ☑

Get an email when someone posts a reply in a thread you're watching.

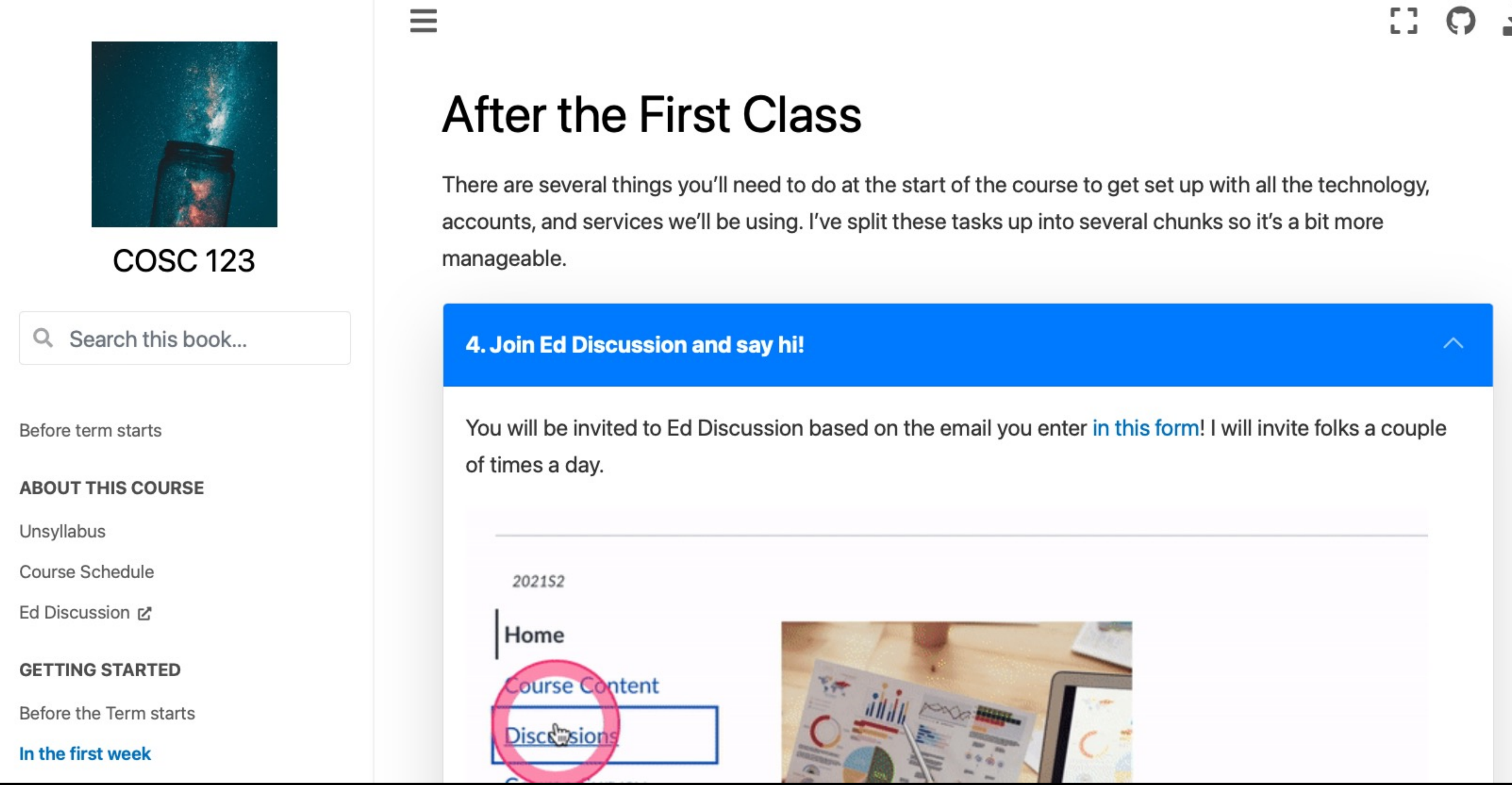
Email me when someone replies to my thread ☑

Get an email when someone posts a direct reply to your thread.

Email me when someone replies to my comment ☑

Get an email when someone posts a direct reply to your comment.

Get invited to Ed Discussion



COSC 123

Search this book...

