School of Computing

Module Title and Code	PROGRAMMING 1 - M34698 - FHEQ_4
Module Coordinator Other lecturers	Xia Han – xia.han@port.ac.uk Nadim Bakhshov Mani Ghahremani
Assessment Item	Item 3
Assessment Title	Python Patchwork Assignment
Date Issued	2025-11-24

Schedule and Deliverables

Deliverable	Value	Format	Deadline / Date	Late deadline ECF deadline
Program	100% - all marks for your program will be awarded in the demo	A single .py file named using your student number (e.g. 2012345.py)	2025-12-10 1pm GMT	2026-01-07 1 pm GMT
Demo		Cloze exercise and 6-7 min demo with a member of staff	In your practical class on 2025-12-16 or 2025-12-18	

Notes and Advice

- The Extenuating Circumstances procedure is there to support you if you have had any circumstances (problems) that have been serious or significant enough to prevent you from attending, completing or submitting an assessment on time. If you complete an Extenuating Circumstances Form (ECF) for this assessment, it is important that you use the correct module code, item number and deadline (not the late deadline) given above.
- ASDAC are available to any students who disclose a disability or require additional support for their academic studies with a good set of resources on the ASDAC moodle site
- The University takes any form of academic misconduct (such as plagiarism or cheating) seriously, so please make sure your work is your own. Please ensure you adhere to our *Student Conduct Policy* and watch the video on Plagiarism.
- Any material included in your coursework should be fully cited and referenced in **APA 7** format. Detailed advice on referencing is available from the library, also see TECFAC 08 Plagiarism.
- Any material submitted that does not meet format or submission guidelines, or falls outside of the submission deadline could be subject to a cap on your overall result or disqualification entirely.
- If you need additional assistance, you can ask your personal tutor, student engagement officer ana.baker@port.ac.uk, academic tutor eleni.noussi@port.ac.uk or your lecturers.
- If you are concerned about your mental well-being, please contact our Wellbeing service.

M34698 Programming 1

Python Coursework: A Patchwork Maker

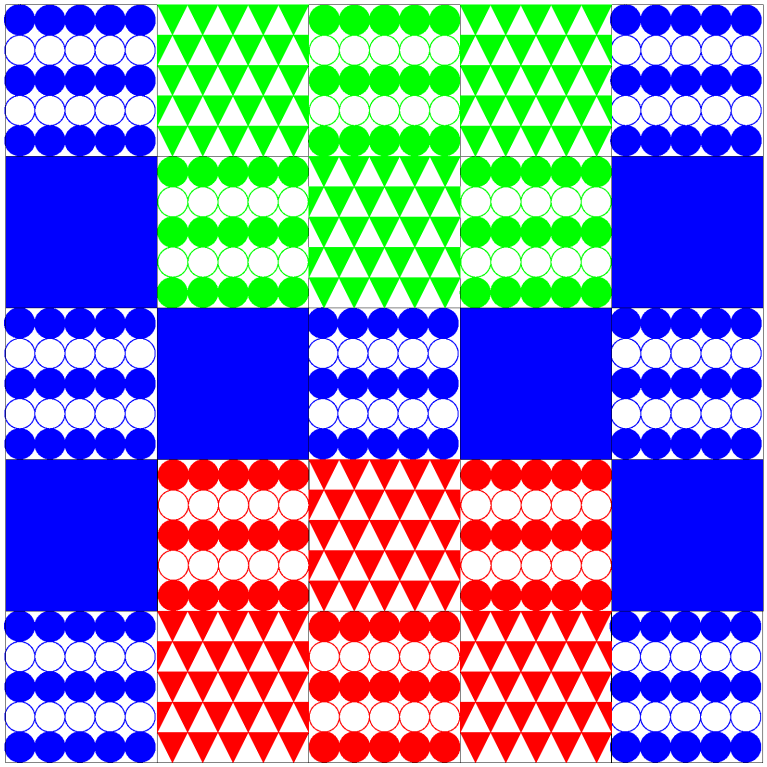
Introduction

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 40% of the module marks (it is assessment item 3 of the module).

You need to submit your program via the module's Moodle site by the deadline of ***1.00pm, Wednesday 10 December 2025***, and are required to ***demonstrate*** your submitted program in your 2-hour practical class timetabled on ***16th or 18th December 2025***. 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 patchworks, 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.



