School of Computing

Module Code	M30299
Module Title	Programming
Module Coordinator	Dr. Matthew Poole <matthew.poole@port.ac.uk>
Assessment Item number	Item 2
Assessment Title	Python Coursework
Date Issued	2021-12-06

Schedule and Deliverables

Deliverable	Value	Format	Deadline / Date	Late/ECF deadline
Program	100%	A single Python file (with .py suffix)	2022-01-14 23:00 GMT	2022-01-28 23:00
Demo		5-10 minute individual coursework demo. All marks for the program will be awarded in the demo.	In your practical of the week beginning 2022-01-17	If you submit late, then you must arrange and have completed your demo by 2022-02-11.

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 *Code of Student Behaviour* 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 xia.han@port.ac.uk or your lecturers.
- If you are concerned about your mental well-being, please contact our Well-being service.

M30299 – Programming

moodle.port.ac.uk

Python Coursework: A Patchwork Sampler

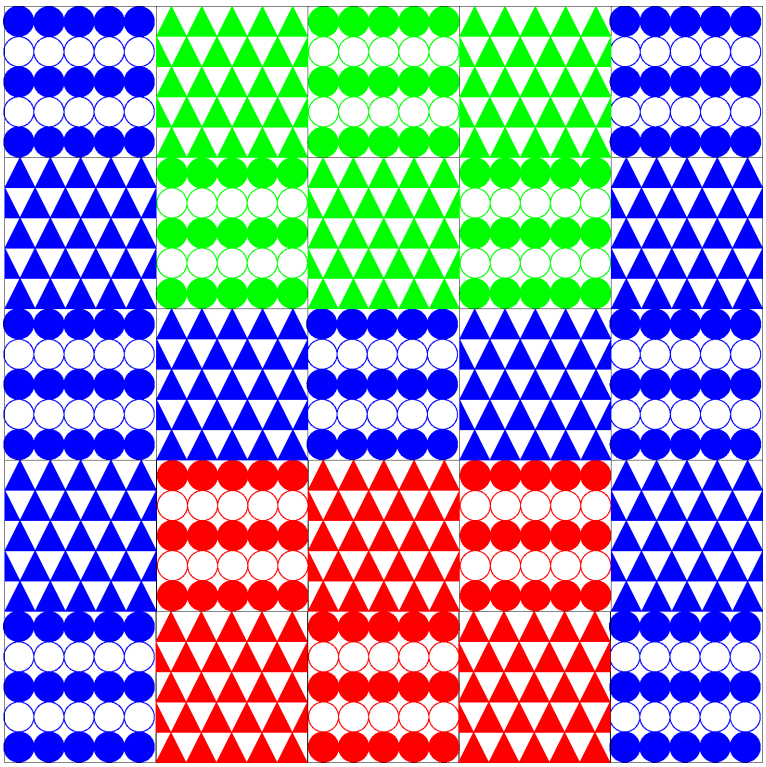
Introduction

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 (it is 100% of assessment item 2 of the module.)

You need to submit your program via the module's Moodle site by the deadline of ***11.00pm, Friday 14th January 2022***, and are required to ***demonstrate*** your submitted program in your 2-hour Programming class timetabled during the week beginning ***17th January 2022***. 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 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 (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**627**, 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 627) 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, 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 even though we haven't covered them in the module.

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, 2001234.py. Click on the link labelled Python Assignment in the Assessment tab and upload your program. If you miss the deadline, you will need to upload it using the Late Submissions link and your mark will be capped according to University regulations.

Demonstration

You need to demonstrate your program to a member of staff in your Programming session timetabled during the week beginning 17th 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 Friday 11th February 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*** (8 marks for each patch design, 10 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 14); (ii) its readability (see lecture 05); 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. 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 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, Xia Han or Eleni Noussi 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 P6, as well as the first few exercises on worksheet P8, before attempting the coursework.

