Assignment 1, Part A: Jigsaw Puzzle

(17%, due 11:59pm Friday, September 9th)

Overview

This is the first part of a two-part assignment. This part is worth 17% of your final grade for IFB104. Part B will be worth a further 8%. 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 mark, 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 task requires you to produce a visual image based on data stored in a list. It tests your abilities to:

- Process sequences of data values;
- Perform arithmetic operations;
- Display information in a visual form; and
- Produce reusable code.

Goal

Jigsaw puzzles are a familiar, traditional pastime. In this assignment you are required to develop a Python program which processes data stored in a list to display an attempt to solve a jigsaw puzzle consisting of four distinct pieces. To do so you will need to use basic Python features and the Turtle graphics module.

You must design four interlocking puzzle pieces which, when assembled in the correct order, produce a single picture. The picture must be non-trivial, and must span all four pieces, but otherwise you have a free choice of what to draw, e.g., cartoon, game or science fiction characters, household objects, corporate or sporting logos, buildings or vehicles, animals or pets, landscapes, etc.

To position the pieces you must develop your code so that they can be drawn in different locations on the screen. A skeletal Python program, `jigsaw_puzzle.py`, is provided with these instructions which draws:

- A four-place template for putting pieces whose position has been chosen; and
- A box to contain unused pieces.

It also contains several data sets, in the form of lists, to guide your drawing of the attempted solution to the puzzle. The lists contain instructions in two or three parts:

- The identity of the jigsaw puzzle piece to draw, from '`Piece A`' to '`Piece D`'.

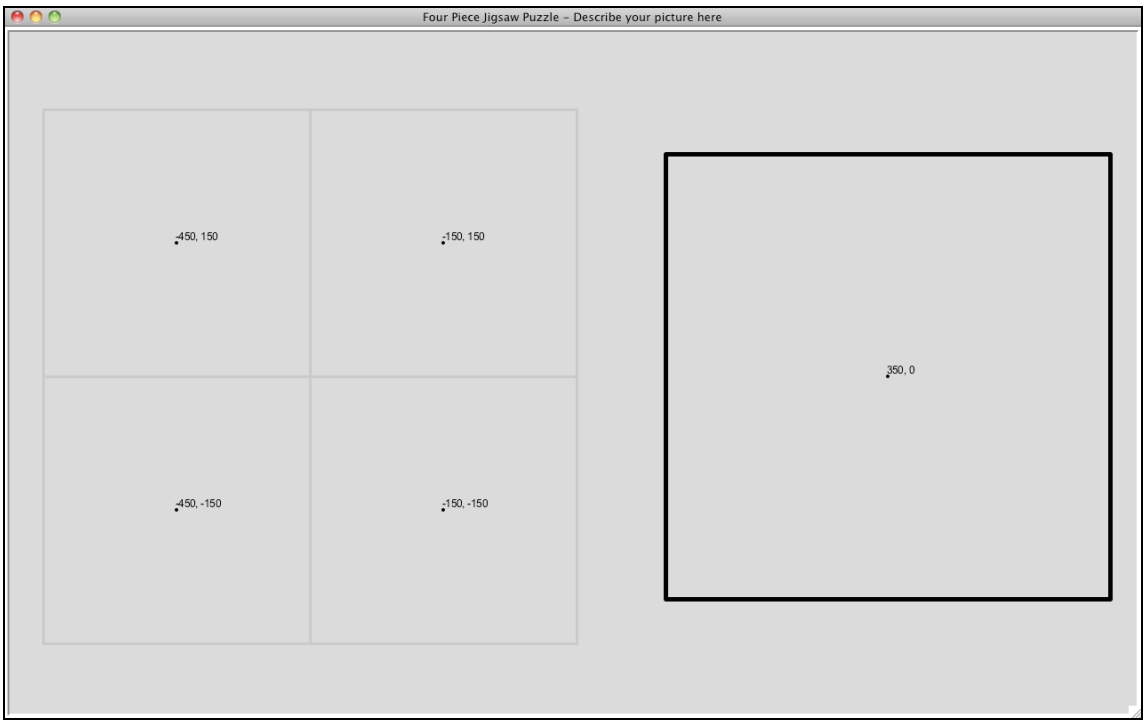
- The place where the piece must be drawn, either in one of the four template locations, '`Top left`', '`Top right`', '`Bottom left`' or '`Bottom right`', or in the box of unused pieces, as indicated by '`In box`'.
- An optional mystery value, '`X`', whose purpose will be revealed only in the second part of the assigment.

