

## Assignment 1, Part A: Treasure Map

(20%, due 11:59pm Sunday, September 9th, 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 repetitive computational problem; and
- Display information in a visual form.

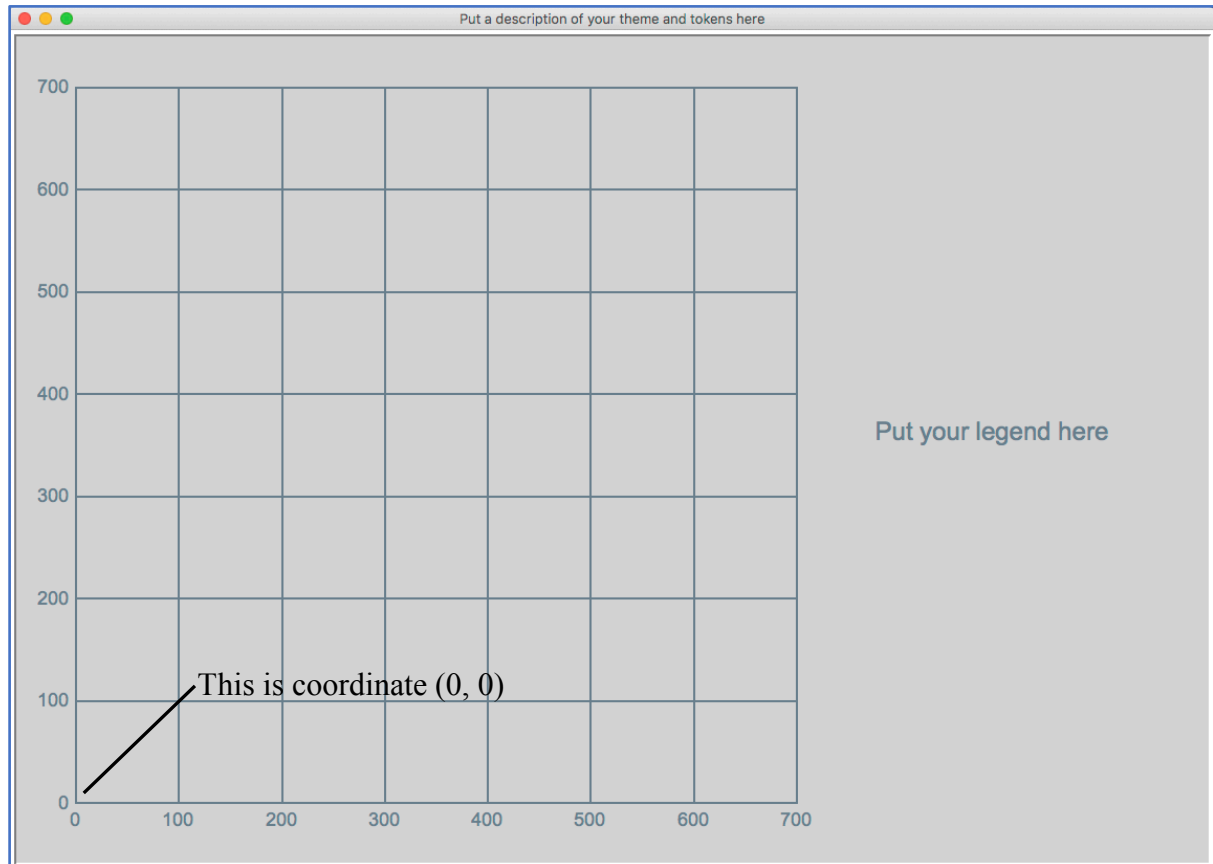
In particular, you will need to think carefully about how to design reusable code segments, via well-planned function definitions and the use of repetition, to make the resulting program concise and easy to understand and maintain.

### Goal

The short-lived *Pokémon Go* craze temporarily reignited interest in old-fashioned treasure hunts. In this assignment you are required to use Python's Turtle graphics module to draw a map summarising the outcomes of such a treasure hunt. To do so you must follow a given set of instructions, provided as a Python list, to place "tokens", representing treasures found, in a square grid defining the search area. Most importantly, the path followed will be generated *randomly*, so your solution must be sufficiently general that it can work correctly for *any possible path* through the grid.

