Assignment: SPACE MOVER
Client's Briefing Part 1 – Transcript

Hello, I'm representing your client who has commissioned you to build a program to solve a data visualisation problem. We need to interpret data sets that appear as long lists of numbers and text. Those data sets are very hard to understand, so we want to visualise their contents.

We're not *completely* sure of our final requirements yet. However, today we are delivering some basic information to allow you to start planning and developing a design.

You will need to submit your solution as a single Python file uploaded via the Canvas submission link before the agreed deadline (see Canvas). The requirements for the assignment, however, will be delivered in two parts.

Part 1

To get you started, we've provided you with a Python template file. Let's have a look at it and see how it works.

> [Shows that you need to enter your student number and name into file `space_mover.py` before it will run]

For cyber security purposes we need to know who wrote the software, so you must sign your name and student number up the top before the program will work.

> [Adds name as a string, and student number as an integer.]

Now it works by displaying a pre-fab drawing canvas, which I'll explain in a moment. You also might notice that we still get a message:

```
No data generation module available
```

in the shell window. Eventually you will need to include a file called `config.py` in the same folder as `space_mover.py` but we haven't provided you with that data generation module yet. It's commercial-in-confidence, so we'll supply it on an as-needed basis, *after* you've had time to demonstrate your technical skills by completing this first part of your solution.

So, what did we end up with? This drawing canvas.

> [Shows the initial layout of the drawing canvas]

As you can see, when the program runs, the template creates lots of cells for drawing things. This is where you are eventually going to visualise the data sets that the data generation module will supply. Each space is a square exactly 100 pixels wide and high. The complete drawing canvas is quite large, so you may need to adjust your computer's display settings to allow it to fit on the screen.

You will also see that one cell has been marked as "special" with a dot, but you don't need to worry about that at this stage. The pre-fab canvas also has some instructions for you which you should replace with various elements of your visualisation.

All you need to do now is design and draw a symbol that will completely cover the exact size of one cell. For the time being, you need to draw that symbol only in the left hand margin where indicated. This will represent a 'legend' for your final drawing i.e. a reference to the user of your program about what the symbol is or represents.

The final program will need to be able to draw that symbol anywhere on the grid – i.e. over the top of any cell on the canvas.

You have a completely free choice of what symbol to draw, but it must represent something biological, and capable of 'moving' under its own power. The symbol can be fictional or real. So it could be:

- a person,
- a bird,
- an insect,
- a reptile,
- a fish,

or whatever you like. The only constraints are that it must be non-trivial and easily recognisable by our data analysts.

To illustrate the idea, I've asked our back-room boffins to create an example, but we want you to produce your own unique solution. For their demo the boffins chose the fictional character "Bluey" as their entity!

[*Runs the sample solution for Part 1*]

Your images do not need to be this complex. However, our sample solution has the following features all of which you **must** duplicate:

a) The symbol chosen is described in the drawing canvas' title and on the canvas itself.

b) The symbol must appear instead of the text placeholder on the left margin of the canvas and will serve as a legend to tell our data analysts what it represents.

c) The symbol in the legend has a brief caption, such as a description or quotation, characterising the entity.

d) The symbol must fit precisely into the grid's cells.

e) The symbol must be: **non-trivial** (meaning it is composed of multiple shapes); and **easily recognisable** (meaning that it must be a close approximation of whatever you're attempting to draw). In particular emojis, smiley faces and other such images composed of a few geometric shapes will *not* be accepted because the individual images are much too simple.

f) For portability of your program code, the symbol must be drawn using standard Turtle graphics primitives only. You may *not* include any separate image files with your solution and only a single Python file may be submitted.

Your job is to display your symbol on the left side of our supplied canvas and to describe it appropriately. To do so you must add Turtle graphics code to the provided Python template program in the area marked "Student's Solution". All of your code must appear in, or be called from, function `start_moving` in the place marked with the **`pass`** statement.

[*Shows the relevant parts of the template code*]

You may not change any of the given code except in the main program at the bottom as indicated in the comments. You can change the *arguments* in these function calls ... but you must not change any other code in the template.

Most importantly, although the symbols and text always appear in the same place for this Part 1, in the next part of the assignment you'll need to draw the symbol multiple times in different cells on the canvas. Therefore, you should avoid "hardwiring" your drawings to absolute *x*-*y* coordinates on the canvas. You can't move a symbol after it's been drawn, so from the beginning you should allow for the future need to draw each symbol at any chosen location on the screen.

[*Ends the briefing by running the sample solution for Part 1 once again*]

I'll be back to give you more instructions in due course.