

Assignment 1, Part A: Patience

(20%, due 11:59pm Sunday, April 14th, end of Week 7)

Overview

This is the first part of a two-part assignment. This part is worth 20% of your final grade for IFB104. Part B will be worth a further 5%. Part B is intended as a last-minute extension to the assignment, thereby testing the maintainability of your code, and the instructions for completing it will not be released until Week 7. Whether or not you complete Part B you will submit only one file, and receive only one assessment, for the whole 25% assignment.

Motivation

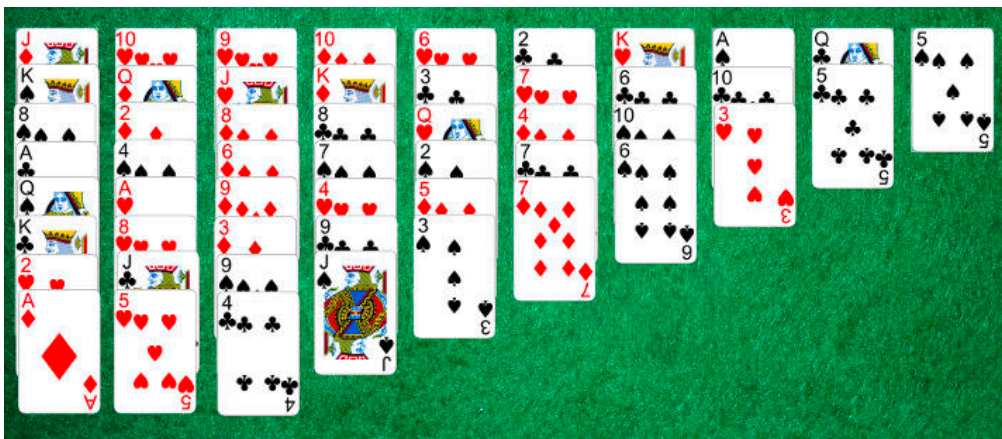
One of the most basic functions of any IT system is to process a given data set to produce some form of human-readable output. This assignment requires you to produce a visual image by following instructions stored in a list. It tests your abilities to:

- Process lists of data values;
- Produce maintainable, reusable code;
- Design a solution to a non-trivial computational problem; and
- Display information in a visual form.

In particular, you will need to think carefully about how to create reusable code segments, via well-planned function definitions and the use of repetition, to make the resulting program concise and easy to understand and maintain.

