

moodle.port.ac.uk

1

cycling continuously through four colours (red, green, blue and magenta in the example above) as we move from left-to-right along each row, and from the rightmost patch of each row to the leftmost patch of the next.

Each patch features a regular geometric design made up of lines, circles, rectangles and/or polygons and has dimensions of  $100 \times 100$  pixels. The two patch designs and the layout of patches are not necessarily as given in the sample above. They are determined by the *final three digits of your 6-digit student number*, and are displayed in the tables on the final two pages. The layout of the patch designs is given by the antepenultimate (fourth) digit of your student number. The two patch designs are given by the penultimate (fifth) and final (sixth) digits of your student number. The patch designs and arrangement shown in the figure above would be used if your student number was, for example, 777032. It's important that your program draws the patch designs accurately, and that it draws the correct designs with the correct arrangement – you will receive no credit for drawing the wrong patch designs or the incorrect arrangement.

Your program should draw the patches using the facilities provided in the graphics module (Line, Circle etc.), and must not use bitmapped images. The designs are intended to test algorithm development skills (e.g. they should involve the use of one or more for loops). For some of the designs, it will be useful to remember that shapes drawn later appear on top of those drawn earlier.

## Main program requirements

Your program should begin by prompting the user to enter:

- the patchwork size (i.e. the common width & height in terms of patches);
- the desired four colours (which may contain repeated colours).

The program's user interface should be friendly and robust; e.g., on entering invalid data, the user should be re-prompted until the entered data is valid. (Valid sizes are 5, 7 and 9, and valid colours are red, green, blue, yellow, magenta and cyan.) Once these details have been entered, the patchwork sample should be drawn in a graphics window of the appropriate size. For example, if the user enters size 5, and colours red, green, blue and magenta, then (in the case that your student number ends in 032) the sample shown above should be drawn in a graphics window of width 500 pixels and height 500 pixels.

## Advanced program feature

The above requirements are what I expect most students to attempt, and carry the vast majority of the marks for functionality. If you would like a further challenge for a few additional marks, then I encourage you to attempt this additional feature.

After the patchwork design has been drawn, you should allow the user to change the colour of patches by clicking on them with the mouse. If the user clicks on a patch, then the colour of that patch should change to the next colour in the inputted colour sequence. (For example, if the user clicks on a blue patch in the figure above, it would turn magenta, and if they click a magenta patch it would turn red.) The user should be able to make as many colour changes to as many patches as they wish.

## Moodle Submission

You should submit your program via the unit's Moodle site by the deadline specified above. Make sure that your program file is named using your student number, and that it has a .py suffix; for example, 123456.py. Click on the link labelled Python Assignment Submission and upload your program. If your submission is late, then your mark will be capped under University rules. Do not leave it until the last minute before submitting—if Moodle reports a late submission then your mark will be capped. Also, if you re-submit after the deadline then the mark will be capped even if you made a first submission before the deadline.

## Demonstration

You need to demonstrate your program to a member of staff in your scheduled practical class during the week beginning 8th December. We will execute your submitted program, and we will ask you questions about how you wrote it and how it works. All the marks for the assignment will be awarded during the demonstration, so you must attend: failure to attend the demonstration will result in zero marks, and demonstrating your program late will result in your mark for the assignment being capped under University rules. If you wish to organise a late demonstration outside a practical session, please email me.

Formal written feedback and your assignment mark will be sent to you via email immediately after your demonstration has been completed. If you do not receive this email, then your mark may not have been recorded and it is your responsibility to inform me if this happens.

### Functionality [30 marks]

In the demonstration we will first assess the **completeness** and **correctness** of the operation of your program, and the **quality** and **robustness** of its user interface. The main program requirements will carry **25 marks**, and the advanced (optional) feature will carry **5 marks**.

### Program code quality [10 marks]

After demonstrating the program's functionality, the member of staff will give you some feedback on the quality of your program code. Your program will be awarded marks based on: (i) its overall structure (how well it has been designed using the principles of top-down design—see lectures P14-15); (ii) its readability (see lecture P05); and (iii) the quality of the algorithms used (e.g. the control structures it employs to draw the patches). Even if your program works well, the code may obtain very few marks if its readability is poor.