### Resources provided

A template Python 3 program, `treasure_map.py`, is provided with these instructions. When run it creates a drawing canvas and displays a simple background image on which you will follow the instructions to draw tokens in the locations where they are found during the imaginary treasure hunt. You have a free choice in the design of the individual tokens. The default image drawn by running the provided Python template appears as shown overleaf. It consists of a simple grid representing the search field, and space for a legend in which you will describe the theme and tokens you have designed.



Note that in this Turtle canvas the “home” coordinate (0, 0) is at the bottom left, rather than the usual default of the canvas’ centre. The canvas contains a grid representing the area explored during the treasure hunt. Each of the grid’s squares measures  $100 \times 100$  pixels. The large blank space on the right is where the legend explaining the meaning of each of your tokens will be drawn.

Your task is to extend this template file so that it can draw tokens in the grid to indicate “treasures” found by following a provided path through the grid. To do so you must design five entirely distinct tokens, each of which can be drawn in any grid square (and in the legend). Your code will consist of a function called `follow_path`, and any auxiliary functions you define to support it. This function takes a single argument, which is a list of instructions specifying where to start the treasure hunt and which steps to follow. This list of instructions is created by a provided function called `random_path` which *randomly* generates the sequence of instructions, so your code must work correctly for *any possible* path through the grid!

### Data format

The `random_path` function used to assess your solution returns a list of instructions representing the steps taken during the treasure hunt. Each of the instructions is expressed as a triple (a sublist of length three). The instructions have two different forms. The first instruction in the path is always of the form

`['Start', location, token_number]`

where the *location* may be any one of five character strings, 'Top left', 'Top right', 'Centre', 'Bottom left' or 'Bottom right', and the *token\_number* is an integer from 0 to 4, inclusive. This instruction tells us in which grid square to begin our treasure hunt and the token that we find there. (Every square we visit yields a token, including the first.) For instance, an initial instruction

```
['Start', 'Bottom right', 3]
```

tells us that our treasure hunt begins in the grid's bottom-right square and that we should draw a token of type 3 there.

The remaining instructions, if any, are all of the form

```
[direction, number_of_squares, token_number]
```

where the *direction* may be 'North', 'South', 'East' or 'West', the *number\_of\_squares* is a positive integer, and the *token\_number* is an integer from 0 to 4, inclusive. This instruction tells us where to go from our current location in the grid and the token to draw in the target square. For instance, an instruction

```
['South', 3, 2]
```

tells us that the next step is to move down by three grid squares and to draw a token of type 2 there.

In addition to the `random_path` 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 path while debugging your code. However, the “fixed” paths will not be used for assessing your solution. Your `follow_path` function must work correctly for *any* path randomly generated by function `random_path`.

### Designing the tokens

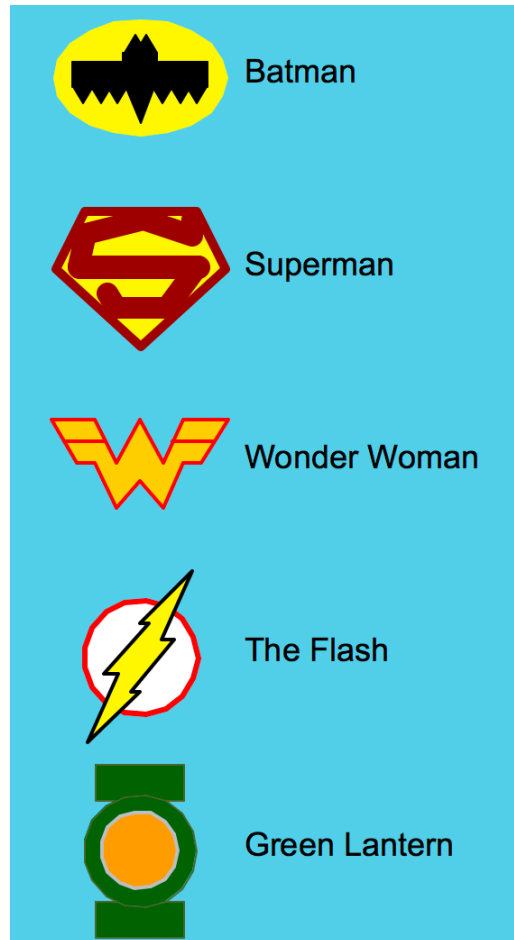
To complete this assignment you must design five *entirely distinct* tokens representing “treasures” found during the hunt. The five tokens must each fit precisely in a  $100 \times 100$  pixel square, must be drawn using Turtle graphics primitives only, must be of a reasonable degree of complexity, 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
- Businesses (banks, restaurants, IT companies, etc)
- Computer or board games (e.g., Monopoly tokens)
- Internet or cloud service providers
- Geographical sites (cities, countries or tourist attractions)
- Vehicles (cars, boats, planes, etc)
- Household objects
- Or anything else suitable for creating five distinct and easily-identifiable “tokens”

