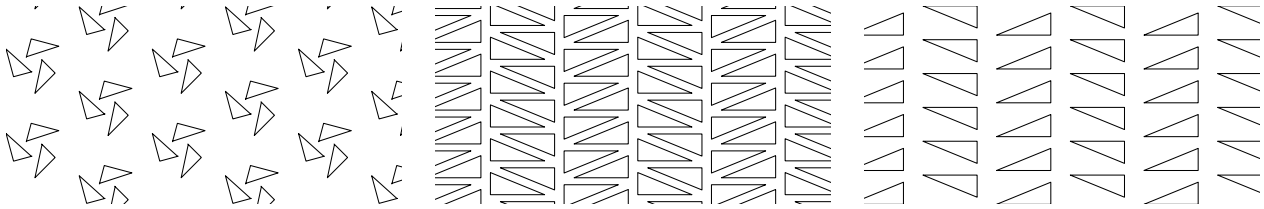
Practical 2: Hand in your solution to MyUni before Fri 16th March at 1pm.

You should read the practical, and prepare before the actual session.

1. - Classify the symmetries of the following three pictures.



   - Using the flow chart at http://mathstat.slu.edu/escher/index.php/Wallpaper_Patterns#Wallpaper_Flow_Chart identify the name of each example.
   - Read a little more about the naming of these groups, and draw an example of *p6m*.

2. Write MATLAB code to draw (and fill in with colour) a polygon.

   - Start MATLAB
   - You should use MATLAB's help command to learn about `fill`, *e.g.,* type

     ```
     help fill
     ```

   - You need to specify a polygon by the x and y coordinates of its vertices.
   - Specify the *colour* using an R-G-B triple
   - *e.g.,* to draw a unit square

     ```
     X = [0.0, 1.0, 1.0, 0.0]
     Y = [0.0, 0.0, 1.0, 1.0]
     C = [0.0, 0.0, 1.0]  % the colour blue
     figure(1)            % create a new figure
     fill(X, Y, C)
     xlim([-0.2, 1.2])   % these commands just extend the axes
     ylim([-0.2, 1.2])   % so we can see better
     axis square         % square up the axes
     ```

   - Draw a hexagon using a modification of this code.

3. Now iterate this code, so that it draws a tessellation built of hexagons over some region

   - You will want to use iterators like: `for i=1:10`
   - You can *nest* iterators inside each other
   - You will have to assign contrasting colours to polygons so that you can see them.
   - If you use the command

     ```
     fill(X, Y, C, 'FaceAlpha', 0.2)
     ```

     the fill will be partially transparent, so you can easily see overlaps.
   - You can find an example of this code in *protected* form on the web page.

4. **Now, use the above to generate a figure, and include this figure into a LaTeX document using Overleaf. Hand in the resulting PDF through MyUni.**

5. *(Extension question)* Write Matlab code that will list the symmetries of a given polygon.

   - define a *function* called `test_sym` that takes as input x and y coordinates of the vertices of the polygon, and which returns
     - the axes of reflectional symmetry (if there are any)
     - the angles and centres of rotational symmetry (if there are any)
   - The prototype of your function should look something like

     ```
     [reflectional, rotational] = function(X, Y)
     ```

   - Assume each vertex is distinct, and that the polygon is simple, but not necessarily convex.
   - Be aware of floating point errors.
   - Other hints
     - you only really need to look at the vertices
     - how would you test if it is a regular polygon?
     - if it's regular what symmetries would it have?
     - if it's irregular, what symmetries could it have?
     - given a particular symmetry, could you write a set of equations that would have to hold?