## General advice

The most important advice is to ***start early and do not leave finishing the work until just before the deadline***. Your work will almost always suffer if you leave it until too late. Furthermore, technical problems are likely to be overcome if encountered early, and do not usually constitute an acceptable reason for lateness. Make sure that your solution employs good, uncomplicated, algorithms. Often, repetitive code is a sign of poor algorithm design (e.g. don't use 25 lines of code to draw 25 circles!).

If you find the task very difficult, remember that you do not have to provide a complete

solution to achieve a pass mark. Make sure that your program executes and gives some graphical output, and that you demonstrate what your program does. To make things easier, you might choose to write a program which, for example:

- draws a patchwork sample containing just one of the patch designs;
- attempts to draw both patch designs but not in the correct arrangement;
- ignores colours.

If you don't know how to start, it is recommended that you see me or go to the Tutor Centre or Learning Support Tutor for some advice as soon as possible.

## Hint

If well designed, your program will consist of a few functions including a `main` function and two patch-drawing functions. (In a good solution there will be other functions.) Each patch drawing function will need parameters representing the graphics window on which to draw the patch, the  $x$ - and  $y$ -coordinates of the top-left corner of the patch, and the patch colour. Make sure that you have completed exercises 9 and 10 on worksheet P06, as well as the first few exercises on worksheet P08, before attempting the coursework.

## Support

Any queries should be addressed to me ([Matthew.Poole@port.ac.uk](mailto:Matthew.Poole@port.ac.uk)). Any points that come to light that may be of general interest will be answered on Moodle, so please check there before asking questions of a general nature. Finally, the Tutor Centre and Xia Han (the Learning Support Tutor) remain a good source of advice on programming issues.

## Important

**This is individual coursework, and so the work you demonstrate and submit for assessment must be your own. Any attempt to pass off somebody else's work as your own, or unfair collaboration, is plagiarism, which is a serious academic offence. Any suspected cases of plagiarism will be dealt with in accordance with University regulations.**

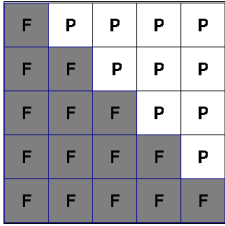
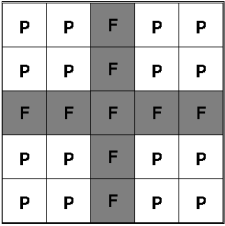
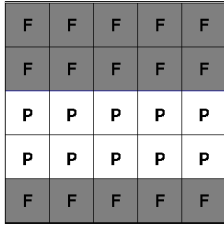
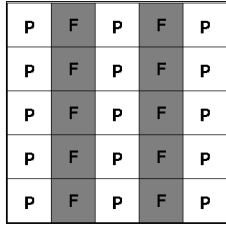
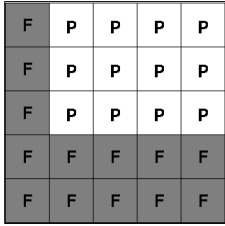
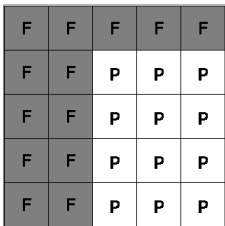
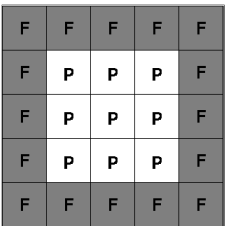
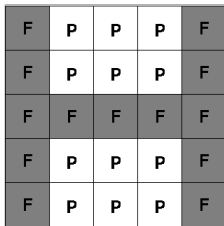
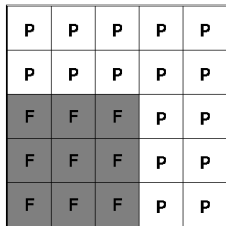
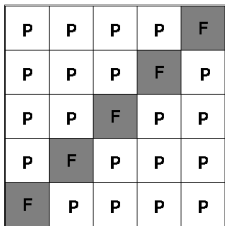
Matthew Poole  
November 2014