### *Illustrative example*

To illustrate the requirements we developed a solution which uses classic DC Comics superheroes as its theme. (Don't copy our example! Develop your own idea!) We wrote Turtle graphics code that could draw the following five tokens, each representing a well-known superhero.

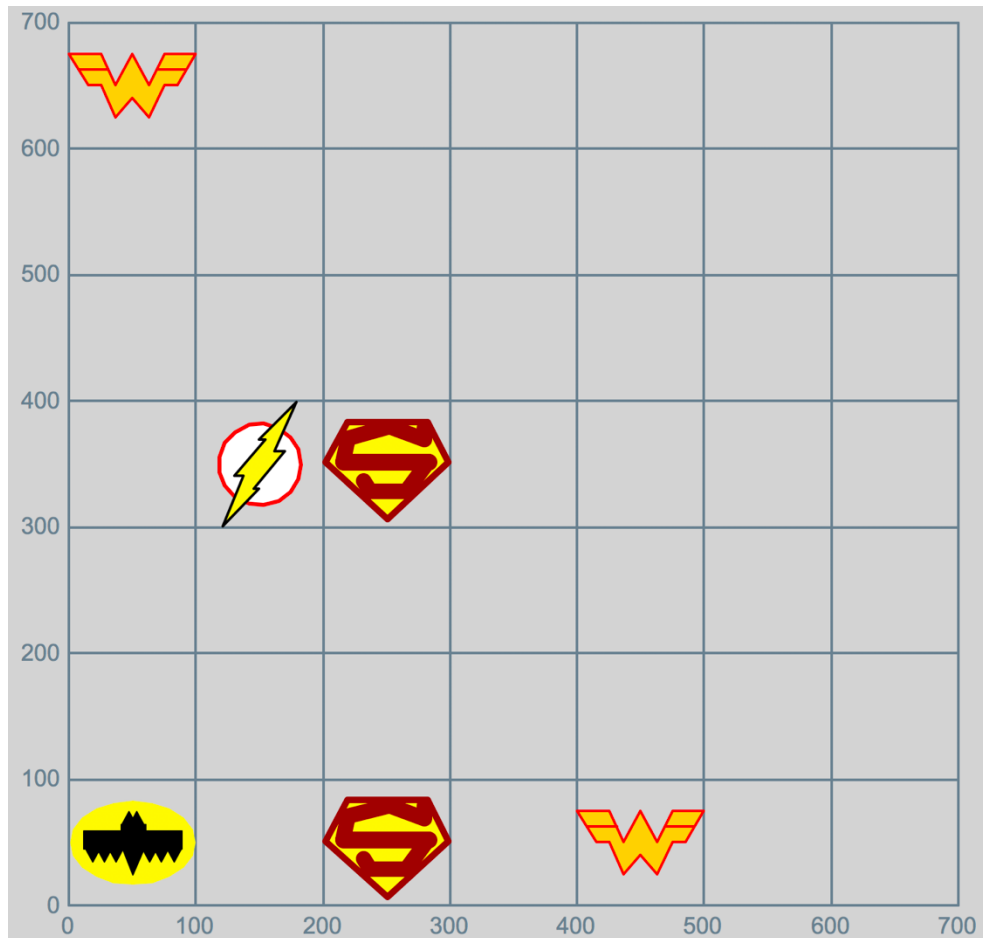


Each of these images is *exactly* 100 pixels high or wide, so will fit perfectly into one of the grid squares. In our solution we use these symbols as our “tokens”, with Batman being Token 0, Superman being Token 1, and so on, up to Green Lantern as Token 4. For your solution you need to similarly design five tokens for use as Token 0 to Token 4, inclusive.