Patchworks are square and made up of square patches. A patchwork can be of three sizes: 5×5, 7×7 or 9×9 patches. Each patch can be one of two different designs or it can be completely plain. It can be one of three different colours, the background colour is always white, and some patches include shapes with blank outlines. Every 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 the patchwork

are not necessarily as given in the illustration above. They are determined by the ***final three digits of your student number***, and are displayed in the tables on the final two pages. The layout and colouring of the patchwork are given by the antepenultimate (third last) digit of your student number. The two patch designs are given by the penultimate and final digits of your student number. For example, if your student number was 2000**127**, 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 must draw the patchwork using the facilities provided in the graphix module (Line, Circle etc.); no other approach is permitted. 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 major parts of the Python language which we haven't yet covered in 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, in this order:

- the patchwork size (i.e. a single figure giving the common width & height in terms of patches);
- each of the desired three colours, entered one-by-one (separate prompts); the program should ensure that the colours are all different from each other.

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, pink, orange and purple.) Once these details have been entered, the patchwork should be drawn in a window of the appropriate size and with a white background. For example, if the user enters size 5, and colours blue, then green and finally red, then (in the case that your student number ends in 127) the patchwork shown on page 1 should be drawn in a 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, it should turn into a sliding puzzle, where the patches become tiles that can be moved around. Your program should follow these steps:

1. Remove the bottom-right tile (patch) to leave a blank space.
2. Ask the user (on the shell) how many tile moves should be made to shuffle the tiles.
3. The program should make this many tile moves by repeatedly selecting (at random) one of tiles immediately neighbouring the blank space (there will be always be two,

three or four such tiles), and sliding that tile to the blank space (leaving a blank space behind). The movement of the tile should be animated and should take about a second.

4. Once shuffled, the user should then be able to repeatedly click on the patchwork; when they click on a tile immediately neighbouring the blank space, the clicked tile should be moved to the blank space leaving a blank space behind (again, this movement should be animated). Clicking anywhere else on the patchwork should have no effect.

5. When the patchwork matches the initial design (from step 1) the program should report a "Well done!" message, and then ask the user if they would like to play again. If they do, then the process should be repeated from step 2; otherwise, the window should close. One difference with a normal sliding puzzle is that some tiles in the patchwork are identical. In your program it is only necessary for the player to match the design (layout and colours) to complete the puzzle—the actual final tile positions need not be exactly the same as they are after step 2.

The challenge feature will require you to make substantial use of lists (e.g. to store information about what kind of patch is at each position in the patchwork, and to store each patch's graphics objects). You may also find the sleep function of the time module helpful for the animation. Finally, for the challenge feature only, you may use Python dictionaries (dicts) and tuples.

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, 2012345.py. Click on the Item 3 link in the Assessments tab and upload your program. If you miss the deadline, you will need to upload it using the late submissions link – your mark will be capped according to University regulations unless you have a valid EC.

Demonstration & Mark Allocation

You need to demonstrate your program in your practical session on 16th or 18th December. You will first complete an exercise that tests your comprehension of your submitted code, and then a member of staff will mark your program and give you feedback. All the marks for the assignment will be awarded at the demonstration. You must attend and complete both parts; failure to do so will result in your work being recorded as zero (and possibly as a non-submission). Demonstrating your program late may result in your mark for the assignment being capped under University rules. Late demonstrations will take place in practicals during the first week of Teaching Block 2, and you must demonstrate your work by 30th January 2026 or it will be recorded as a non-submission.

Formal written feedback and your functionality and code quality marks 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 Xia if this happens.

Code comprehension [5 marks]

You will need to demonstrate your understanding of the code you have submitted by means of a computer-based cloze exercise, similar to those given in the group activities in the practical classes. The system will display your code with comments removed and with 10 symbols (keywords, variables, function names, operators, etc.) replaced by blanks. You will have 10 minutes to fill in these blanks with the correct symbols from your program. In the rare case that your submission is too short for the system to generate 10 blanks, then a smaller number will be presented and your mark for this exercise will be limited.

If your score for this cloze exercise leads the module team to suspect that you did not write the complete program yourself, your work will be subject to an academic misconduct process which may result in your assignment mark being reduced to zero.