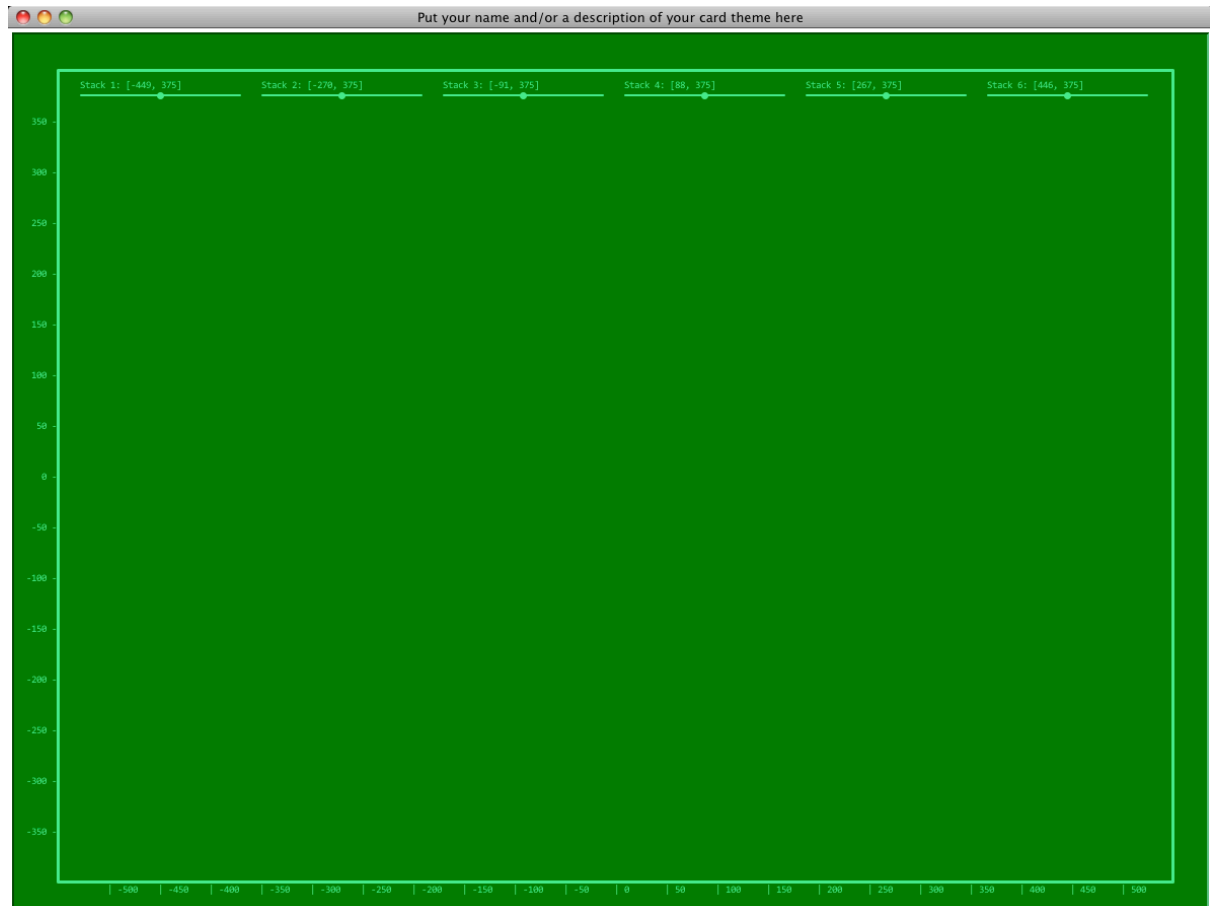
Goal

Even in the 21st Century, traditional card games continue to be popular, although they are now more likely to be played on a smartphone rather than using a cardboard deck. In this assignment you are required to use Python's Turtle graphics module to draw a game of Patience (also known as Solitaire in the United States). To do so you must follow a given list of instructions to draw a specified number of cards in each of several stacks, similar to the game shown below with ten stacks. Most importantly, the game to be drawn will be generated *randomly* each time your program runs, so your solution must be sufficiently general that it can work correctly for *any possible game* that can be described.



Resources provided

A template Python 3 program, `patience.py`, is provided with these instructions. When run it creates a drawing canvas and displays a simple background image representing the card table and the locations for each of the stacks of cards, as shown below.



For convenience the x and y coordinate axes are marked along the edge of the “table”. The locations at which to start drawing each of the six card stacks are indicated up the top with each stack’s centre point marked via a dot at the given coordinates.

In addition to drawing the card table, whenever the template is run it prints a list of instructions in the interpreter’s shell window, e.g.,

Cards to draw (stack, suit, no. cards, option):

```
[['Stack 5', 'Suit B', 5, 2],
 ['Stack 2', 'Suit A', 1, 0],
 ['Stack 6', 'Suit D', 7, 0],
 ['Stack 3', 'Suit C', 6, 0],
 ['Stack 1', 'Suit A', 4, 0]]
```

Your program is required to draw cards in four distinct suits, on separate stacks, as specified by such lists of instructions. However, you have a free choice in the design of the individual cards.

Your code will consist of a function called `deal_cards`, and any auxiliary functions you define to support it. This function takes a single argument, which is a list of instructions describing the game to be drawn. This list of instructions is created by a provided function called `random_game` which *randomly* generates the list of instructions, so your code must work correctly for *any possible* game that can be produced!

Data format

The `random_game` function used to assess your solution returns a list of instructions defining how to draw a game. Each of the instructions is expressed as four values:

1. The name of the stack on which to draw cards.
2. The suit of cards to be drawn on that stack.
3. The number of cards to draw on the stack.
4. A mystery “extra” value, whose purpose will be revealed only in Part B of the assignment. You should ignore it while completing Part A.

As configured in the provided `patience.py` template, up to six stacks can be populated, with up to ten cards per stack. The suits of cards are simply called suit A to D in the instructions. It is your choice to decide what each suit represents and the overall theme for the card deck.

In addition to the `random_game` function, the template file also contains a number of “fixed” data sets. These are provided to help you develop your code, so that you can work with a known game layout, rather than a random one, while debugging your code. However, the “fixed” games will not be used for assessing your solution. Your `deal_cards` function must work correctly for *any* layout randomly generated by function `random_game`.

Designing the cards

To complete this assignment you must design four *entirely distinct* cards representing the four suits in your card deck. The cards must be drawn using Turtle graphics primitives only, must be of a reasonable degree of complexity, and must all be part of some common theme. You have a free choice of theme and are strongly encouraged to be imaginative!

Some possible themes you may consider are:

- Cartoon or comic characters
- TV or movie characters
- Sporting teams
- Business logos (banks, restaurants, IT companies, etc)
- Computer game characters
- Geographical sites (cities, countries or tourist attractions)
- Vehicles (cars, boats, planes, etc)
- Household objects
- Or anything else suitable for creating four distinct and easily-identifiable suits

You also need to choose an appropriate size for your cards, so that they are clearly visible but will not be drawn beyond the boundaries of the card table. Most importantly, the design on the cards must cover most of its surface area so that it is still possible to tell which suit each card has even if it is overlapped by another card.