Before term starts

ABOUT THIS COURSE

Unsyllabus

Course Schedule

Ed Discussion

GETTING STARTED

Before the Term starts

In the first week

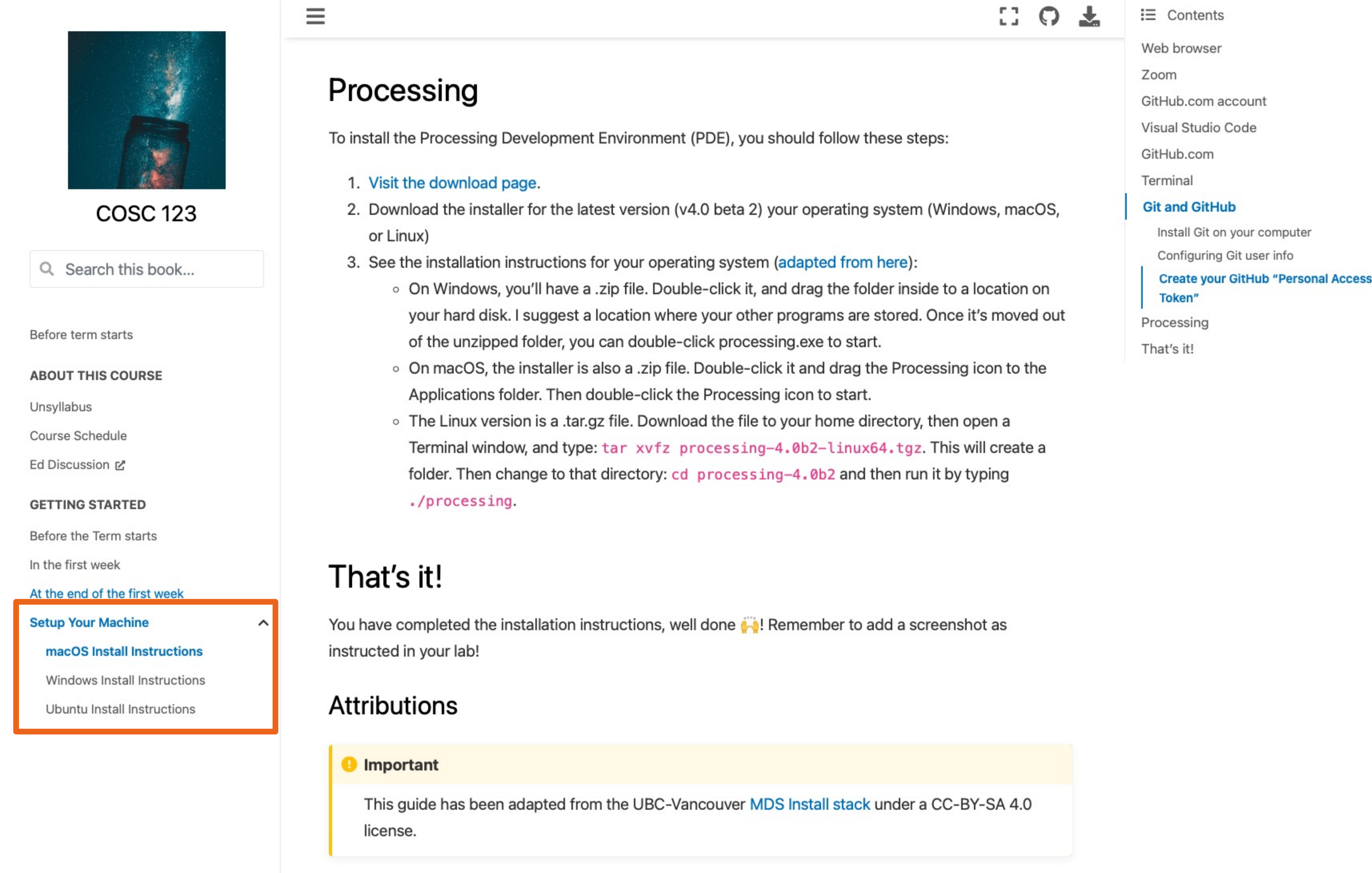
After the First Class

There are several things you'll need to do at the start of the course to get set up with all the technology, accounts, and services we'll be using. I've split these tasks up into several chunks so it's a bit more manageable.

4. Join Ed Discussion and say hi!

You will be invited to Ed Discussion based on the email you enter in this form! I will invite folks a couple of times a day.

Setup your Machine



COSC 123

Search this book...

Before term starts

ABOUT THIS COURSE

Unsyllabus

Course Schedule

Ed Discussion

GETTING STARTED

Before the Term starts

In the first week

At the end of the first week

Setup Your Machine

macOS Install Instructions

Windows Install Instructions

Ubuntu Install Instructions

Processing

To install the Processing Development Environment (PDE), you should follow these steps:

1. Visit the download page.
2. Download the installer for the latest version (v4.0 beta 2) your operating system (Windows, macOS, or Linux)
3. See the installation instructions for your operating system (adapted from here):
   - On Windows, you'll have a .zip file. Double-click it, and drag the folder inside to a location on your hard disk. I suggest a location where your other programs are stored. Once it's moved out of the unzipped folder, you can double-click processing.exe to start.
   - On macOS, the installer is also a .zip file. Double-click it and drag the Processing icon to the Applications folder. Then double-click the Processing icon to start.
   - The Linux version is a .tar.gz file. Download the file to your home directory, then open a Terminal window, and type: `tar xvfz processing-4.0b2-linux64.tgz`. This will create a folder. Then change to that directory: `cd processing-4.0b2` and then run it by typing `./processing`.

That's it!

You have completed the installation instructions, well done 🙌! Remember to add a screenshot as instructed in your lab!

Attributions

Important

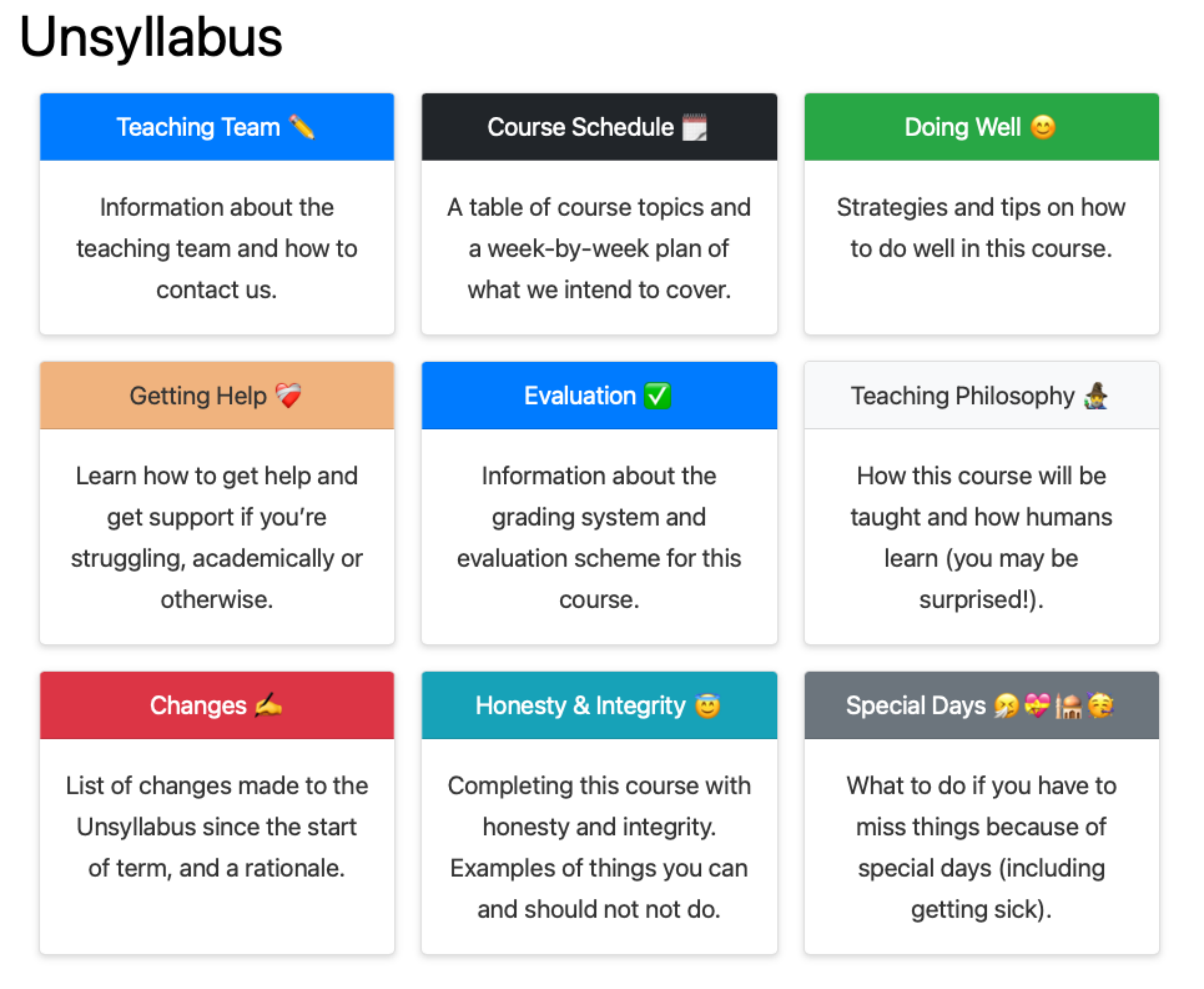
This guide has been adapted from the UBC-Vancouver MDS Install stack under a CC-BY-SA 4.0 license.

