**Objectives**

1. Create a parser for arbitrary polynomials
   1. Pre-processing
      1. Handle implicit multiplication (regex)
      2. Write powers with exponentiation operator
   2. Conversion from infix to postfix notation
2. Generate a list of points for the graph
   1. Define interval
   2. Decide frequency of points
   3. Calculate x values through postfix stack
3. Create a mock-up display for testing purposes
   1. Create Pygame display compatible with screen (320x480px)
   2. Draw axes with defined interval
   3. Plot points
   4. Interpolate between points
4. Add translation functions
   1. Add zoom function
      1. Scale axes by factor centred on middle of screen
      2. Recalculate points with new interval
      3. Replot and interpolate points
   2. Add move function
      1. Change interval and recalculate points
      2. Add reset button
5. Extend parser to other Cartesian equations
   1. Radical equations
      1. Parse roots with exponentiation operator
   2. Exponential equations
      1. Define constants (e, pi)
   3. Reciprocal graphs
      1. Parse fractions as division operator
      2. Handle asymptotes
         1. Draw dashed line OR approximate continuously
   4. Logarithmic equations
      1. Parse logarithms as single character, allow for bases
   5. Trigonometric equations
      1. Parse main functions
         1. Handle tan asymptote
      2. Parse reciprocal functions
         1. Handle asymptotes
6. Extend parser to parametric equations
   1. Allow for inputs of explicit y(t) and x(t)
   2. Determine bounds for t for given interval
   3. Parse each as Cartesian equations
7. Extend parser to calculus
   1. Differentiation
      1. Calculate x values for interval
      2. Approximate central difference
      3. Handle asymptotes (high gradient)
   2. Integration
      1. Calculate x values for interval
      2. Use trapezium rule to get area of slices
      3. Divide by strip width to plot y values
      4. Handle asymptotes (high area)
8. Add support for multiple equations
   1. Add multiple input option with colour key
   2. Plot both graphs (different colours)
9. Testing
   1. Check basic test cases for all types of equation
   2. Check test cases for multiple types combined
   3. Test cases where calculus doesn’t work well
10. Design physical calculator
    1. Make list of buttons from what I was able to implement
    2. Arrange buttons in a layout
    3. Design electronics
       1. Create schematic with battery connected to raspberry pi
       2. Connect buttons to raspberry pi in schematic
       3. Connect rotary encoder for zoom OR use touchscreen
          1. Order screen to test code
       4. Arrange components on PCB
       5. Add solar panel
          1. Connect to solar power manager and rechargeable battery
       6. Route and wire PCB
    4. CAD design
       1. Design case
       2. Design buttons
11. Build physical calculator
    1. Create BOM
    2. Order PCB
    3. Order other electronics
    4. 3D print CAD parts
    5. Solder remaining components to PCB
    6. Run program on raspberry pi
       1. Check power supply
       2. Load and run on boot
       3. Scale Pygame for the screen
       4. Change code for touchscreen if required
    7. Assemble the calculator