Illustrative example

To illustrate the requirements we developed a solution which uses a theme inspired by the “My Family” stickers that people put on their car windscreens. (Don’t copy our example! Develop your own idea!) We wrote Turtle graphics code that could draw the following four cards, each representing a family member, and used these images as our four suits.

Suit A (Girl):



Suit B (Dad):



Suit C (Mum):



Suit D (Boy):



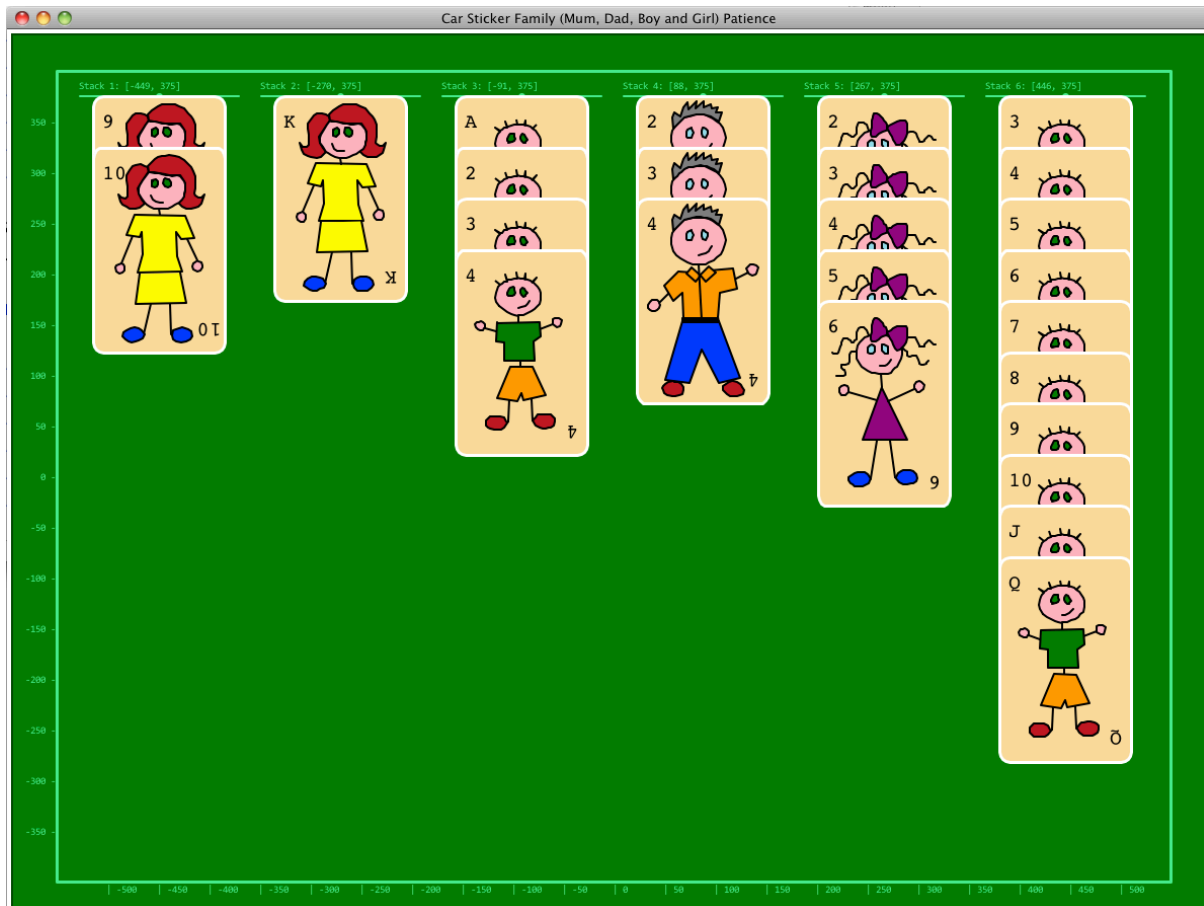
From this basis, our implementation of the `deal_cards` function can follow any layout generated by function `random_game`, drawing the corresponding layout of cards. For instance, consider the following randomly-generated game:

```
[['Stack 1', 'Suit C', 2, 0],  
 ['Stack 6', 'Suit D', 10, 0],  
 ['Stack 2', 'Suit C', 1, 1],  
 ['Stack 4', 'Suit B', 3, 0],  
 ['Stack 3', 'Suit D', 4, 0],  
 ['Stack 5', 'Suit A', 5, 2]]
```

This requires us to draw two cards from suit C on stack 1, ten cards from suit D on stack 6, one card from suit C on stack 2, and so on. The resulting card layout is shown overleaf. Notice that we have set the window title to describe our theme.