From this basis, our implementation of the `follow_path` function can follow any plan generated by function `random_path`, drawing tokens at each step. For instance, consider the following short path:

```
[['Start', 'Top left', 2],  
 ['South', 6, 0],  
 ['East', 4, 2],  
 ['West', 2, 1],  
 ['North', 3, 1],  
 ['West', 1, 3]]
```

This requires us to start in the top-left square of the grid and draw Token 2 there, which in our case is Wonder Woman. Next we go six squares south and draw Token 0, which is Batman. Then we go four squares east and draw Token 2, i.e., Wonder Woman again. We then reverse direction and go two squares west and draw Token 1, Superman. Next we go three squares north and draw Token 1 again. The final step is to go one square west and draw Token 3, The Flash. The resulting map of the treasure hunt's discoveries in the grid is shown below.



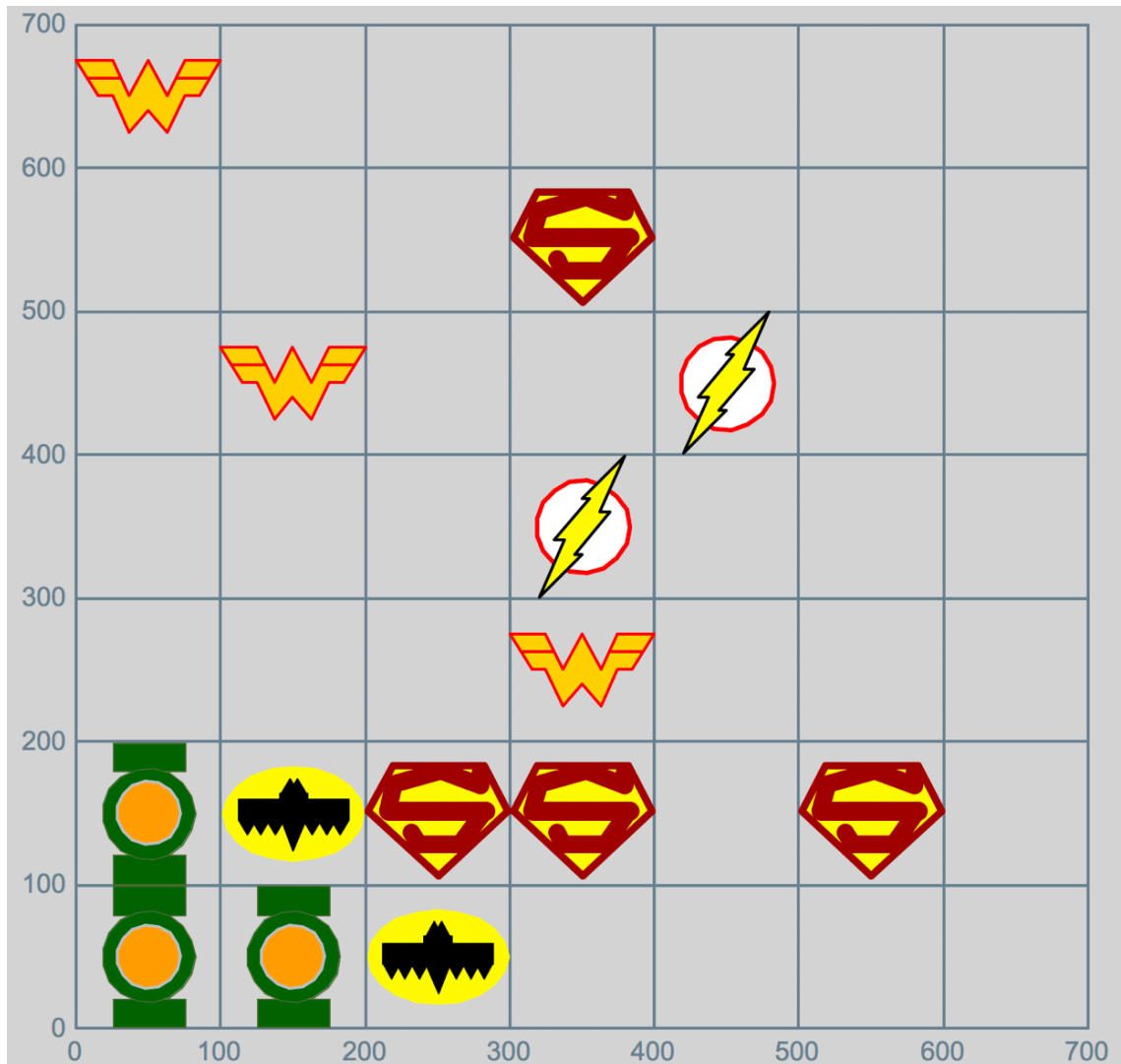
Notice that each token is of the appropriate style and fits perfectly into the correct square according to the instructions in the path provided.

