U30299 – Programming

moodle.port.ac.uk

Python Coursework: A Patchwork Sampler

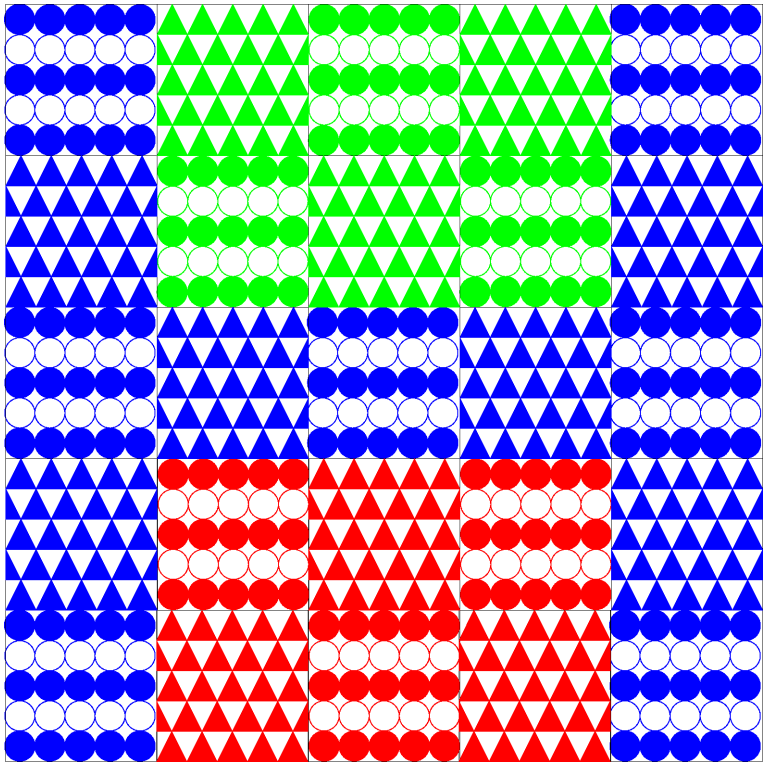
Introduction

This coursework assignment is designed to give you practice in applying all of the main programming concepts you've seen in the module so far to solve a larger and more complex problem. The assignment will be marked out of 50 and carries 25% of the module marks.

You need to submit your program via the module's Moodle site by the deadline of ***11.00pm, Friday 10th January 2020***, and are required to ***demonstrate*** your submitted program in your 2-hour Programming class timetabled during the week beginning ***13th January 2020***. Study this handout thoroughly in order to understand exactly what is expected from you for the coursework.

Your task

Your task is to write a program to display patchwork samples, an example of which is illustrated below. The actual patchworks your program will display will depend on your student number and on the user's inputs.



A patchwork sample is made up of patches of two different designs and up to three colours. Patchworks are square and can be of three different sizes: 5×5 patches, 7×7 patches and 9×9 patches. Each patch features a regular geometric design made up of lines, circles, rectangles and/or polygons and has dimensions of 100×100 pixels. The two patch designs and the layout and colouring 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 and colouring 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. For example, if your student number was 777**838**, the patch designs and patch arrangement for a 5×5 patchwork with colours blue, green and red are those illustrated on page 1.

It's important that your program draws the patch designs accurately, and that it draws the correct designs, layout and colouring – you will receive no credit for drawing the wrong patch designs or patch 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. You should not use parts of the Python language which we haven't covered in this part of the module; for example, do not use exception handling and do not define your own classes.

Main program requirements

Your program should begin by prompting the user, using a text (shell)-based interface, to enter:

- the patchwork size (i.e. the common width & height in terms of patches);
- the initial letter of each of the desired three colours (e.g. 'r' for 'red'); users are allowed to choose any colour more than once.

The program's user interface should be easy to use, helpful and robust; e.g., on entering invalid data, the user should be given appropriate feedback and re-prompted until the entered data is valid. (Valid sizes are 5, 7 and 9, and valid colours are red, green, blue, magenta, orange 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 blue (b), then green (g) and finally red (r), then (in the case that your student number ends in 838) the patchwork shown on page 1 should be drawn in a graphics window of width 500 pixels and height 500 pixels.