We have chosen a width and height for the cards so that they fit comfortably on the card table, even if the maximum number of cards in a stack must be drawn. Importantly, we have chosen the sizes of the images on the cards, and the amount of overlap, so that it is always possible to tell which suit a card belongs to, even if it is partially obscured by an overlapping card.

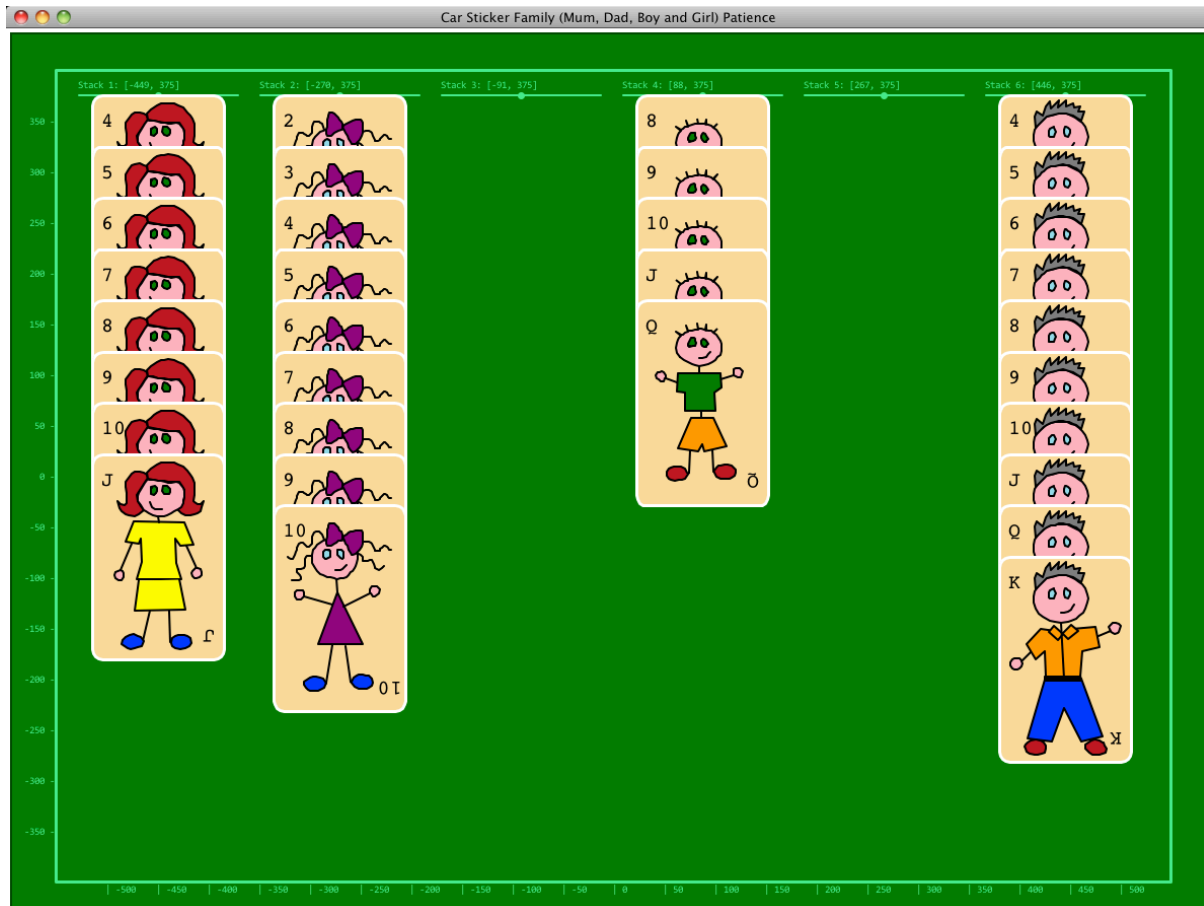
Also, a further requirement is to make each card in each stack distinct. Here we have done this by writing a different face value on each card, although any visual modification that distinguishes each card in the same stack is acceptable. (Of course, whatever feature you add to the cards to distinguish them must be visible when they are overlapped by another card.) It is only necessary to make the cards within a stack distinct, not necessarily between stacks, because function `random_game` may generate more than the standard 13 cards per suit.