As a more complex example, consider the following much longer randomly-generated path.

```
[['Start', 'Centre', 3], ['North', 2, 1],
 ['South', 3, 2], ['South', 1, 1],
 ['West', 2, 0], ['East', 4, 1],
 ['West', 3, 1], ['West', 2, 4],
 ['North', 5, 2], ['South', 6, 4],
 ['East', 2, 0], ['West', 1, 4],
 ['North', 4, 2], ['East', 3, 3]]
```

In this case we begin in the centre square with Token 3, The Flash. We then go: two squares north to draw Token 1, Superman; three squares south to draw Token 2, Wonder Woman;

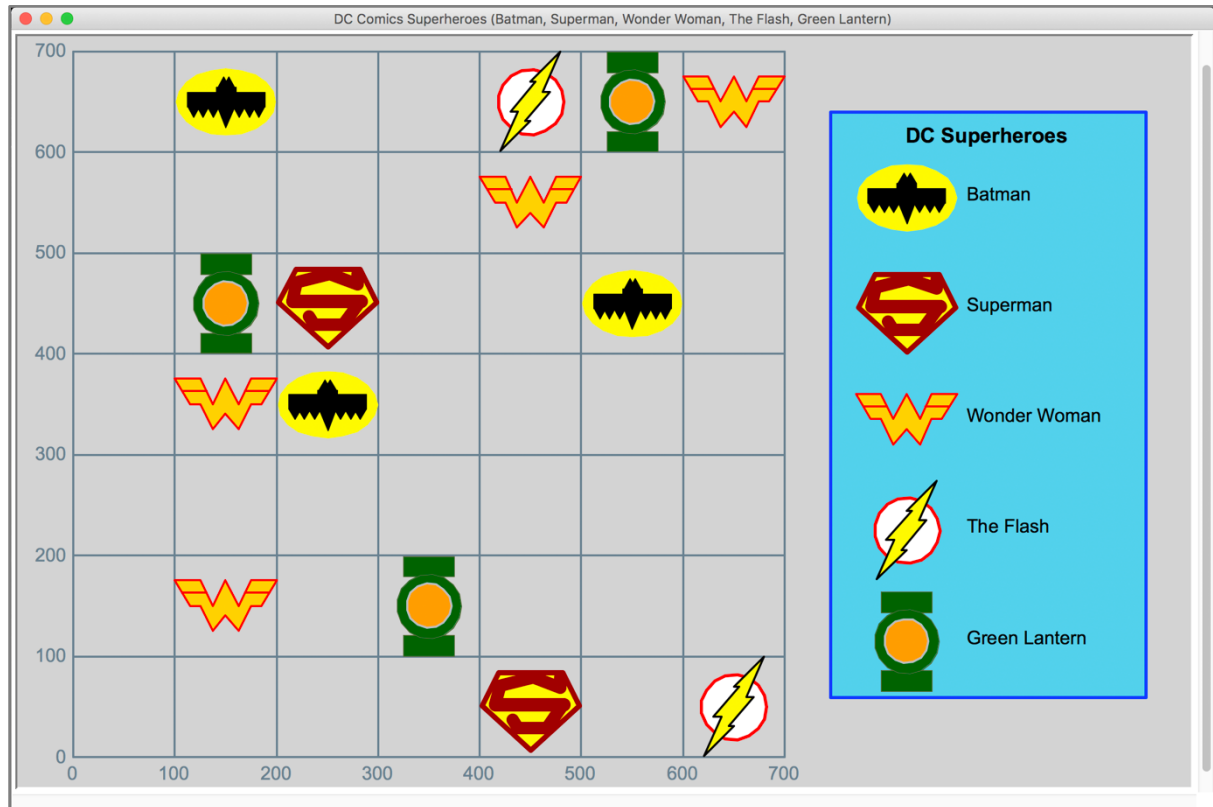
another square south to draw Token 1 again; two squares west to draw Token 0, Batman; and so on until all the specified tokens are drawn. The resulting image appears below.