All four jigsaw pieces must have a different shape, with protruding "tabs" that interlock into corresponding "blanks" in other pieces, just like a physical jigsaw puzzle. When assembled correctly the four pieces must form a perfect square. The picture produced by assembling the puzzle correctly must be non-trivial, and must span all four pieces. For instance, four separate images, one per piece, would be unacceptable. Although it's difficult to generalise the artistic requirements for this assignment, given the wide range of pictures that could be chosen, it's expected the assembled image would involve several different shapes of several different colours and the resulting picture must be immediately recognisable.

You are required to use Turtle graphics to draw the pieces in the places specified by any of the given data sets, and your code should work for any other similar data sets in the same format. Furthermore, you must provide your own data set for the correct solution to your particular puzzle.

Illustrative example

To get you started, you will find a Python file, `jigsaw_puzzle.py`, accompanying these instructions. When you run this program it will produce a drawing canvas as shown below. There is a neutral grey background and five squares. The four squares on the left mark the template where the four-piece jigsaw puzzle will be assembled. The bigger square on the right is the "box" in which unused jigsaw pieces are stored.

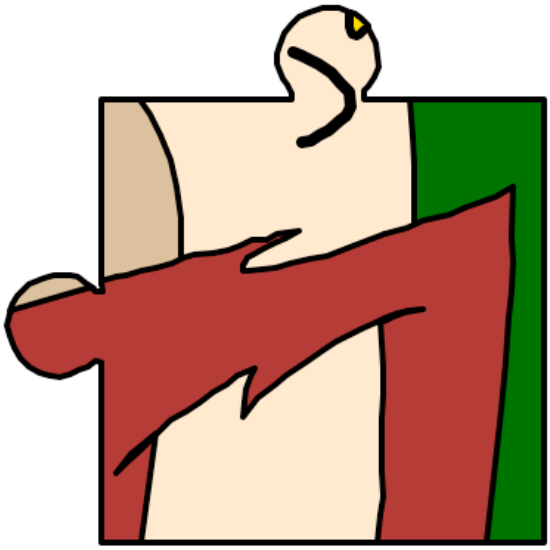


To help you develop your solution, the centre coordinates of each square are also marked. Each of the squares in the template on the left is a 300 pixel square, so the pieces of your jigsaw puzzle must all be of this basic size, excluding tabs. The box allows for tabs to protrude from pieces as far as 100 pixels in any direction, so your pieces should not have tabs larger than this.

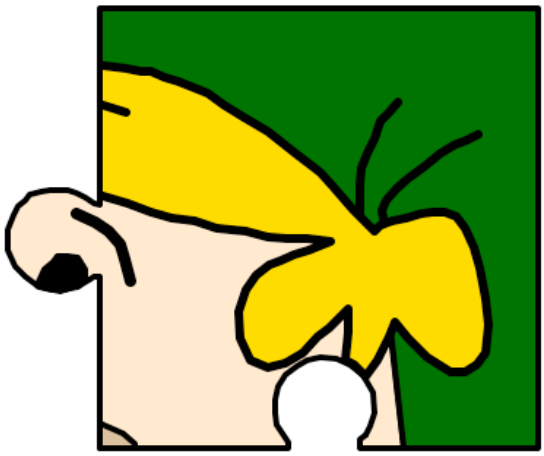
Your first challenge is to develop code that will use Turtle graphics to draw four distinct jigsaw puzzle pieces with these dimensions. As an illustrative example, we have developed code that draws the following four pieces:

Piece A

Piece B



Piece C



Piece D

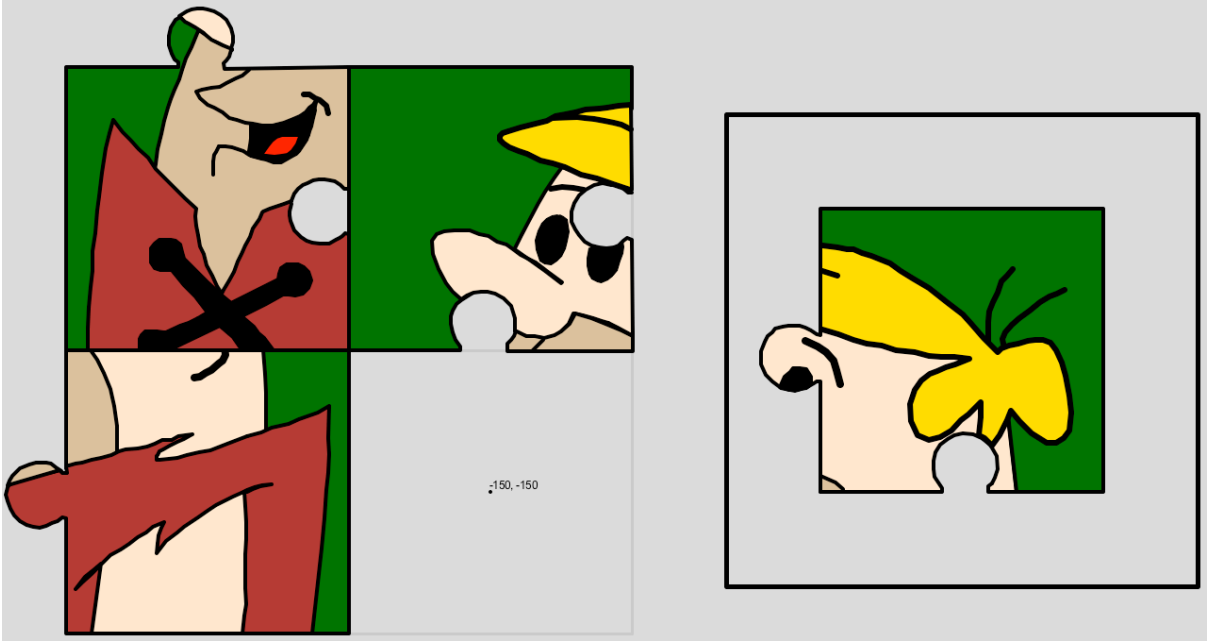
Excluding the protruding tabs and indented blanks, each of these pieces is a 300 × 300 pixel square. When assembled correctly they produce a single image which forms a perfect 600 × 600 pixel square. Notice that each piece has a different shape as well as a different image.

The pieces are "fully interlocking", meaning that each piece shares at least one tab/blank with both of its immediate neighbours.

In the Python template file you will find several data sets in the form of lists, e.g.,