Functionality [35 marks]

In the next part of the demonstration, a member of staff will 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 ***27 marks*** (7 marks for each of the two patch designs, 7 marks for the patchwork layout and colouration, and 6 marks for the user interface), 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 lecture 8); (ii) its readability (see lecture 3.2); and (iii) the quality of the algorithms used (e.g. the control structures it employs to draw the patchwork).

Make sure that your program uses good, uncomplicated, algorithms. Also, try to ensure that your program would require minimal changes if the requirements were changed so that more colours were included as being valid, and extra patchwork sizes (11, 13, ...) were allowed. 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.

Note that assessment of code quality will not apply to the challenge feature.

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 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, ask for help as early as possible (see the Support section).

Hint

If well designed, your program will consist of a few functions including a `main` function and three patch-drawing functions – one for each design and one for plain patches. (In a good solution there will be other functions.) Each patch drawing function might take parameters representing the 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 6, as well as the first few exercises on worksheet 8, before attempting the coursework.

Support

The last few practicals before the deadline can be used for getting help with the coursework. Academic Tutors Simon Jones and Eleni Noussi are also able to give advice on the coursework via one-to-one support sessions.

Other queries should be addressed to Xia by email. Any reported errors on this handout, or other common issues, will be communicated via the Python Patchwork Coursework Frequently Asked Questions document in the General section on Moodle, so please check there before asking a question.

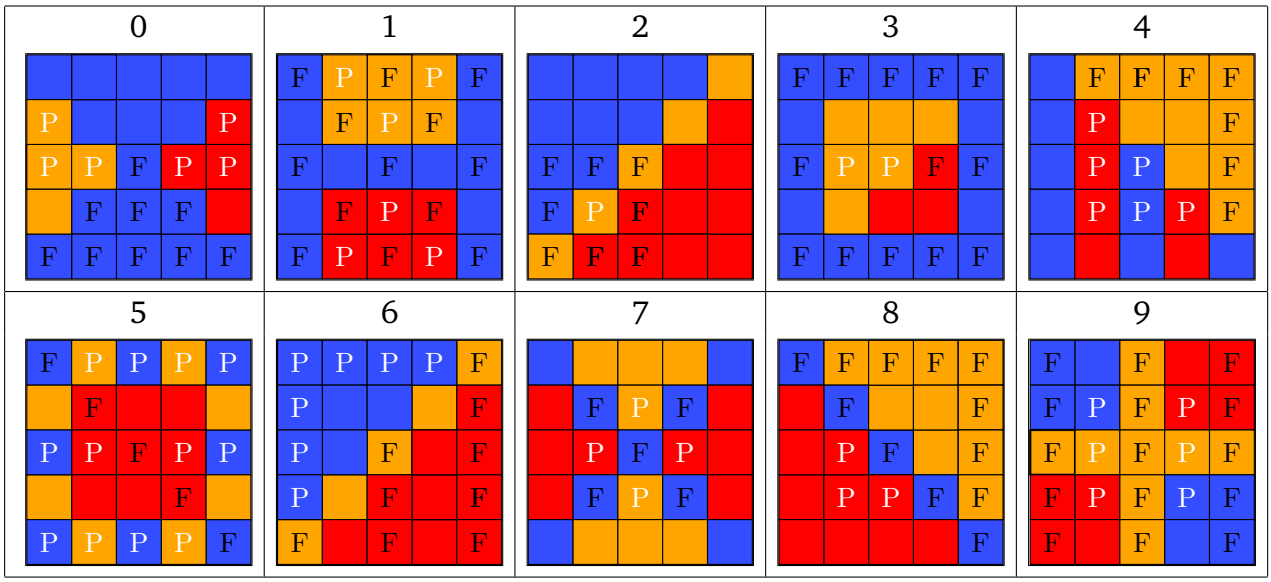
You are allowed to ask others for limited help, but you must write your own code. ***You should not, therefore, make use of AI assistance (ChatGPT, Gemini, Claude, Copilot Chat, Cursor, etc.)***. Remember that anything you submit must be your own work and you will need to demonstrate comprehension of your submitted code.