As well as drawing tokens in the grid according to a provided path, the final requirement for this part of the assignment is to complete the treasure map with a legend which clearly describes the chosen theme and the meaning of each of the five tokens. This must be placed in the space to the right of the grid but is always the same regardless of the path followed.

The figure overleaf shows a complete treasure map, illustrating both the grid and legend, in this case for the following path:

```
[['Start', 'Bottom right', 3], ['West', 2, 1],
 ['North', 5, 2], ['North', 1, 3], ['East', 1, 4],
 ['East', 1, 2], ['West', 5, 0], ['South', 3, 2],
 ['East', 1, 0], ['North', 1, 1], ['East', 3, 0],
 ['West', 4, 4], ['South', 3, 2], ['East', 2, 4]].
```



### Requirements and marking guide

To complete this part of the assignment you are required to extend the provided `treasure_map.py` Python file by completing function `follow_path` so that it can draw tokens as specified by the data sets generated by the `random_path` function. Your `follow_path` function must work correctly for any values returned by the `random_path` 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 five entirely distinct tokens within a common theme (5%).** Your program must be able to draw five clearly distinct tokens, each fitting exactly into a  $100 \times 100$  pixel square. Each token must be of a reasonable degree of complexity, involving multiple Turtle shapes. It must be easy to distinguish each token from those adjacent to it in the grid.
2. **Identifying the theme and tokens (4%).** When executed your code must draw a “legend” on the right of the canvas which clearly describes your theme and indicates the meaning/identity of each of the five tokens. The legend must include a visual copy of each token and they should be in the same order as your chosen identity for them from Token 0 to Token 4. You must also put a title at the top of the Turtle drawing canvas describing your theme and tokens.
3. **Placing tokens in the grid as per any given path (7%).** Your code must be capable of drawing tokens in grid squares, exactly as dictated by any valid data set provided to function `follow_path`. The tokens must be positioned precisely within the grid

squares corresponding to the path described. The drawings of the tokens must preserve their integrity no matter where they are drawn, with no spurious additional or missing lines. Your solution for drawing the tokens must work correctly for *any* values returned by the `random_path` function.

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 say what it does. Similarly, the names of your functions and variables should be indicative of their purpose, not just “i”, “j”, 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.*

You must complete the assignment using basic Turtle graphics, random number and maths functions only. You may not import any additional modules or files into your program other than those already included in the given `treasure_map.py` template. In particular, you may not import any image files for use in creating your drawing.

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!

### ***Artistic merit – The Hall of Fame!***

You will not be assessed on the artistic merit of your solution, only the ability to create five entirely distinct tokens and follow any provided path. 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.)

### ***Development hints***

- Before creating code to draw the individual tokens give careful thought to how you can write the code so that the tokens can be drawn in any square of the grid, and in the legend. In particular, you need to avoid “hardwiring” your drawing to fixed coordinates in the drawing canvas.
- The easiest part of this assignment is the legend on the right, because it is the same regardless of the given path, so you may want to complete it first.
- If you are unable to complete the whole task, just submit whatever you can get working. You will receive *partial marks* for incomplete solutions.
- To help you debug your code we have provided some “fixed” paths. Feel free to use these when developing your program, and add additional ones if you like, but keep in mind that these data sets will not be used for assessing your solution. Your `fol-`



`low_path` function must work for *any* list that can be returned by function `random_path`.

- Part B of this assignment will require you to change your solution slightly 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 modify when the instructions for Part B are released.

### ***Deliverable***

You must develop your solution by completing and submitting the provided Python 3 file `treasure_map.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 `follow_path` 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*.

### ***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 secure. 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 have a *private* repository

that cannot be seen by anyone else. For instance, students can apply for a *free* private repository in GitHub to keep their work 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!

### ***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, September 9th, 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 deadline. 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](mailto:ithelpdesk@qut.edu.au); 3138 4000) for assistance and advice. Teaching staff will *not* 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, September 7th.

## Appendix: Some standard Turtle graphics colours you can use

Source: [www.discoveryplayground.com/computer-programming-for-kids/rgb-colors/](http://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	f0fff0	
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	66cdaa	
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	7cfc00	
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	