Break

Slides courtesy of Dr. Abdallah Mohamed.

Unsyllabus (Demo)

Navigating the course website



Unsyllabus

Teaching Team ✏️

Information about the teaching team and how to contact us.

Course Schedule 🗓️

A table of course topics and a week-by-week plan of what we intend to cover.

Doing Well 😊

Strategies and tips on how to do well in this course.

Getting Help ❤️‍🩹

Learn how to get help and get support if you're struggling, academically or otherwise.

Evaluation ✅

Information about the grading system and evaluation scheme for this course.

Teaching Philosophy 🧙

How this course will be taught and how humans learn (you may be surprised!).

Changes ✍️

List of changes made to the Unsyllabus since the start of term, and a rationale.

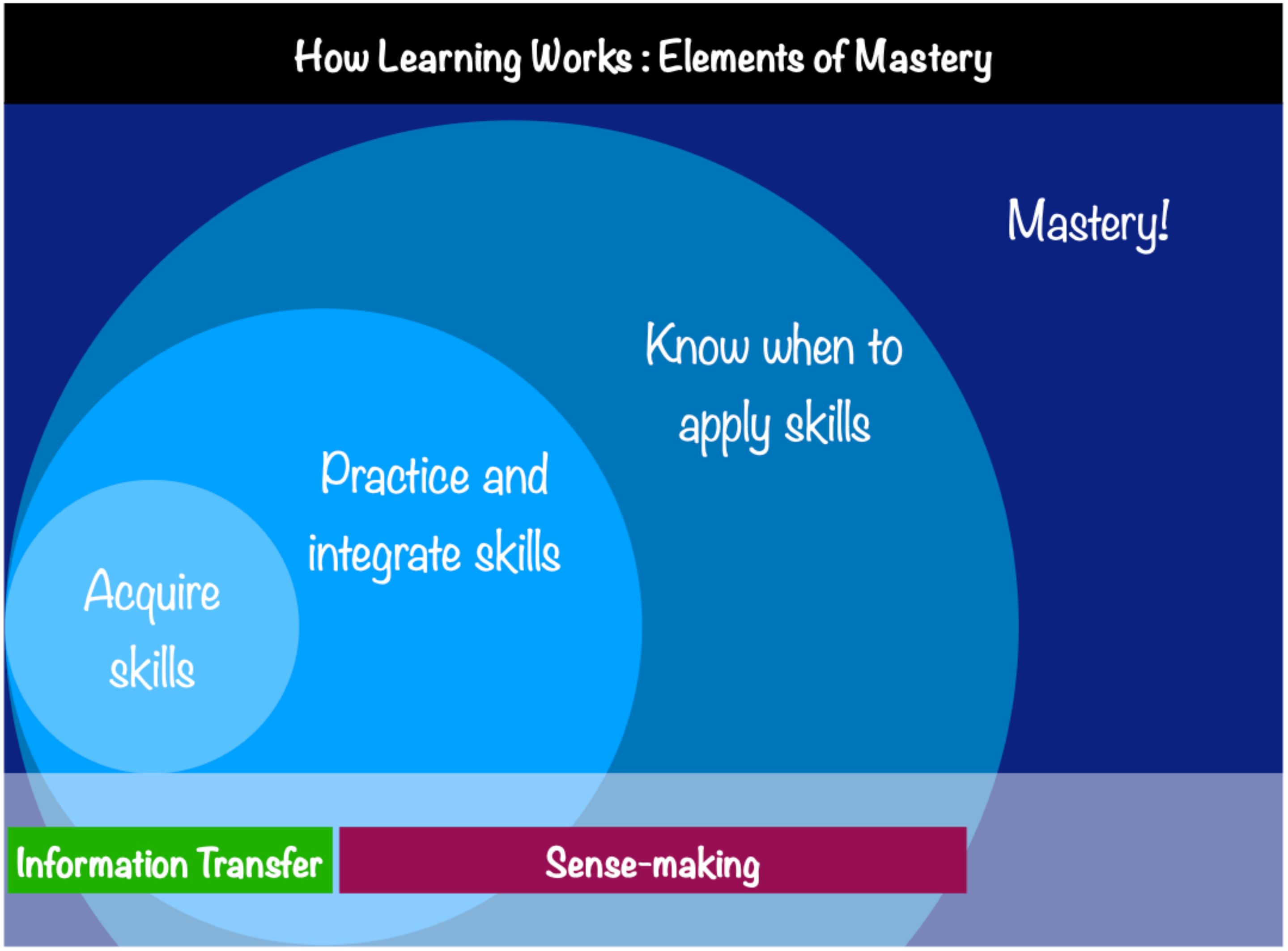
Honesty & Integrity 😇

Completing this course with honesty and integrity. Examples of things you can and should not not do.