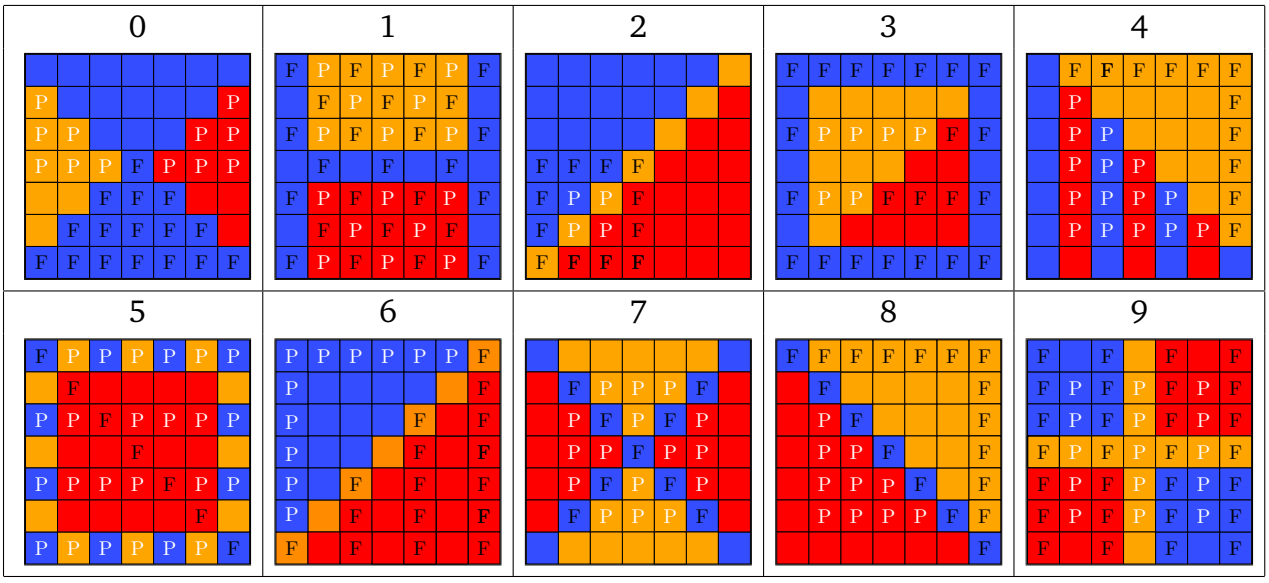
Patchwork layout and patch designs

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). Note the colour of the outline of the shapes – sometimes this is the patch colour, sometimes it is black. We are 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 digit of student number

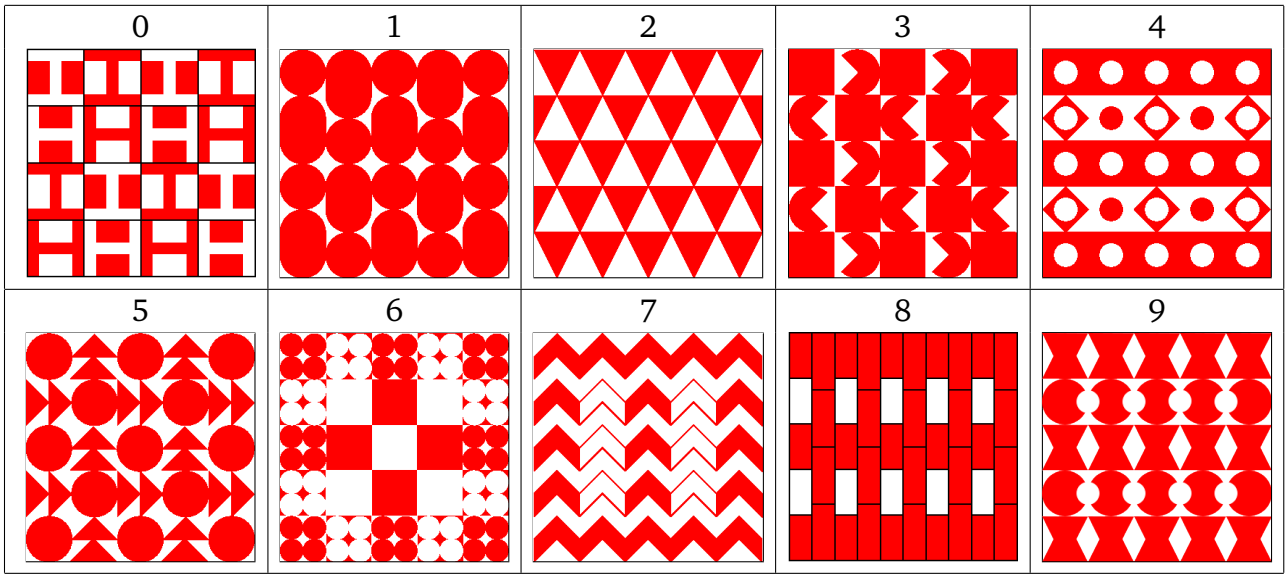
The tables below describe how patches should be arranged and coloured as determined by the antepenultimate digit of your student number for 5×5 and 7×7 patchworks. The tables assume that the first chosen colour is blue, the second is orange and the third colour is red. It is important that the order 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 labelled with a black 'F' should have the final digit design; patches labelled with a white 'P' should have the penultimate digit design; the unlabelled patches should be plain. Note that the arrangement and colouring is regular so you should be able to determine what it would look like for patchworks of size 9×9.