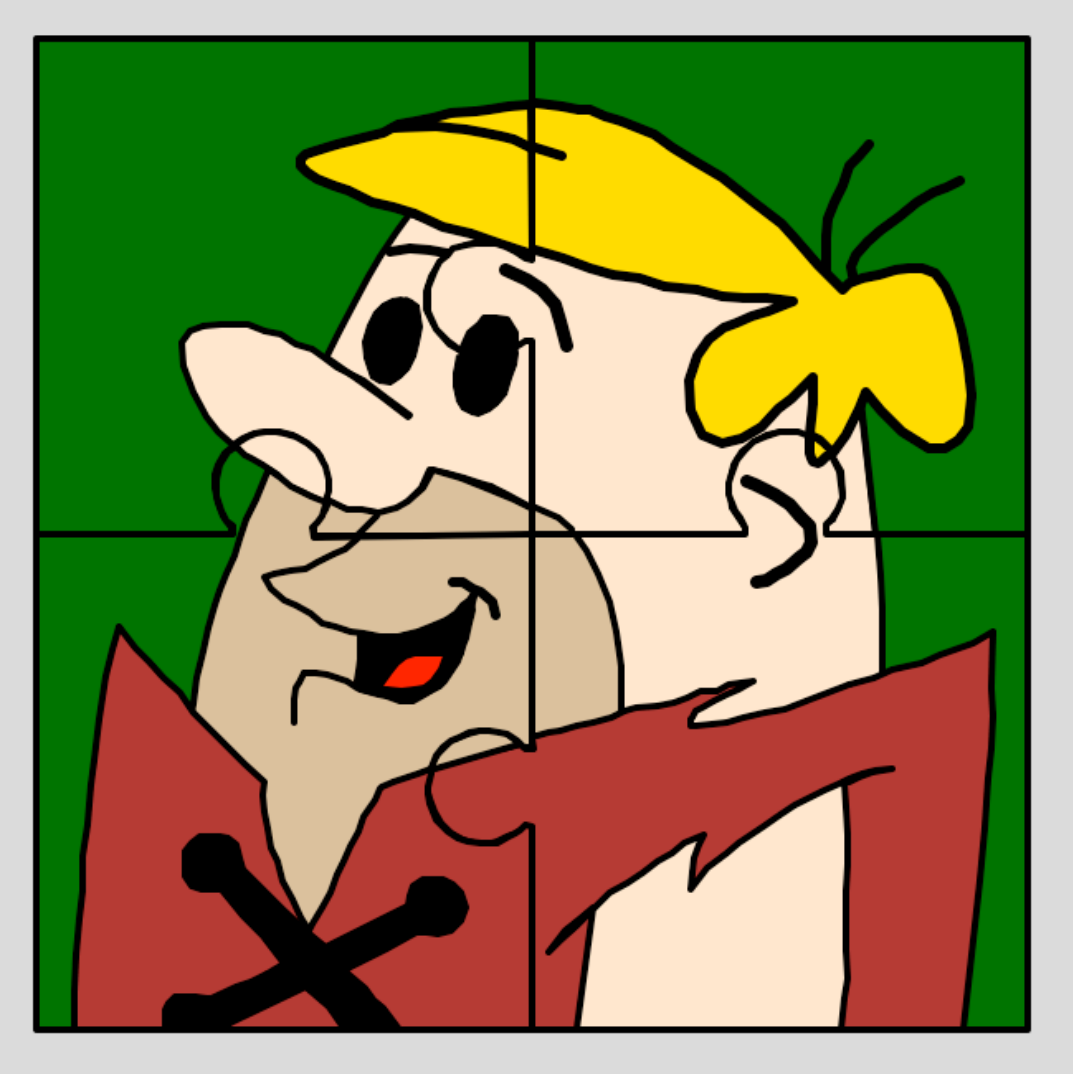
```
attempt_27 = [['Piece C', 'Bottom left'], ['Piece D', 'In box'],
              ['Piece A', 'Top left'], ['Piece B', 'Top right']]
```

These lists represent attempts to solve the puzzle. Each element of the list is itself a list which specifies which piece is to be drawn and where. For instance, `attempt_27` above requires us to draw our 'Piece C' centred in the bottom left, 'Piece D' centred in the box, and so on. Unfortunately this particular attempted solution is unsuccessful for the pieces we have designed and produces the following image.



Not only are the pieces misplaced, but the uppermost tab on Piece C (at the bottom left) has been entirely obscured by Piece A (at the top left). Nonetheless, we are required to draw the pieces wherever instructed, even if their tabs overlap. Similarly, some of the data sets require several pieces to be left in the box, in which case they should simply be drawn on top of each other. Also, some data sets don't mention all of the pieces, in which case only the pieces listed should be drawn. Finally, some lists contain a third value, 'X', associated with certain pieces. For now you can ignore these optional values. Their purpose will be revealed in Part B of this assignment.

Your final task is to provide instructions for correctly assembling your particular collection of jigsaw pieces, using the list "`solution`" in the provided Python file. In our case the correct solution is to put Piece B in the top left, Piece D in the top right, Piece C in the bottom right and Piece A in the bottom left. Applying these instructions finally reveals our overall picture to be a portrait of Fred Flintstone's next door neighbour and best friend, Barney Rubble, as shown overleaf…



Requirements and marking guide

To complete this task you are required to extend the provided `jigsaw_puzzle.py` Python file by completing function `draw_attempt` so that it can draw jigsaw puzzle pieces at the places specified by a data set provided as its single parameter. Your code must work for all the supplied "attempt" data sets and any other data set in the same format. You must also provide your own "solution" data set that completes your puzzle correctly.