In some cases, not all stacks will be populated. A stack should be left empty if we are told to put zero cards on the stack or if there is no instruction for that stack at all. Consider the following game layout:

```
[['Stack 2', 'Suit A', 9, 9],
 ['Stack 4', 'Suit D', 5, 0],
 ['Stack 1', 'Suit C', 8, 3],
 ['Stack 6', 'Suit B', 10, 0],
 ['Stack 3', 'Suit C', 0, 0]]
```

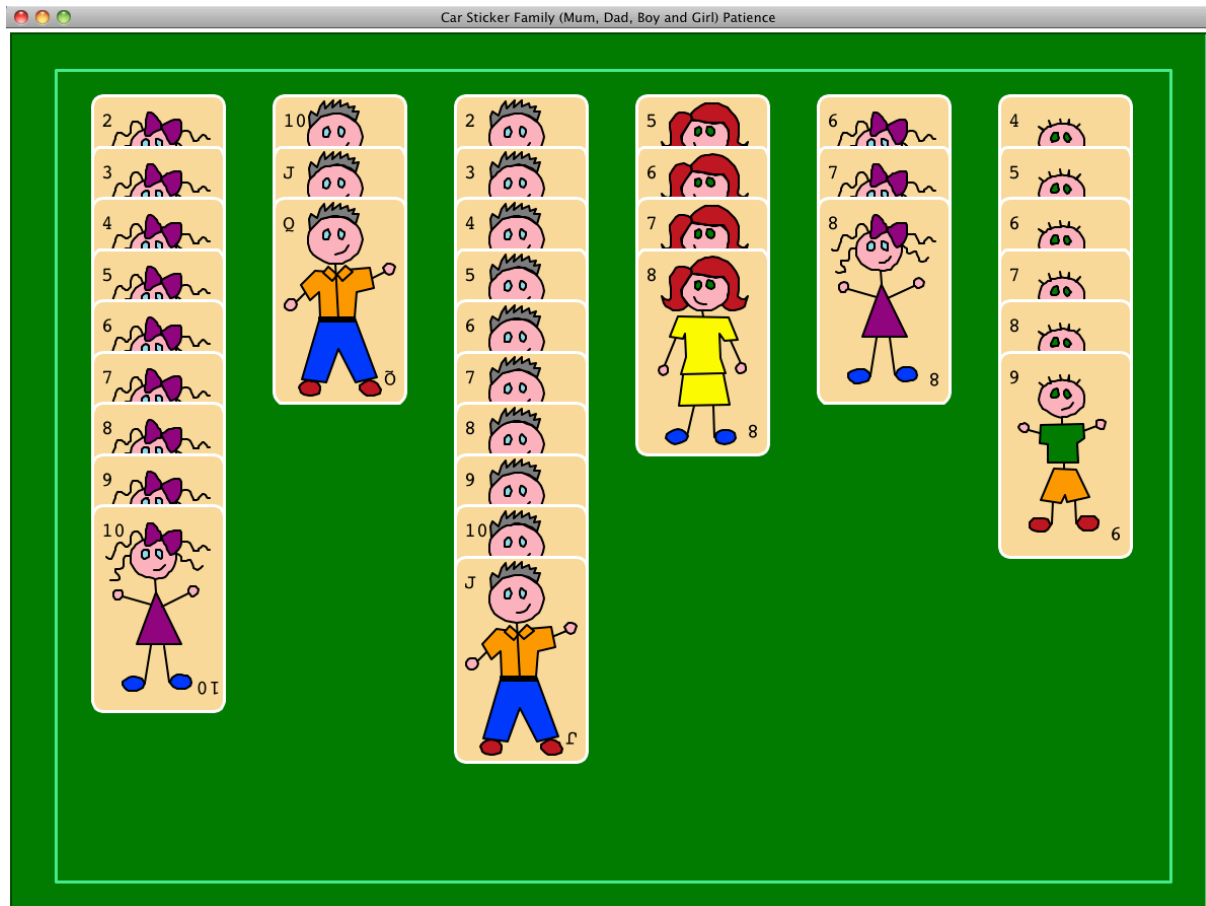
In this case, as shown overleaf, stack 3 is empty because the instruction says to put zero cards on it, and stack 5 is empty because no instruction has been generated for it at all. The resulting image appears below.



Finally, once you're happy with your solution, you can turn off the distracting coordinates along the axes and lines marking the start of the stacks by changing the argument to function `create_drawing_canvas` in the template file's main program. For instance, consider the following data set.

```
[['Stack 6', 'Suit D', 6, 0],
 ['Stack 1', 'Suit A', 9, 0],
 ['Stack 4', 'Suit C', 4, 3],
 ['Stack 3', 'Suit B', 10, 0],
 ['Stack 5', 'Suit A', 3, 2],
 ['Stack 2', 'Suit B', 3, 0]]
```

Running our solution on this data set with the argument to `create_drawing_canvas` set to `False` produced the “clean” image shown overleaf.