Support

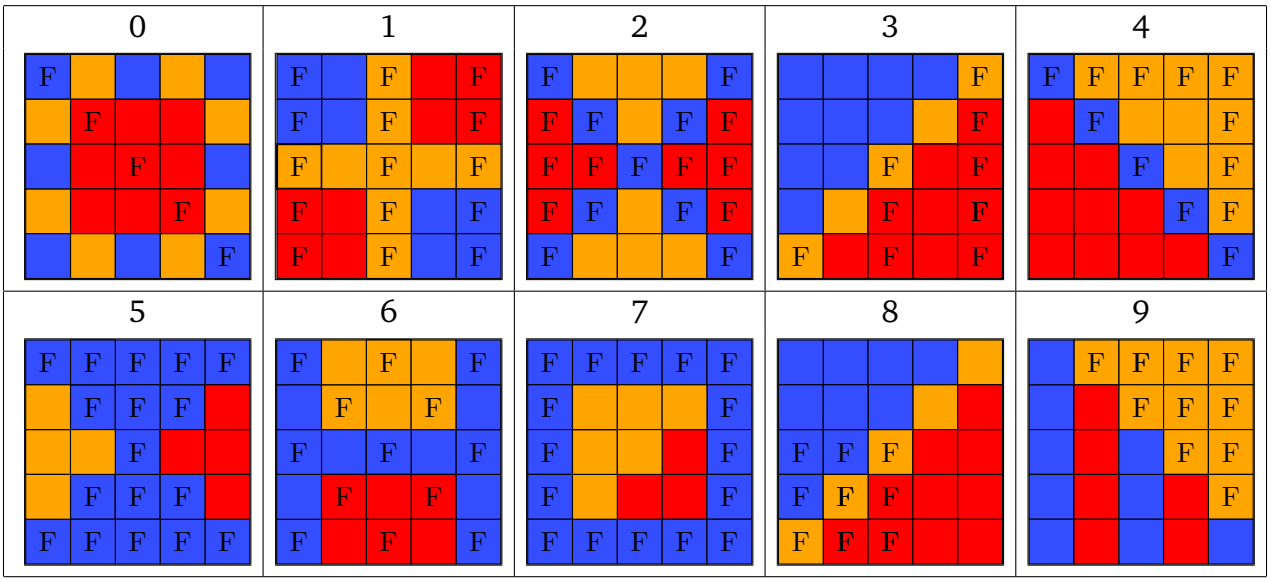
Queries should be addressed to me by email (`Matthew.Poole@port.ac.uk`), although please note that I will be unavailable between 21st December and 3rd January. Any points that come to light that may be of general interest will be communicated via the Python Coursework Frequently Asked Questions link in the General section on Moodle, so please check there before asking a question. The last few practicals before the deadline can be used for help on the coursework. The Academic Tutors, Xia Han and Eleni Noussi, are also able to give advice on the coursework.