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

1. **Drawing four distinct puzzle pieces (4%)**. Your program must be able to draw *four distinct puzzle pieces*, each of a *different shape*. The basic shape of each piece must be a 300 × 300 pixel square, but they can have any number of "tabs" that protrude by up to 100 pixels (and corresponding indented "blanks"). When assembled correctly, as per your provided solution, all of the pieces must *fit together precisely* as a 600 × 600 pixel square in the provided template, with no parts overlapping or sticking out.

Also your puzzle must be *fully interlocking*, meaning that when assembled correctly each piece shares at least one tab/blank with each of its immediate neighbours.

2. **Drawing a picture in four parts (4%)**. Each of your puzzle pieces must contain one piece of a single complete picture. Each piece must contain a *different, non-trivial part of the overall picture*. The whole picture must span all four pieces. When assembled correctly, as per your provided solution, the parts of the complete picture must *align correctly*. The picture should be clearly recognisable and of a reasonable degree of complexity, involving multiple shapes and colours.
3. **Relocating puzzle pieces (5%)**. Your code must be capable of drawing each of the four puzzle pieces *at any of the five marked places*, either in the four-place jigsaw template on the left or in the unused pieces box on the right. The pieces must preserve their appearance no matter where they are drawn and must fit perfectly into the marked places (although "tabs" may stick out, of course). Your solution for relocating the pieces must work for *all of the provided data sets* and *any other data sets* in the same format. (You cannot "hardwire" your code for specific data sets and you may not change the data sets provided.)
4. **Providing a solution to the puzzle (2%)**. You must provide a *"solution" list*, using the same data format as the "attempt" data sets, whose contents tell us how to solve your particular puzzle correctly. When your `draw_attempt` function is called with this list as its argument your program should draw a perfect solution to the puzzle. Your solution list should *not* contain the optional third value, '`X`', in any of its elements. NB: If you do not provide a solution list we will not be able to assess how well your pieces fit together and you cannot receive full marks for Criteria 1 and 2 above.
5. **Code quality and presentation (2%)**. Your program code 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 say what it does, e.g., "Draw Barney's right eye", "Draw Barney's left ear", etc.
6. ***Extra feature (8%)***. *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 just before the final deadline for Assignment 1.*

You must complete the task using *basic Turtle graphics and maths functions only*. You may not import any additional modules or files into your program other than those already included in the given `jigsaw_puzzle.py` file. In particular, you *may not import any image files* for use in creating your puzzle pieces.

Most importantly, you are *not* required to copy the puzzle piece shapes shown in this document. Instead you are strongly encouraged to *be creative* and to choose your own shapes and an overall image that interests you personally.

Development hints

- This is not a difficult task, but due to the need to create puzzle pieces that fit together properly it can be a time-consuming one, so you are strongly encouraged to *design your puzzle pieces carefully* before developing any program code.
- The hardest part of this assignment is the need to allow the pieces to be drawn in different locations. You therefore need to devise a way of drawing each piece so that you can control its position either by (a) making all drawing moves *relative* to the starting position (e.g., using Turtle's `forward`, `left` and `right` commands) or (b) by calculating *absolute* positions for each drawing move (e.g., using Turtle's `goto` command) in terms of a given position.
- If you are unable to complete the whole task, just submit whatever parts you can get working. You will receive *partial marks for incomplete solutions*.
- To help you debug your code we have provided several data sets, numbered 1 to 12, which draw just one piece at a time. You can use these to help *create the code for each puzzle piece separately*.
- Part B of this assignment will require you to add an extra feature to your solution in a short space of time. You are therefore encouraged to keep *code maintainability* in mind while developing your solution to Part A. Make sure your code is neat and well-commented so that you will find it easy to extend when the instructions for Part B are released.