Requirements and marking guide

To complete this part of the assignment you are required to extend the provided `patience.py` Python file by completing function `deal_cards` so that it can draw cards as specified by the data sets generated by the `random_game` function. Your `deal_cards` function must work correctly for any values that can be returned by the `random_game` function.

Your submitted solution will consist of a single Python 3 file, and must satisfy the following criteria. Percentage marks available are as shown.

1. **Drawing four entirely distinct cards within a common theme (6%).** Your program must be able to draw four clearly distinct cards, each representing a different suit. The suits must belong to a common theme which is identified by a title on the drawing canvas. Each card must be large enough to be clearly visible but not so big that they won't fit within the space available for the stacks. The images on each card must be drawn using basic Turtle graphics moves and must be of a reasonable degree of complexity, involving multiple Turtle shapes and colours. The images must cover the majority of the card's area so that it is still possible to identify each card even when it is partially overlapped by another.
2. **Placing cards in the stacks as per the instructions (7%).** Your code must be capable of drawing cards in stacks, exactly as dictated by any valid data set provided to function `deal_cards`. The correct suit must be drawn in each stack, as per the in-

structions provided and the correct number of cards must be drawn in each stack. The cards must be centred precisely within each stack and must overlap as if they have been dealt from top to bottom. The cards must not be drawn outside the area marked as the “card table”. The drawings of the cards must preserve their integrity no matter where they are drawn, with no spurious additional or missing lines. Your solution for drawing the cards must work correctly for *any* values that can be returned by the `random_game` function.

3. **Distinguishing each card’s face value (3%).** Within each stack of cards belonging to the same suit, each card must be made distinct from the others. This can be done either using Turtle drawing or by writing text on the card. It must be possible to see the difference between all cards in the same stack, even when they are partially obscured by other cards.
4. **Code quality and presentation (4%).** Your program code, for both Parts A and B of the assignment, must be presented in a professional manner. See the coding guidelines in the *IFB104 Code Presentation Guide* (on Blackboard under *Assessment*) for suggestions on how to achieve this. In particular, given the obscure and repetitive nature of the code needed to draw complex images using Turtle graphics, each significant code segment must be clearly commented to describe its purpose. Similarly, the names of your functions and variables should be indicative of their role, not just “i”, “j”, “k”, etc. Also, you must use function definitions and loops to avoid unnecessary duplication of similar or identical code segments. To get full marks for this criterion you must provide a significant amount of code to assess.
5. **Extra feature (5%).** *Part B of this assignment will require you to make a last-minute extension to your solution. The instructions for Part B will not be released until shortly before the final deadline for Assignment 1.*

Finally, you are *not* required to copy the example shown in this document. Instead you are strongly encouraged to be creative in the design of your solution. Surprise us!

Portability

An important aspect of software development is to ensure that your solution will work correctly on all computing platforms (or at least as many as possible). For this reason you must **complete the assignment using standard Turtle graphics, random number and maths functions only**. You may not import any additional modules or files into your program other than those already imported by the given template file. In particular, you may not import any image files to help create your drawings or use non-standard image processing modules such as Pillow.

Security warning and plagiarism notice

This is an individual assessment item. All files submitted will be subjected to software plagiarism analysis using the MoSS system (<http://theory.stanford.edu/~aiken/moss/>). Serious violations of the university’s policies regarding plagiarism will be forwarded to the Science and Engineering Faculty’s Academic Misconduct Committee for formal prosecution.

As per QUT rules, you are not permitted to copy *or share* solutions to individual assessment items. In serious plagiarism cases SEF’s Academic Misconduct Committee prosecutes both the copier and the original author equally. It is your responsibility to keep your solution se-

cure. In particular, **you must not make your solution visible online via cloud-based code development platforms such as GitHub**. Note that free accounts for such platforms are usually public. If you wish to use such a resource, do so only if you are certain you have a *private* repository that cannot be seen by anyone else. For instance, university students can apply for a *free* private repository in GitHub to keep their assignments secure (<https://education.github.com/pack>). However, we recommend that the best way to avoid being prosecuted for plagiarism is to keep your work well away from the Internet!

Artistic merit – The Hall of Fame!

You will not be assessed on the artistic merit of your solution, however, a “Hall of Fame” containing the solutions considered the most artistic or ambitious by the assignment markers will be created on Blackboard. (Sadly, additional marks will not be awarded to the winners, only kudos.)

Deliverable

You must develop your solution by completing and submitting the provided Python 3 file `patience.py` as follows.