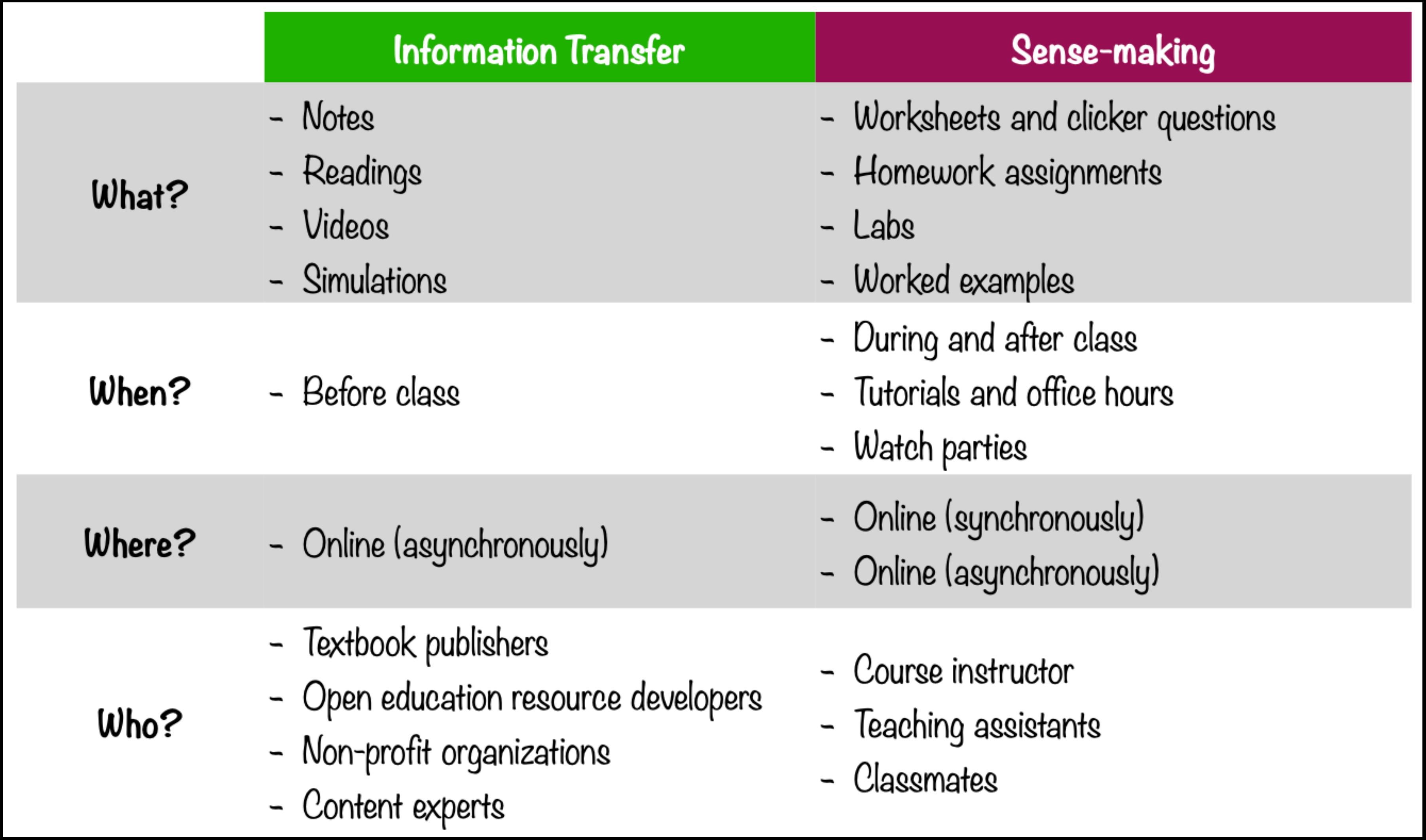
Special Days 🤧💝🕌🥳

What to do if you have to miss things because of special days (including getting sick).

How Learning Works


How Learning Works : Elements of Mastery
Mastery!
Know when to apply skills
Practice and integrate skills
Acquire skills
Information Transfer
Sense-making

How Learning Works



	Information Transfer	Sense-making
What?	- Notes - Readings - Videos - Simulations	- Worksheets and clicker questions - Homework assignments - Labs - Worked examples
When?	- Before class	- During and after class - Tutorials and office hours - Watch parties
Where?	- Online (asynchronously)	- Online (synchronously) - Online (asynchronously)
Who?	- Textbook publishers - Open education resource developers - Non-profit organizations - Content experts	- Course instructor - Teaching assistants - Classmates

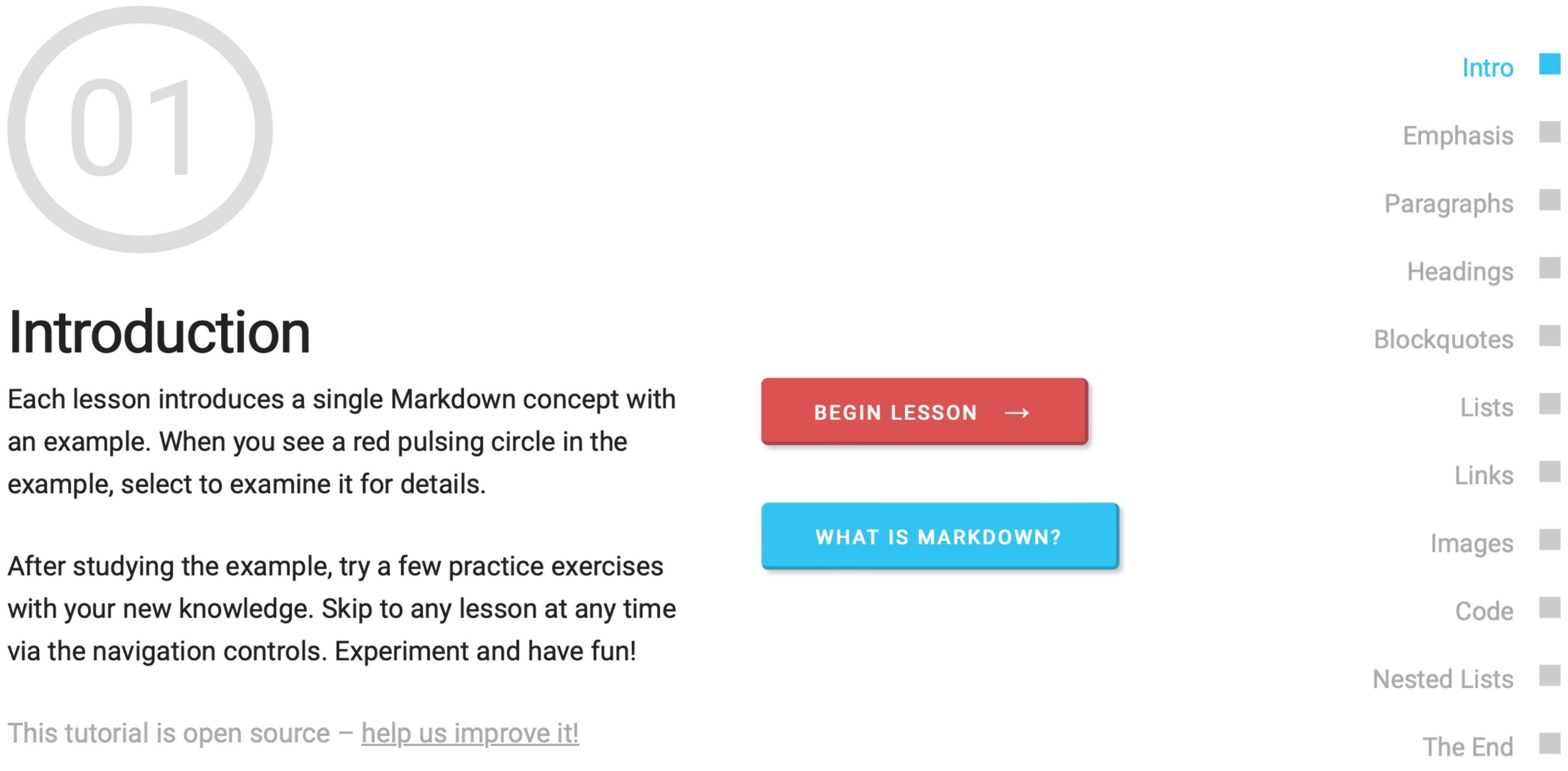
Course features

- 48 hour grace-period on all due dates and deadlines.
- Lab attendance is not mandatory (attend any and all sections that work for you).
- Classes are recorded, but not live streamed. Recordings are available 24-48 hours after the class.
- Many opportunities to demonstrate your learning.
- Weekly learning logs and reflections to make you think about your learning (metacognition).
- Each test has a "bonus test" available one week later; for each test, we will take the better score of the pair.