Deliverable

You must develop your solution by completing and submitting the provided Python file `jigsaw_puzzle.py` as follows. **Do not submit any other files! Do not submit a compressed archive ('zip' or 'rar')!**

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 and they will not be marked.*
2. Complete your solution by developing Python code to replace the dummy `draw_attempt` function. You must complete your solution using *only the modules already imported by the provided template*. You may *not* use or import any other modules to complete this program. In particular, you may *not* import any image files into your solution.
3. Submit *a single Python file* containing your solution for marking. Do *not* submit an archive (e.g., in 'zip' or 'rar' formats) containing several files. Only a single file will be accepted, so you cannot accompany your solution with other files or pre-defined images.

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 will be available on Blackboard under *Assessment* for uploading your solution file before the deadline (11:59pm Friday, September 9th). You can submit as many drafts of your solution as you like. You are strongly encouraged to *submit draft solutions* before the deadline.

Special note for Microsoft Windows 10 users

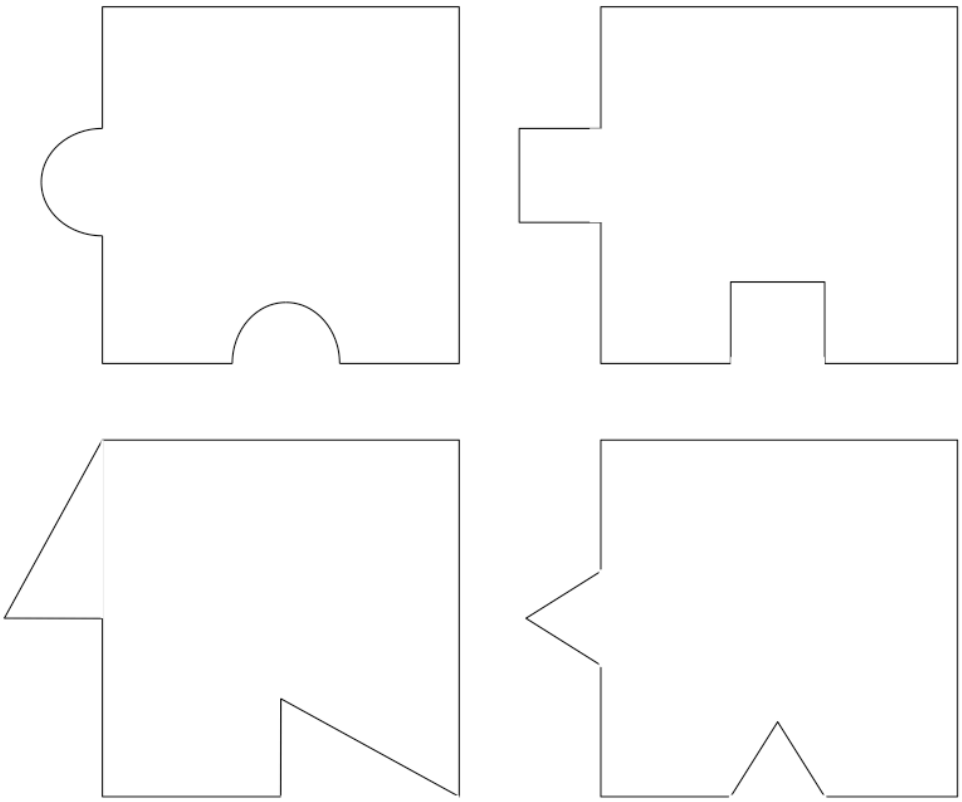
Recently some students have reported difficulties when uploading files to Blackboard from computers running the Microsoft Windows 10 operating system. In these cases Blackboard rejects the upload attempt with a red warning message saying that the file is suspected of being malware. This issue has been investigated by QUT's IT Helpdesk and is believed to be a problem with the Windows 10 operating system, especially when using the Edge browser.

The IT Helpdesk has advised that if this problem is encountered then the file to be uploaded should first be opened using any Windows 10 application, such as Notepad, running on the same machine from which the upload is being attempted, after which the upload should work.

Nonetheless, the precise source of this problem has not been fully identified. Therefore, students using Microsoft Windows 10 who encounter problems uploading their Python files to Blackboard should contact the IT Helpdesk (ithelpdesk@qut.edu.au; 3138 4000) for assistance and advice.