Challenge 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 edit the patchwork in the following way. The user should be able to choose various options by pressing particular keys (on the window, not the shell):

- 's' should enter ***selection mode***. In this mode the user should be able to select any number of patches by clicking on them with the mouse; selected patches should be displayed with thick black borders. Selecting a currently selected patch should deselect it. Whilst in selection mode a 20×20 pixel black-filled rectangle should appear in the top-left corner of the window; clicking in this box should exit selection mode (without selecting/deselecting the top-left patch) and the box should disappear.
- 'd' should deselect all selected patches.
- 'p' should change all selected patches to the 'penultimate' digit design, keeping their colours the same.

- ‘f’ should change all selected patches to the ‘final’ digit design, keeping their colours the same.
- The initial letter of any valid colour (‘r’, ‘g’, ‘b’, ‘m’, ‘o’ or ‘c’) should change all selected patches to that colour, keeping their designs the same.
- all other should keys have no effect.

(Note that none of these key options should work whilst in selection mode—in selection mode the user just uses the mouse; and when not in selection mode mouse clicks should have no effect.) For example, to re-colour two patches cyan, the user will press ‘s’ (to enter selection mode), click on the two patches (to select them), click in the black box in the top-left hand corner (to exit selection mode), press ‘c’ to colour the patches cyan, then press ‘d’ to deselect the patches. The user should be able to repeatedly choose these options to make as many edits as they wish.

Ideally, the above operations should actually remove and recreate, or recolour, the graphics objects that make up patch designs, rather than drawing new objects on top of existing ones.

Moodle Submission

You should submit your program via the module’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 Programming session timetabled during the week beginning 13th January. 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 timetabled session, please email me—you must have given your demonstration by the end of January or you will receive a mark of zero.

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 [40 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 ***32 marks***, and the challenge (optional) feature will carry ***8 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). Make sure that your program uses 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!). Even if your program appears to work well, the code may obtain very few marks if it is poorly written.

General advice

Most importantly, ***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.

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, Nadim Bakhshov or Xia Han 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

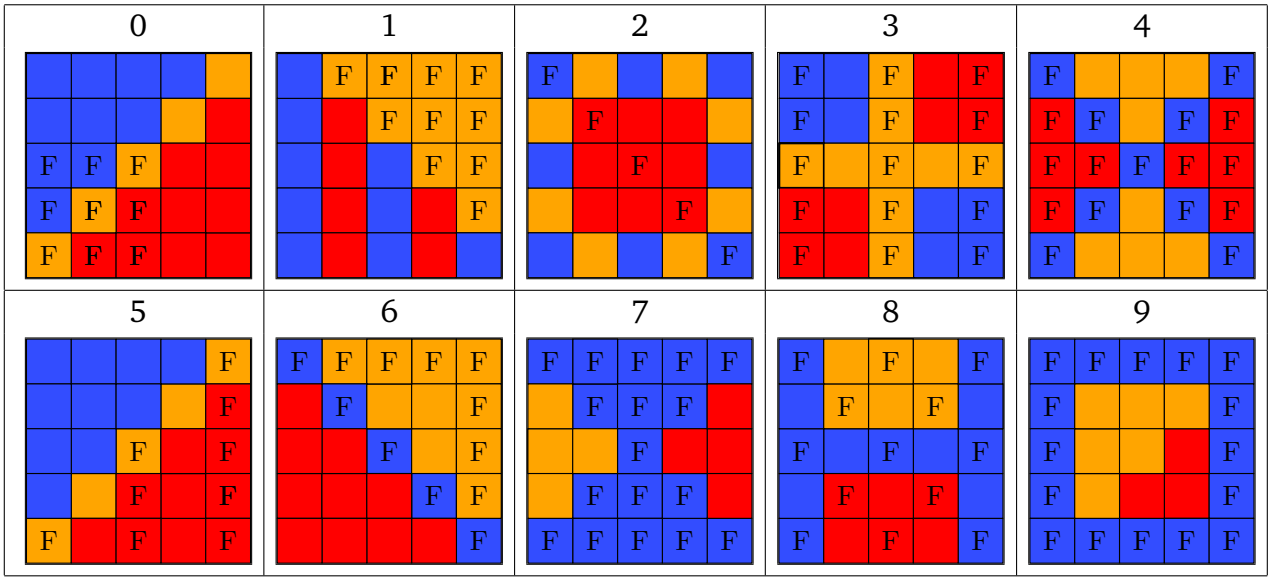
Queries should be addressed to me (`Matthew.Poole@port.ac.uk`). Any points that come to light that may be of general interest will be communicated via email, so please check this before asking questions of a general nature. The last few practicals this term can be used for help on the coursework. The Learning Support Tutor, Xia Han, is also able to give advice on the coursework.