Course features

- No high-stakes exams (the single largest assessment item is the final exam).
- All course assessments are completely open book, open notes, and open web (except for cheating websites like Chegg, CourseHero, Slader, Bartleby, etc...)
- Plenty of TA and instructor student hours and several outside of normal business hours.
- Class website that outlines exactly what you should do when to help you manage your time.
- Tonnes of supplemental materials including other - instructional videos in case you want a different perspective.
- A true willingness from the instructor (me) to help you learn and succeed in this course!

Markdown Tutorial (20 mins)



Introduction

Each lesson introduces a single Markdown concept with an example. When you see a red pulsing circle in the example, select to examine it for details.

After studying the example, try a few practice exercises with your new knowledge. Skip to any lesson at any time via the navigation controls. Experiment and have fun!

This tutorial is open source – help us improve it!

See you on Friday!

Getting Started with Processing

Slides courtesy of Dr. Abdallah Mohamed.

Key Points

1) What is Processing

2) Experiment with the Processing Development Environment.

3) Printing on the console

The Processing Language

- Processing is a programming environment that aims to help create **visually oriented applications**, such as sketches, animations, and games.

- Processing consists of:
  - The Processing Development Environment (PDE).
    - The software we will use to write and run our code in this course.
    - Has a minimalist set of features suitable for developing small programs
  - The Processing core **API** and other libraries
    - A collection of functions (aka commands or methods) for performing the different actions in a program.
  - A language syntax identical to Java.
    - ***Processing is Java***, but with simpler syntax.
    - Processing was ported to other languages later (e.g. JS, Python).

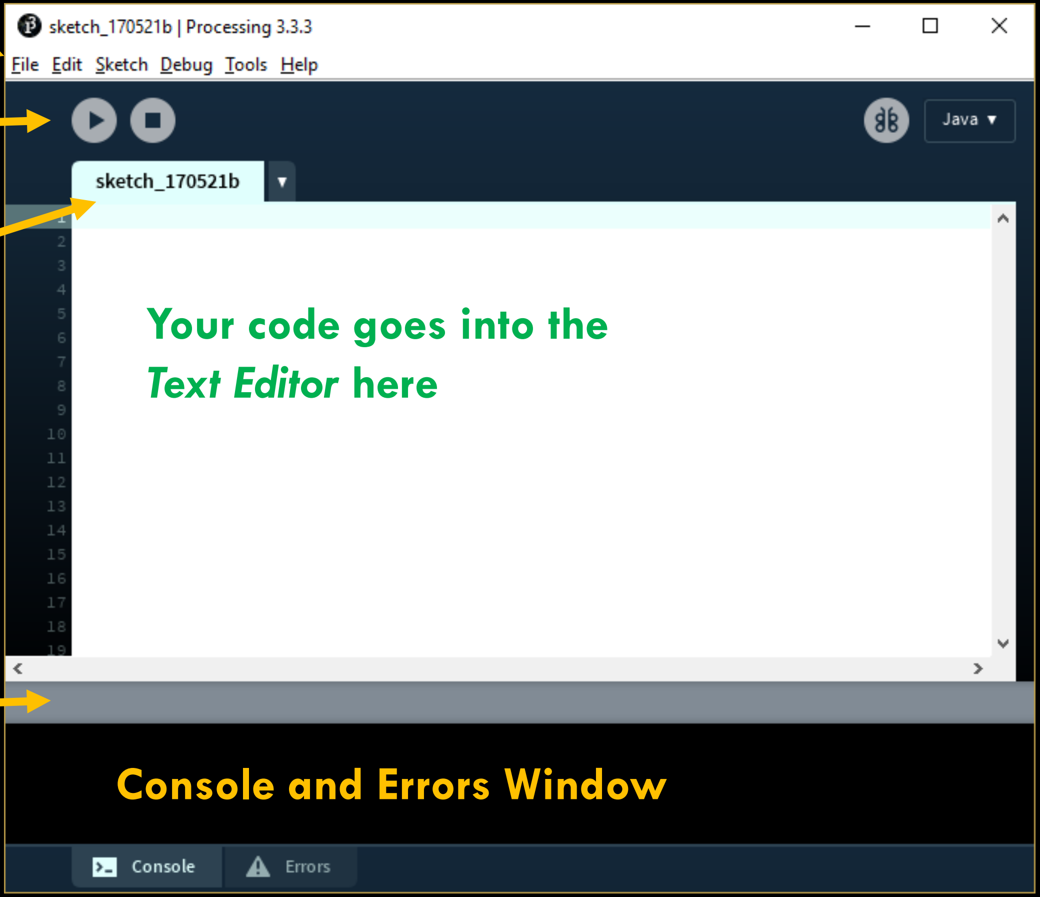
Processing Development Environment (PDE)

Menu

Run / Stop

Sketch name

Message Area



sketch_170521b | Processing 3.3.3
File Edit Sketch Debug Tools Help
Java ▾
sketch_170521b
Console
Errors

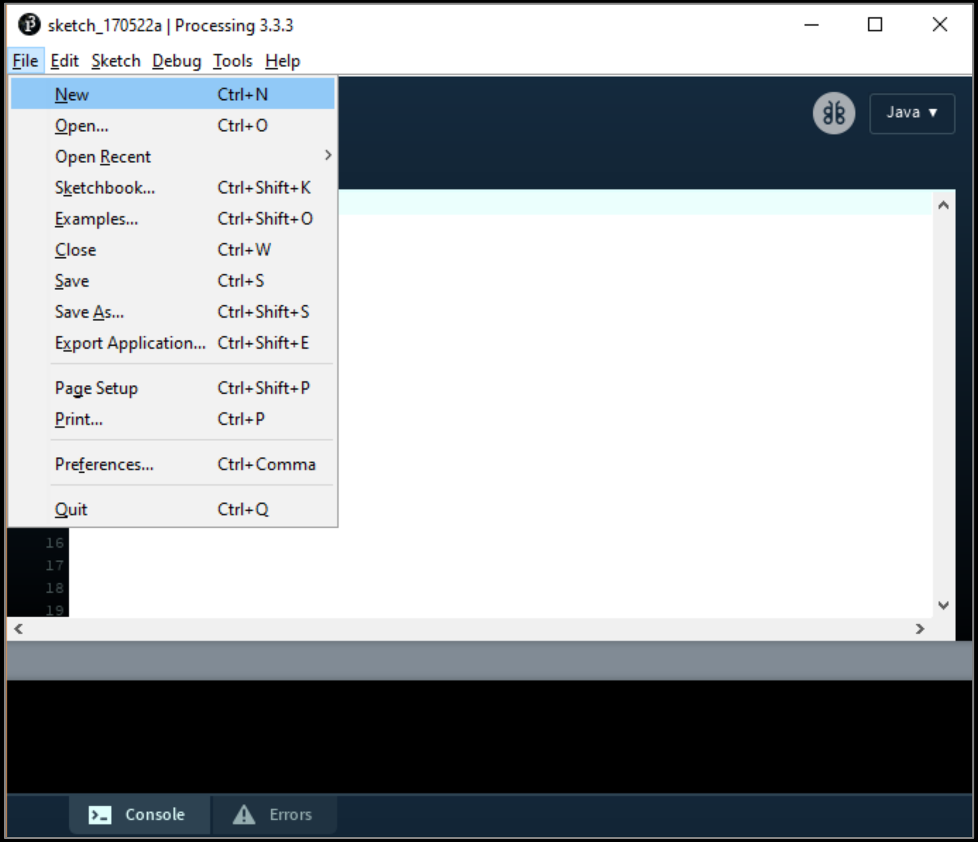
Your code goes into the *Text Editor* here