Appendix A: Other types of jigsaw puzzle tabs and blanks

Traditionally jigsaw puzzle pieces interlock via the presence of protruding "tabs" and corresponding indented "blanks". In our illustrative example above we have used circular tabs and blanks similar to those found in most contemporary puzzles. However, you are encouraged to be creative in the design of your pieces and to consider other ways in which the pieces can be made to interlock. For instance, some simpler forms of tabs and blanks that would be acceptable for this assignment are shown below.



However, whichever forms of tabs and blanks you choose, all four pieces of your puzzle must be shaped differently and the four pieces must be "fully interlocking" when the puzzle is assembled correctly.

Appendix B: Some standard Turtle graphics colours

Red colors						Green colors						Brown colors					
IndianRed	CD	5C	5C	205	92	92	GreenYellow	AD	FF	2F	173	255	47	Cornsilk	FF	F8	DC	255	248	220
LightCoral	F0	80	80	240	128	128	Chartreuse	7F	FF	00	127	255	0	BlanchedAlmond	FF	EB	CD	255	235	205
Salmon	FA	80	72	250	128	114	LawnGreen	7C	FC	00	124	252	0	Bisque	FF	E4	C4	255	228	196
DarkSalmon	E9	96	7A	233	150	122	Lime	00	FF	00	0	255	0	NavajoWhite	FF	DE	AD	255	222	173
LightSalmon	FF	A0	7A	255	160	122	LimeGreen	32	CD	32	50	205	50	Wheat	F5	DE	B3	245	222	179
Crimson	DC	14	3C	220	20	60	PaleGreen	98	FB	98	152	251	152	BurlyWood	DE	B8	87	222	184	135
Red	FF	00	00	255	0	0	LightGreen	90	EE	90	144	238	144	Tan	D2	B4	8C	210	180	140
FireBrick	B2	22	22	178	34	34	MediumSpringGreen	00	FA	9A	0	250	154	RosyBrown	BC	8F	8F	188	143	143
DarkRed	8B	00	00	139	0	0	SpringGreen	00	FF	7F	0	255	127	SandyBrown	F4	A4	60	244	164	96
Pink colors							MediumSeaGreen	3C	B3	71	60	179	113	Goldenrod	DA	A5	20	218	165	32
Pink	FF	C0	CB	255	192	203	SeaGreen	2E	8B	57	46	139	87	DarkGoldenrod	B8	86	0B	184	134	11
LightPink	FF	B6	C1	255	182	193	ForestGreen	22	8B	22	34	139	34	Peru	CD	85	3F	205	133	63
HotPink	FF	69	B4	255	105	180	Green	00	80	00	0	128	0	Chocolate	D2	69	1E	210	105	30
DeepPink	FF	14	93	255	20	147	DarkGreen	00	64	00	0	100	0	SaddleBrown	8B	45	13	139	69	19
MediumVioletRed	C7	15	85	199	21	133	YellowGreen	9A	CD	32	154	205	50	Sienna	A0	52	2D	160	82	45
PaleVioletRed	DB	70	93	219	112	147	OliveDrab	6B	8E	23	107	142	35	Brown	A5	2A	2A	165	42	42
Orange colors							Olive	80	80	00	128	128	0	Maroon	80	00	00	128	0	0
LightSalmon	FF	A0	7A	255	160	122	DarkOliveGreen	55	6B	2F	85	107	47	**White colors**						
Coral	FF	7F	50	255	127	80	MediumAquamarine	66	CD	AA	102	205	170	White	FF	FF	FF	255	255	255
Tomato	FF	63	47	255	99	71	DarkSeaGreen	8F	BC	8F	143	188	143	Snow	FF	FA	FA	255	250	250