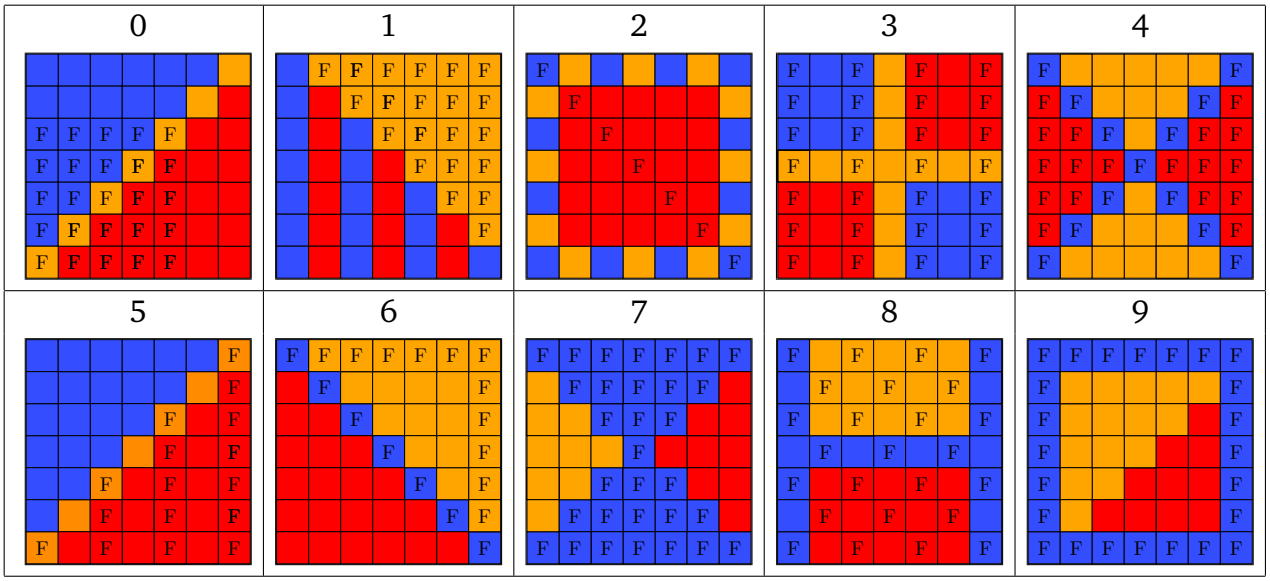
Patchwork layout and patch designs

Make sure that your program draws patchworks determined by the final three digits of your student number. The patch colour shown on the final 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