Console and Errors Window

The code represents a sketch. Each sketch is actually a subclass of the `PApplet` Java class

PDE: Creating and Running a Sketch

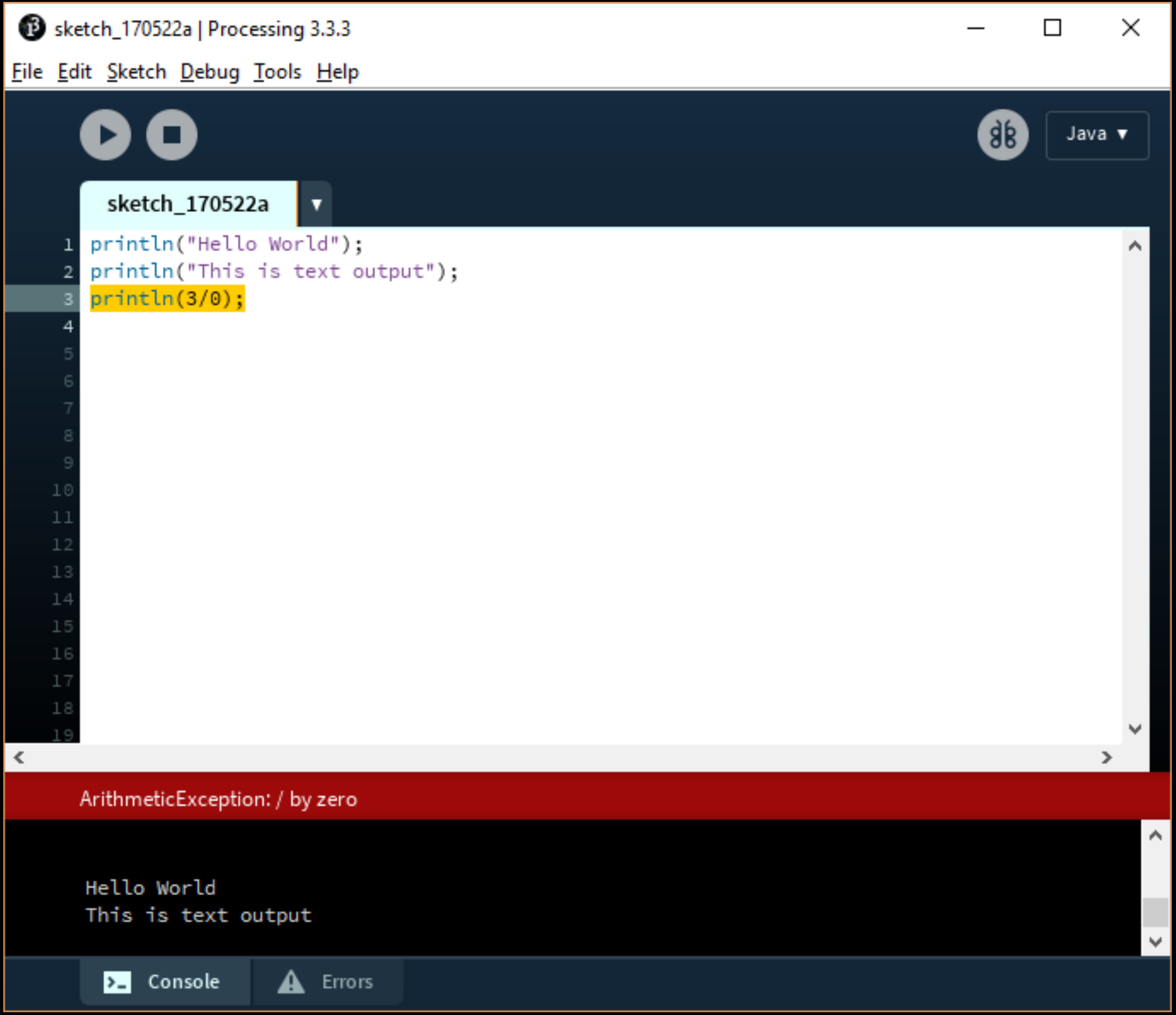
- To create a program code file, select `File->New` or
- Your new program is called a ***sketch*** in Processing. Sketches are saved in a folder on your computer called ***sketchbook***.
- To write your code, start typing in the Text Editor" area of the PDE.
- Use the buttons ***Run*** and ***Stop*** on the toolbar to run or terminate your program.



PDE: The Console Window

- The console window displays
  1) Text output, e.g. when printing text using `print()` and `println()` functions.

  2) Error messages


sketch_170522a | Processing 3.3.3
File Edit Sketch Debug Tools Help
Java
sketch_170522a
Console
Errors

```
println("Hello World");
println("This is text output");
println(3/0);
```

ArithmeticException: / by zero

```
Hello World
This is text output
```

Functions

- A ***function*** is a sequence of statements that performs a specific action.
  - Creating a function avoids repeating statements and allows for better code organization.
- A function must have a name. Whenever we want to perform the function's action, we need to call (invoke) the function by its name.
  - For example, to print something on the console, we write

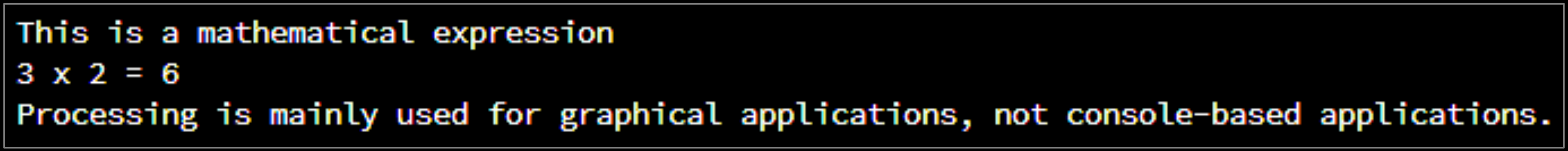
```
println("Hello World");
```

- *Processing* comes with a library of **predefined functions** that may be used to perform different actions such as drawing shapes. To use these functions, you need to call their names with the appropriate parameters.
  - In Java, a function is also called a "method".