OrangeRed	FF	45	00	255	69	0	LightSeaGreen	20	B2	AA	32	178	170	Honeydew	F0	FF	F0	240	255	240
DarkOrange	FF	8C	00	255	140	0	DarkCyan	00	8B	8B	0	139	139	MintCream	F5	FF	FA	245	255	250
Orange	FF	A5	00	255	165	0	Teal	00	80	80	0	128	128	Azure	F0	FF	FF	240	255	255
Yellow colors							**Blue/Cyan colors**							AliceBlue	F0	F8	FF	240	248	255
Gold	FF	D7	00	255	215	0	Aqua	00	FF	FF	0	255	255	GhostWhite	F8	F8	FF	248	248	255
Yellow	FF	FF	00	255	255	0	Cyan	00	FF	FF	0	255	255	WhiteSmoke	F5	F5	F5	245	245	245
LightYellow	FF	FF	E0	255	255	224	LightCyan	E0	FF	FF	224	255	255	Seashell	FF	F5	EE	255	245	238
LemonChiffon	FF	FA	CD	255	250	205	PaleTurquoise	AF	EE	EE	175	238	238	Beige	F5	F5	DC	245	245	220
LightGoldenrodYellow	FA	FA	D2	250	250	210	Aquamarine	7F	FF	D4	127	255	212	OldLace	FD	F5	E6	253	245	230
PapayaWhip	FF	EF	D5	255	239	213	Turquoise	40	E0	D0	64	224	208	FloralWhite	FF	FA	F0	255	250	240
Moccasin	FF	E4	B5	255	228	181	MediumTurquoise	48	D1	CC	72	209	204	Ivory	FF	FF	F0	255	255	240
PeachPuff	FF	DA	B9	255	218	185	DarkTurquoise	00	CE	D1	0	206	209	AntiqueWhite	FA	EB	D7	250	235	215
PaleGoldenrod	EE	E8	AA	238	232	170	CadetBlue	5F	9E	A0	95	158	160	Linen	FA	F0	E6	250	240	230
Khaki	F0	E6	8C	240	230	140	SteelBlue	46	82	B4	70	130	180	LavenderBlush	FF	F0	F5	255	240	245
DarkKhaki	BD	B7	6B	189	183	107	LightSteelBlue	B0	C4	DE	176	196	222	MistyRose	FF	E4	E1	255	228	225
Purple colors							PowderBlue	B0	E0	E6	176	224	230	**Gray colors**						
Lavender	E6	E6	FA	230	230	250	LightBlue	AD	D8	E6	173	216	230	Gainsboro	DC	DC	DC	220	220	220
Thistle	D8	BF	D8	216	191	216	SkyBlue	87	CE	EB	135	206	235	LightGrey	D3	D3	D3	211	211	211
Plum	DD	A0	DD	221	160	221	LightSkyBlue	87	CE	FA	135	206	250	Silver	C0	C0	C0	192	192	192
Violet	EE	82	EE	238	130	238	DeepSkyBlue	00	BF	FF	0	191	255	DarkGray	A9	A9	A9	169	169	169
Orchid	DA	70	D6	218	112	214	DodgerBlue	1E	90	FF	30	144	255	Gray	80	80	80	128	128	128
Fuchsia	FF	00	FF	255	0	255	CornflowerBlue	64	95	ED	100	149	237	DimGray	69	69	69	105	105	105
Magenta	FF	00	FF	255	0	255	MediumSlateBlue	7B	68	EE	123	104	238	LightSlateGray	77	88	99	119	136	153
MediumOrchid	BA	55	D3	186	85	211	RoyalBlue	41	69	E1	65	105	225	SlateGray	70	80	90	112	128	144
BlueViolet	8A	2B	E2	138	43	226	MediumBlue	00	00	CD	0	0	205	Black	00	00	00	0	0	0
DarkViolet	94	00	D3	148	0	211	DarkBlue	00	00	8B	0	0	139							
DarkOrchid	99	32	CC	153	50	204	Navy	00	00	80	0	0	128							
DarkMagenta	8B	00	8B	139	0	139	MidnightBlue	19	19	70	25	25	112							
Purple	80	00	80	128	0	128														
Indigo	4B	00	82	75	0	130														
SlateBlue	6A	5A	CD	106	90	205														
DarkSlateBlue	48	3D	8B	72	61	139														
MediumSlateBlue	7B	68	EE	123	104	238														