1. Complete the “statement” at the beginning of the Python file to confirm that this is your own individual work by inserting your name and student number in the places indicated. *We will assume that submissions without a completed statement are not your own work!*
2. Complete your solution by developing Python code to replace the dummy `deal_cards` function. You should complete your solution using only the standard Python 3 modules already imported by the provided template. In particular, you must *not* use any Python modules that must be downloaded and installed separately because the markers will not have access to these modules. Furthermore, you may *not* import any image files into your solution; the entire image must be drawn using Turtle graphics drawing primitives.
3. Submit *a single Python file* containing your solution for marking. Do *not* submit multiple files. Only a single file will be accepted, so you cannot accompany your solution with other files or pre-defined images. **Do not submit any other files! Submit only a single Python 3 file!**

Apart from working correctly your program code must be well-presented and easy to understand, thanks to (sparse) commenting that explains the *purpose* of significant code segments and *helpful* choices of variable and function names. *Professional presentation* of your code will be taken into account when marking this assignment.

If you are unable to solve the whole problem, submit whatever parts you can get working. You will receive *partial marks for incomplete solutions*.

How to submit your solution

A link is available on the IFB104 Blackboard site under *Assessment* for uploading your solution file before the deadline (11:59pm Sunday, April 14th, end of Week 7). You can *submit as many drafts of your solution as you like*. You are strongly encouraged to *submit draft solutions* before the deadline as insurance against computer or network problems near the dead-



line. If you are unsure whether or not you have successfully uploaded your file, upload it again!

Students who encounter problems uploading their Python files to Blackboard should contact the *IT Helpdesk* (ithelpdesk@qut.edu.au; 3138 4000) for assistance and advice. Teaching staff will *not* be available to answer email queries on the weekend the assignment is due, so ensure that you have successfully uploaded at least one solution by close-of-business on Friday, April 12th.

Appendix: Some standard Turtle graphics colours you can use

Source: www.discoveryplayground.com/computer-programming-for-kids/rgb-colors/

Whites/Pastels

Color Name	RGB CODE	HEX #	Sample
Snow	255-250-250	ffaafa	
Snow 2	238-233-233	eee9e9	
Snow 3	205-201-201	cdc9c9	
Snow 4	139-137-137	8b8989	
Ghost White	248-248-255	f8f8ff	
White Smoke	245-245-245	f5f5f5	
Gainsboro	220-220-220	dccdc	
Floral White	255-250-240	ffaaf0	
Old Lace	253-245-230	fdf5e6	
Linen	240-240-230	faf0e6	
Antique White	250-235-215	faebd7	
Antique White 2	238-223-204	eedfcc	
Antique White 3	205-192-176	cdc0b0	
Antique White 4	139-131-120	8b8378	
Papaya Whip	255-239-213	ffefd5	
Blanched Almond	255-235-205	ffeacd	
Bisque	255-228-196	ffe4c4	
Bisque 2	238-213-183	eed5b7	
Bisque 3	205-183-158	cdb79e	
Bisque 4	139-125-107	8b7d6b	
Peach Puff	255-218-185	ffdab9	
Peach Puff 2	238-203-173	eecbad	
Peach Puff 3	205-175-149	cdaf95	
Peach Puff 4	139-119-101	8b7765	
Navajo White	255-222-173	ffdead	
Moccasin	255-228-181	ffe4b5	
Cornsilk	255-248-220	fff8dc	
Cornsilk 2	238-232-205	eee8dc	
Cornsilk 3	205-200-177	cdc8b1	
Cornsilk 4	139-136-120	8b8878	
Ivory	255-255-240	fffff0	

Ivory 2	238-238-224	eeeeee0	
Ivory 3	205-205-193	cdcdc1	
Ivory 4	139-139-131	8b8b83	
Lemon Chiffon	255-250-205	fffacd	
Seashell	255-245-238	fff5ee	
Seashell 2	238-229-222	eee5de	
Seashell 3	205-197-191	cdc5bf	
Seashell 4	139-134-130	8b8682	
Honeydew	240-255-240	f0ff0	
Honeydew 2	244-238-224	e0eee0	
Honeydew 3	193-205-193	c1cdc1	
Honeydew 4	131-139-131	838b83	
Mint Cream	245-255-250	f5fffa	
Azure	240-255-255	f0ffff	
Alice Blue	240-248-255	f0f8ff	
Lavender	230-230-250	e6e6fa	
Lavender Blush	255-240-245	fff0f5	
Misty Rose	255-228-225	ffe4e1	
White	255-255-255	ffffff	





