Exercise

Output Text to the Console

- Use `print()` and `println()` to display the following text on the console. Note that the number 6 on the second line is computed as 3*2.

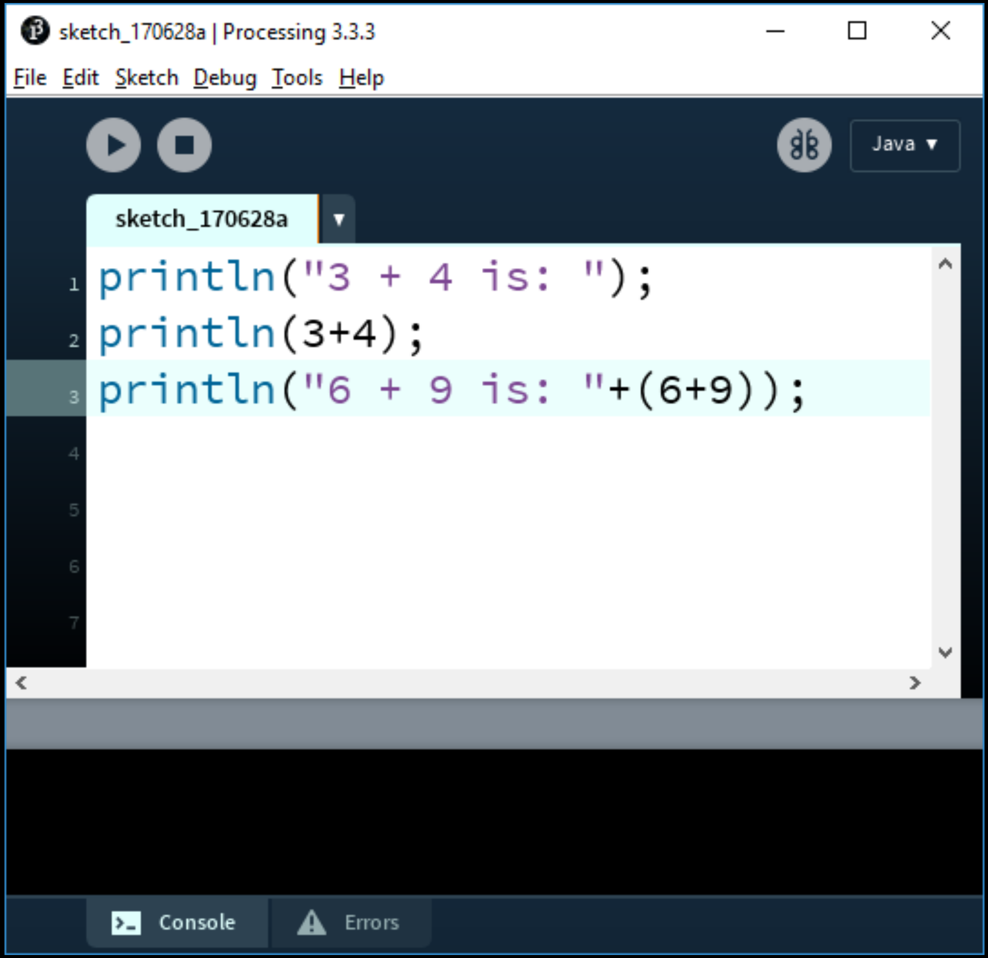


```
This is a mathematical expression
3 x 2 = 6
Processing is mainly used for graphical applications, not console-based applications.
```

Exercise

Output Text to the Console

- What is the output of this program? Explain.



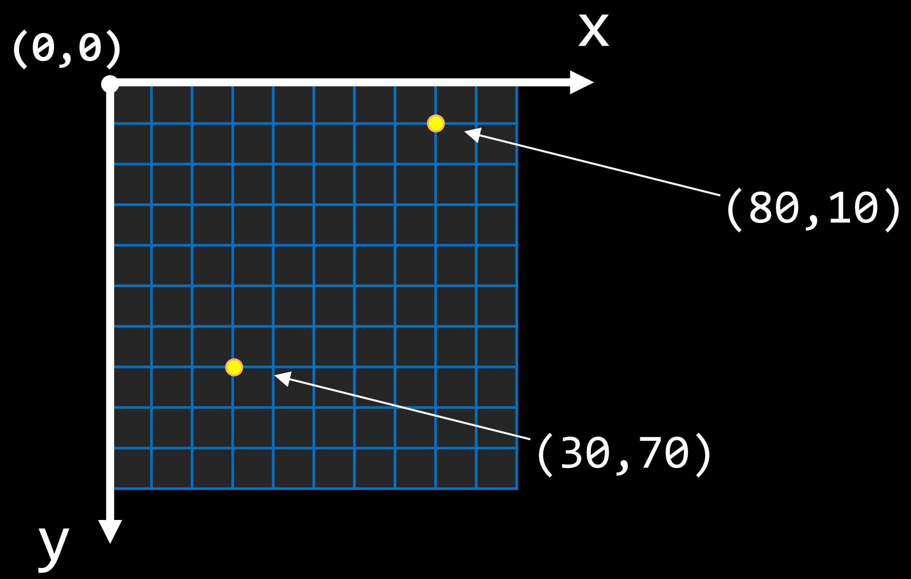
sketch_170628a | Processing 3.3.3
File Edit Sketch Debug Tools Help
Java
sketch_170628a
Console
Errors

```
println("3 + 4 is: ");
println(3+4);
println("6 + 9 is: "+(6+9));
```

2D Coordinate System

The Coordinate System

- Drawing on the screen is done by specifying coordinates which refer to a location on the screen.
- By default
  - ***origin*** is the upper-left hand corner of the screen.
  - x coordinate is horizontal, getting bigger as we move right.
  - y coordinate is vertical, getting bigger as we move down.



Question

Coordinate system

Assume we have the 100x100 sketch shown below. Each small square is 10x10 pixels. What is the (x , y) location of the point?