## Patchwork layout and patch designs

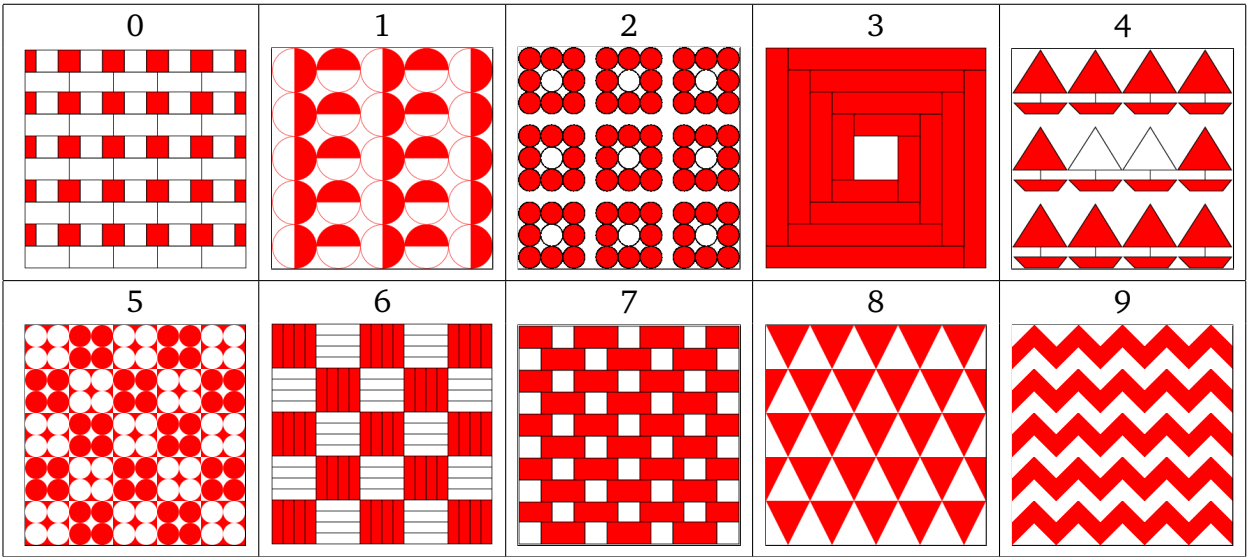
Make sure that your program draws patchworks determined by the final three digits of your student number. The patch colour shown on the next page is red (but will in general depend on the user's inputs). The background can be left grey or, if you prefer, white. Note the colour of the outline of the shapes – sometimes this is the patch colour, sometimes it is black. I am not concerned too much if the edges of patches “collide” with other patches or the edge of the window by one pixel. If you wish, you can draw black borders around patches in order to separate them, but this is not necessary.

### Antepenultimate (fourth) digit of student number

This table describes where patches given by the penultimate (P) digit and final (F) digit of your student number should be drawn on your patchwork. The text in the table states where the final (F) digit patch should be drawn, and the diagram illustrates the case for a  $5 \times 5$  patchwork. Together, the text and example should make clear what the arrangement would be for a  $7 \times 7$  or  $9 \times 9$  patchwork, but please ask me if you are unsure.

<p>0 bottom-left triangle</p> 	<p>1 middle row &amp; column</p> 	<p>2 top two &amp; bottom rows</p> 	<p>3 even-numbered columns</p> 	<p>4 left column &amp; bottom 2 rows</p> 
<p>5 top row &amp; 2 left columns</p> 	<p>6 around border</p> 	<p>7 left/right columns &amp; middle row</p> 	<p>8 bottom left 3 x 3 square</p> 	<p>9 diagonal from bottom left</p> 

Penultimate digit of student number



Final digit of student number

Note that patch design 4 in this table is made up of 20 straight lines (there are no curved lines).