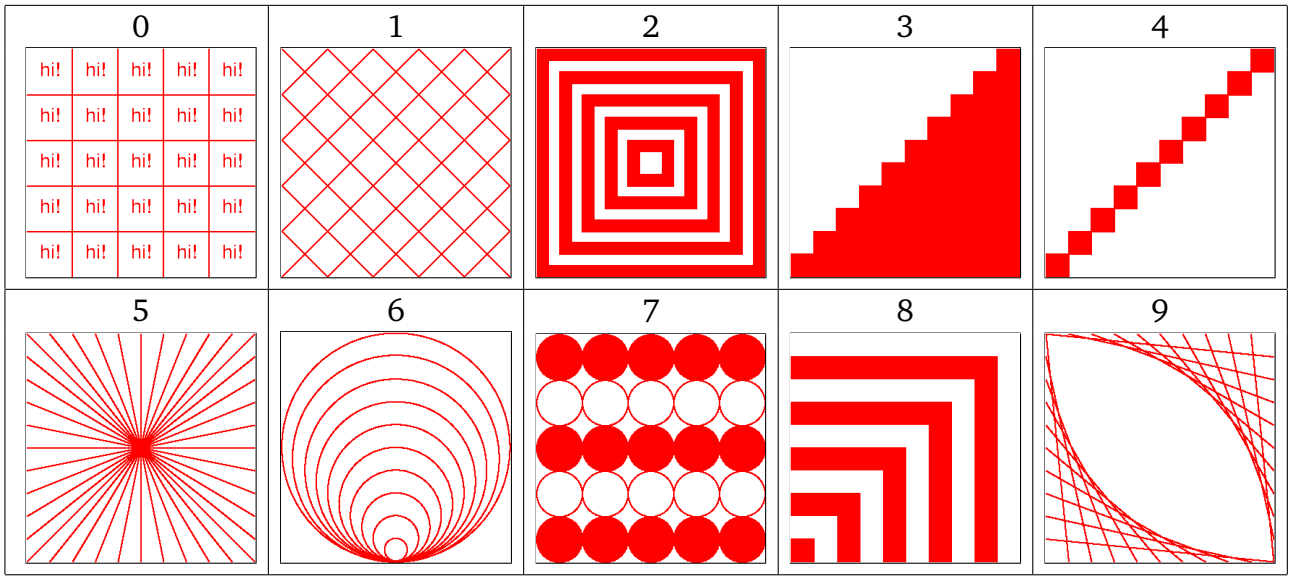
Penultimate digit of student number

Note that all coordinates should be multiples of 10 except in designs 0, 4, 6 and 9 where some are multiples of 5; the circles in designs 4 and 6, and some circles in design 9 have radius 5.



Final digit of student number

Note that all coordinates should be multiples of 10 except in designs 2 and 6 where some are multiples of 5; the circles in design 6 have radii that are multiples of 5. Patch design 9 is made up of 20 straight lines (there are no curved lines).



Important

This is an individual coursework, and so the work you submit for assessment must be your own. Submission of work that is not 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.