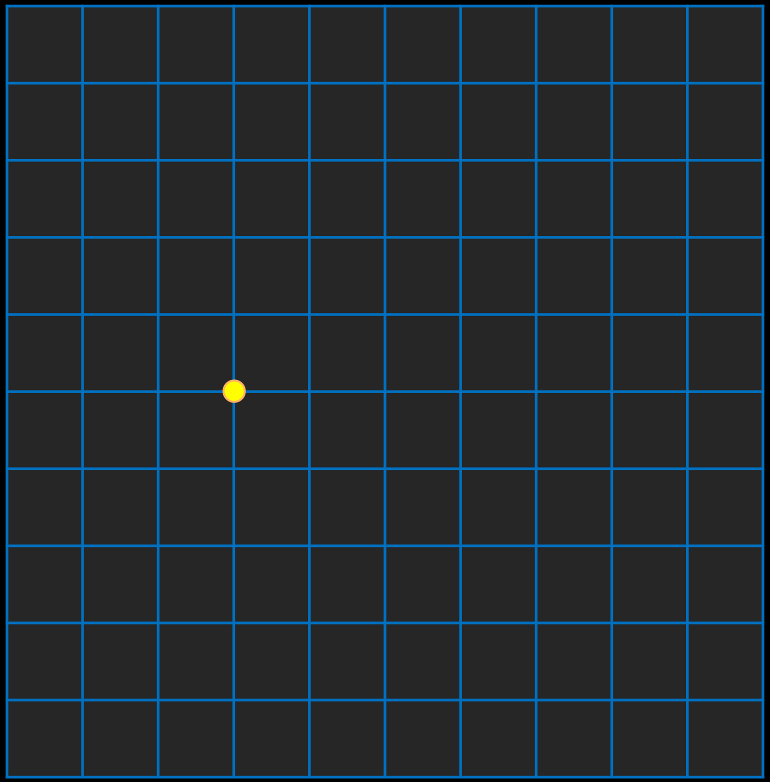
A. (30, 50)

B. (50, 30)

C. (3, 5)

D. (5, 3)

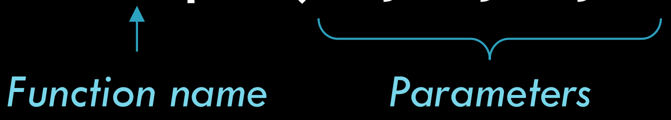
E. None of the above



Drawing Primitive Shapes

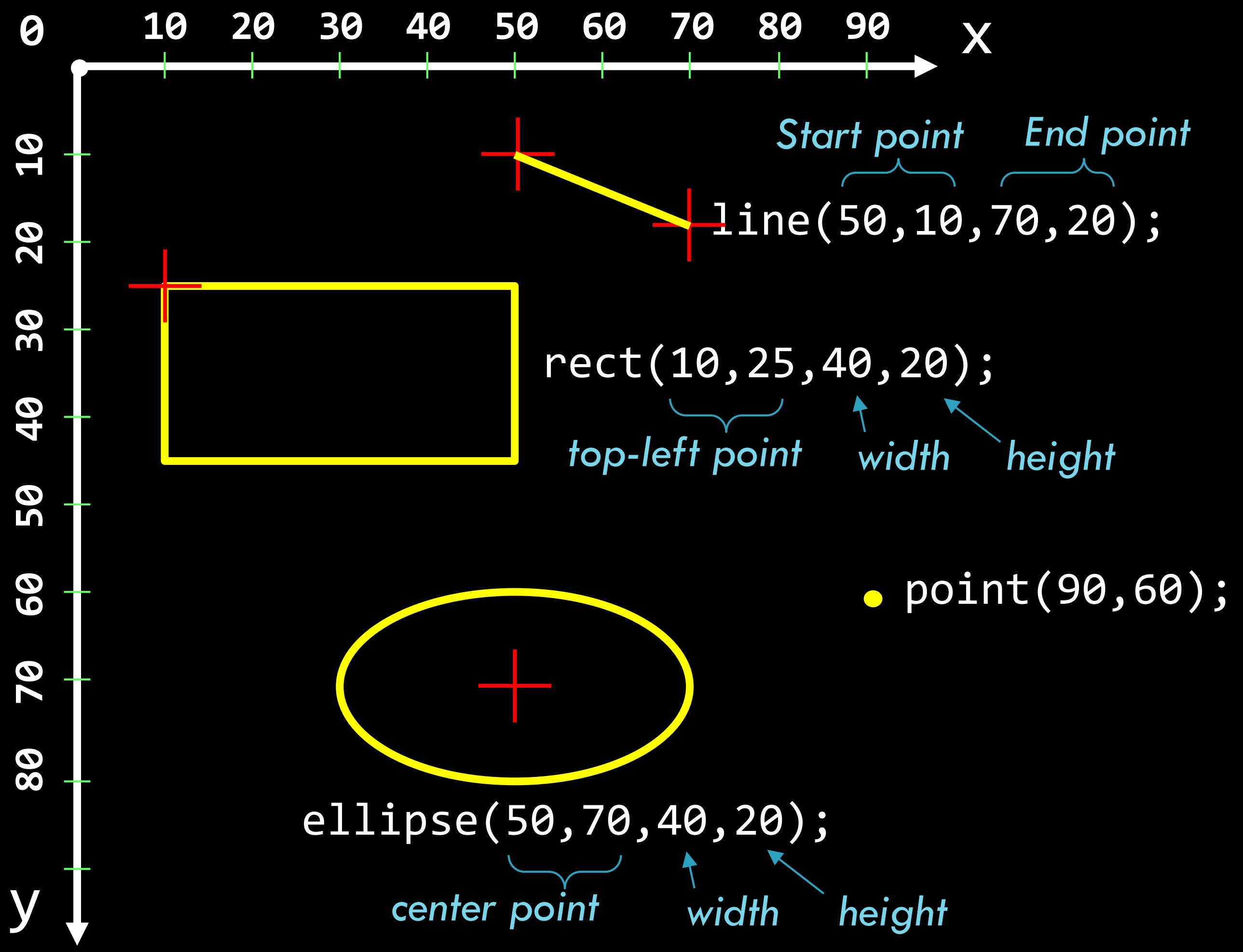
- To draw shapes on the screen, we call the function that represent each shape with arguments representing the shape dimensions.

- Example of primitive shapes
  - Point: `point(90,60);`
  - Line: `line(50,10,70,20);`
  - Rectangle: `rect(10,25,40,20);`
  - Ellipse: `ellipse(50,70,40,20);`



Function name *Parameters*

Drawing Primitive Shapes, *cont'd*



```
line(50,10,70,20);
```

Start point *End point*

```
rect(10,25,40,20);
```

top-left point *width* *height*

```
point(90,60);
```

```
ellipse(50,70,40,20);
```

center point *width* *height*

Drawing Primitive Shapes, *cont’d*

- Here is the Processing code and output

```
// draw the shapes
line(50,10,70,20);
rect(10,25,40,20);
point(90,60);
ellipse(50,70,40,20);
```