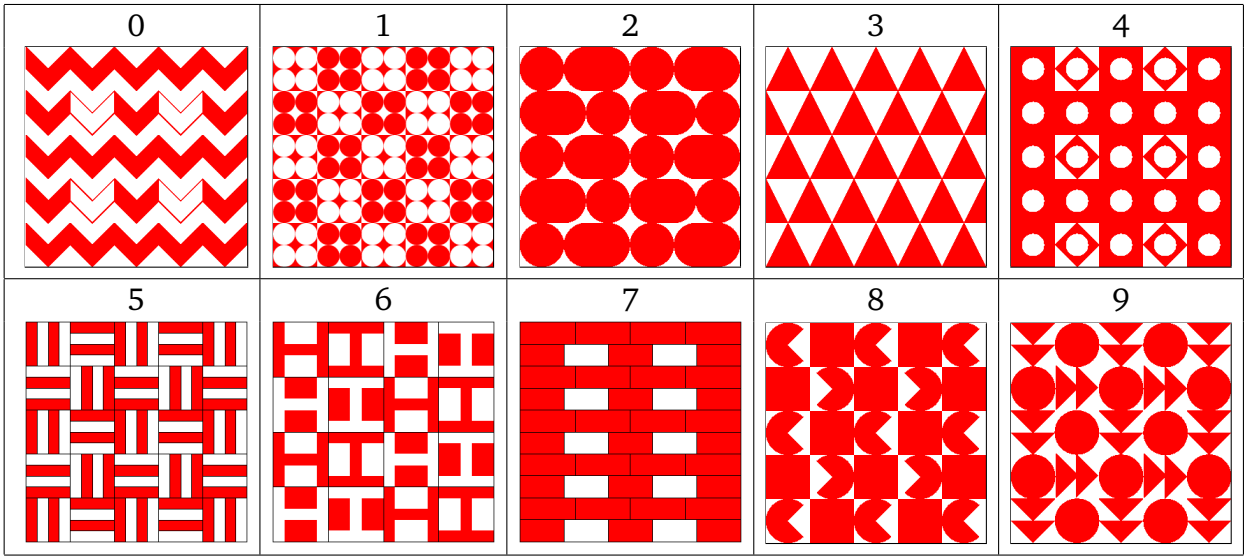
The tables below describe how patches should be arranged and coloured as determined by the fourth digit of your student number in the case of 5×5 and 7×7 patchworks (from these you should be able to determine the layout for a 9×9 patchwork). The tables assume that the first chosen colour is blue (the user entered b), the second is orange (o) and the third colour is red (r). It is important that the order of the inputted colours is used correctly. Notice that the top-left patch always takes the first inputted colour, and the next colour you see if you scan from left-to-right starting from this patch (continuing scanning from the left of the next row if required) will be the second inputted colour. Patches marked 'F' are those corresponding to the final digit of your student number; blank patches are those given by the penultimate digit of your student number.





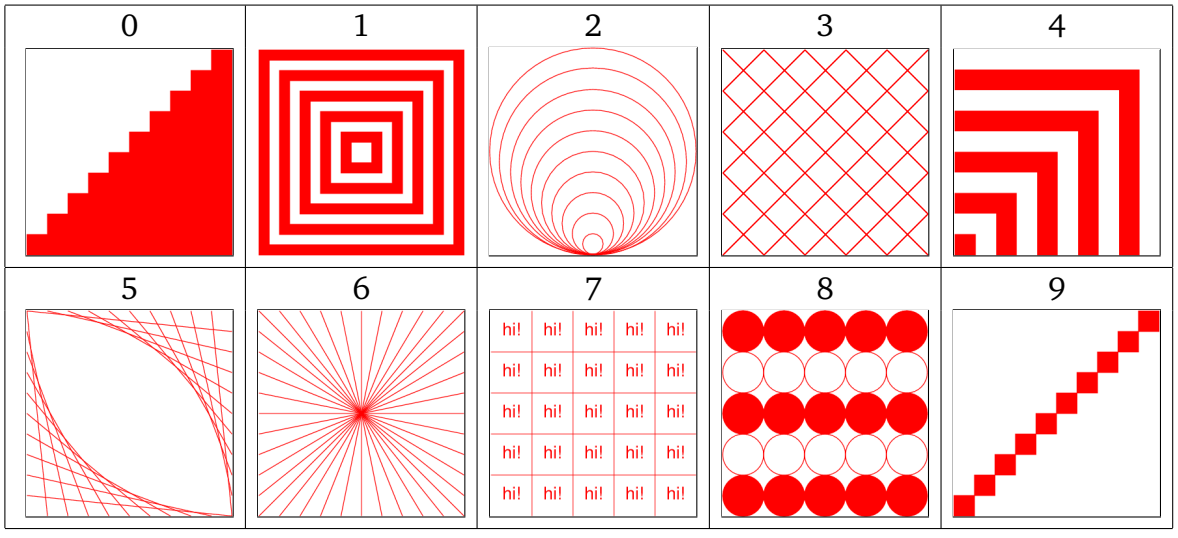
Penultimate digit of student number

Note that all coordinates should be multiples of 10 except in designs 1, 5, 6 and 7 where some are multiples of 5; the circles in designs 1 and 4 have radius 5.



Final digit of student number

Note that all coordinates should be multiples of 10 except in designs 1 and 2 where some are multiples of 5. Patch design 5 is made up of 20 straight lines (there are no curved lines).



Important

This is an 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 2019