Blues

Color Name	RGB CODE	HEX #	Sample
Midnight Blue	25-25-112	191970	
Navy	0-0-128	000080	
Cornflower Blue	100-149-237	6495ed	
Dark Slate Blue	72-61-139	483d8b	
Slate Blue	106-90-205	6a5acd	
Medium Slate Blue	123-104-238	7b68ee	
Light Slate Blue	132-112-255	8470ff	
Medium Blue	0-0-205	0000cd	
Royal Blue	65-105-225	4169e1	
Blue	0-0-255	0000ff	
Dodger Blue	30-144-255	1e90ff	
Deep Sky Blue	0-191-255	00bfff	
Sky Blue	135-206-250	87ceeb	
Light Sky Blue	135-206-250	87cefa	
Steel Blue	70-130-180	4682b4	
Light Steel Blue	176-196-222	b0c4de	
Light Blue	173-216-230	add8e6	
Powder Blue	176-224-230	b0e0e6	
Pale Turquoise	175-238-238	afeeee	
Dark Turquoise	0-206-209	00ced1	
Medium Turquoise	72-209-204	48d1cc	
Turquoise	64-224-208	40e0d0	
Cyan	0-255-255	00ffff	
Light Cyan	224-255-255	e0ffff	
Cadet Blue	95-158-160	5f9ea0	







Grays

Color Name	RGB CODE	HEX #	Sample
Black	0-0-0	000000	
Dark Slate Gray	49-79-79	2f4f4f	
Dim Gray	105-105-105	696969	
Slate Gray	112-138-144	708090	
Light Slate Gray	119-136-153	778899	
Gray	190-190-190	bebebe	
Light Gray	211-211-211	d3d3d3	














Greens

Color Name	RGB CODE	HEX #	Sample
Medium Aquamarine	102-205-170	66cdad	
Aquamarine	127-255-212	7fffd4	
Dark Green	0-100-0	006400	
Dark Olive Green	85-107-47	556b2f	
Dark Sea Green	143-188-143	8fbc8f	
Sea Green	46-139-87	2e8b57	
Medium Sea Green	60-179-113	3cb371	
Light Sea Green	32-178-170	20b2aa	
Pale Green	152-251-152	98fb98	
Spring Green	0-255-127	00ff7f	
Lawn Green	124-252-0	7fcf00	
Chartreuse	127-255-0	7fff00	
Medium Spring Green	0-250-154	00fa9a	
Green Yellow	173-255-47	adff2f	
Lime Green	50-205-50	32cd32	
Yellow Green	154-205-50	9acd32	
Forest Green	34-139-34	228b22	
Olive Drab	107-142-35	6b8e23	
Dark Khaki	189-183-107	bdb76b	
Khaki	240-230-140	f0e68c	











Yellow

Color Name	RGB CODE	HEX #	Sample
Pale Goldenrod	238-232-170	eee8aa	
Light Goldenrod Yellow	250-250-210	fafad2	
Light Yellow	255-255-224	ffffe0	
Yellow	255-255-0	ffff00	
Gold	255-215-0	ffd700	
Light Goldenrod	238-221-130	eedd82	
Goldenrod	218-165-32	daa520	
Dark Goldenrod	184-134-11	b8860b	


















Browns

Color Name	RGB CODE	HEX #	Sample
Rosy Brown	188-143-143	bc8f8f	
Indian Red	205-92-92	cd5c5c	
Saddle Brown	139-69-19	8b4513	
Sienna	160-82-45	a0522d	
Peru	205-133-63	cd853f	
Burlywood	222-184-135	deb887	
Beige	245-245-220	f5f5dc	
Wheat	245-222-179	f5deb3	
Sandy Brown	244-164-96	f4a460	
Tan	210-180-140	d2b48c	
Chocolate	210-105-30	d2691e	
Firebrick	178-34-34	b22222	
Brown	165-42-42	a52a2a	

Oranges

Color Name	RGB CODE	HEX #	Sample
Dark Salmon	233-150-122	e9967a	
Salmon	250-128-114	fa8072	
Light Salmon	255-160-122	ffa07a	
Orange	255-165-0	ffa500	
Dark Orange	255-140-0	ff8c00	
Coral	255-127-80	ff7f50	
Light Coral	240-128-128	f08080	
Tomato	255-99-71	ff6347	
Orange Red	255-69-0	ff4500	
Red	255-0-0	ff0000	

Pinks/Violets

Color Name	RGB CODE	HEX #	Sample
Hot Pink	255-105-180	ff69b4	
Deep Pink	255-20-147	ff1493	
Pink	255-192-203	ffc0cb	
Light Pink	255-182-193	ffb6c1	
Pale Violet Red	219-112-147	db7093	
Maroon	176-48-96	b03060	
Medium Violet Red	199-21-133	c71585	
Violet Red	208-32-144	d02090	
Violet	238-130-238	ee82ee	
Plum	221-160-221	dda0dd	
Orchid	218-112-214	da70d6	
Medium Orchid	186-85-211	ba55d3	
Dark Orchid	153-50-204	9932cc	
Dark Violet	148-0-211	9400d3	
Blue Violet	138-43-226	8a2be2	
Purple	160-32-240	a020f0	
Medium Purple	147-112-219	9370db	
Thistle	216-191-216	d8bfd8	