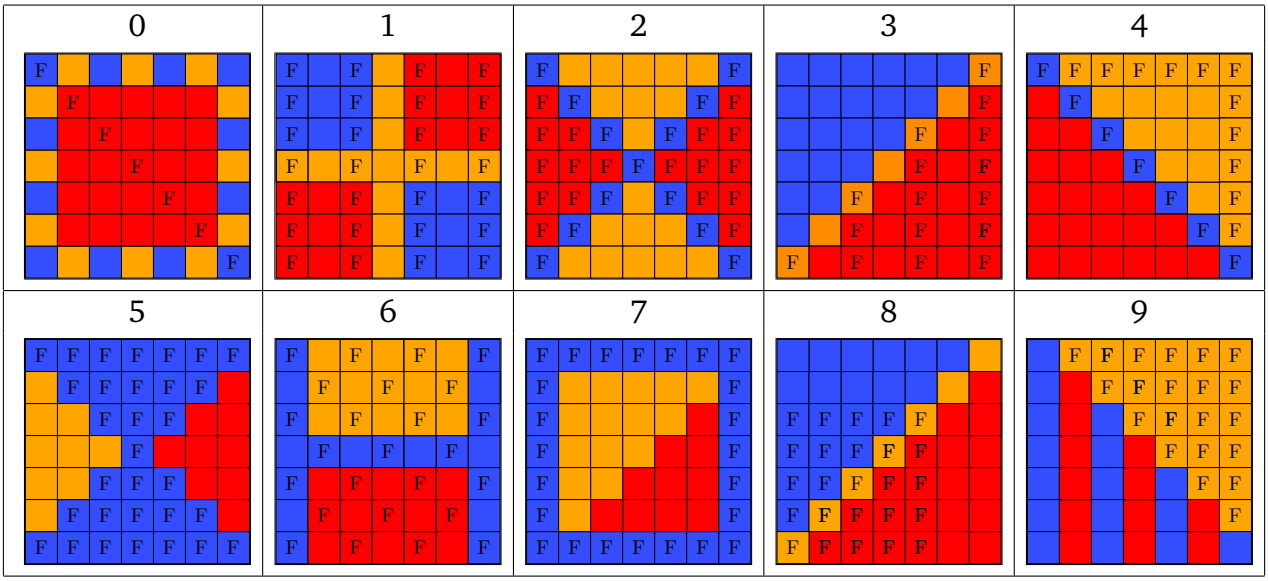
Patchwork layout and patch designs

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 digit of student number

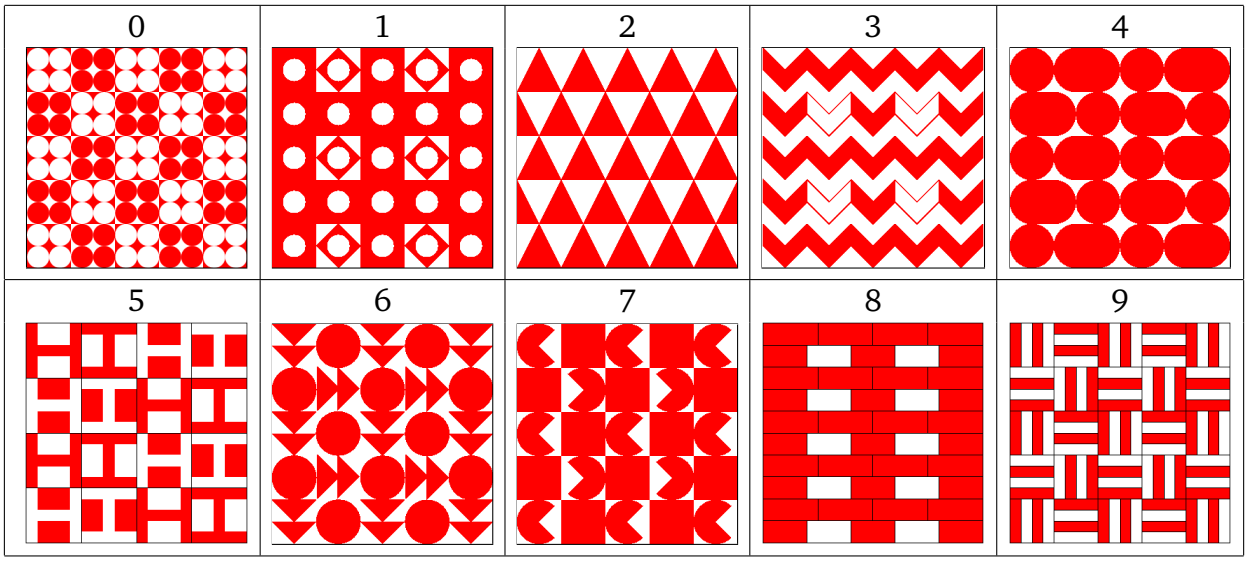
The tables below describe how patches should be arranged and coloured as determined by the antepenultimate digit of your student number in the case of 5×5 and 7×7 patchworks (from these you should be able to determine the layout for a 9×9 patchwork). The tables assume that the first chosen colour is 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.





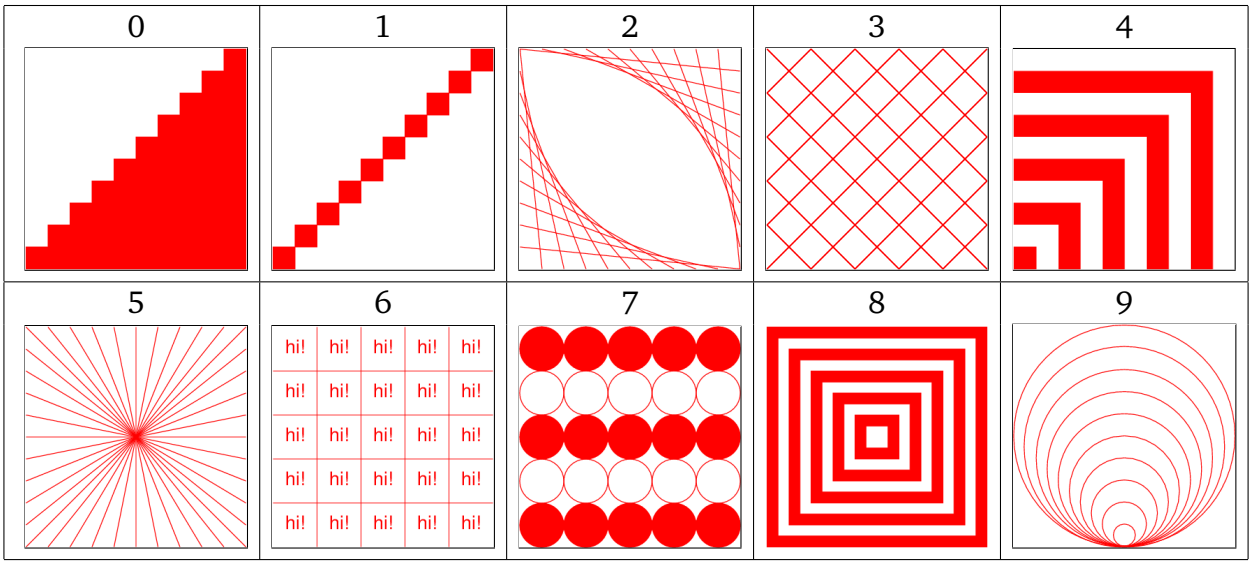
Penultimate digit of student number

Note that all coordinates should be multiples of 10 except in designs 0, 5, 8 and 9 where some are multiples of 5; the circles in designs 0 and 1 have radius 5.



Final digit of student number

Note that all coordinates should be multiples of 10 except in designs 8 and 9 where some are multiples of 5; the circles in design 9 have radii that are multiples of 5. Patch design 2 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
December 2021