Assessment Task 1: Client's Briefing #1, Partial Transcript

Hello, we haven't met before but my name's Elton and I'm your client from *QUT Data Services.* We need *you* to help us build a program to solve a data visualisation problem our customers have. They're data analysts and they 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 need to do this fairly quickly because we have some deadlines coming up. We want the first part finished by Friday of Week 4 (that's the 24th of March) and the entire program completed by Friday of Week 7 (the 21st of April).

Today I'm just going to tell you about the first part, the part that's due at the end of Week 4. 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* `assignment_1.py` *before it will run*]
>
> [*Shows that you need to include file* `assignment_1_config.py` *in the same folder as* `assignment_1.py` *before it will run*]

Now it should work. And it does! It displays some sort of 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, but that's okay because we haven't provided you with the data generation module as yet. You'll get that after you've completed Task 1A.

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

[*Shows the initial layout of the drawing canvas*]

As you can see it contains a grid of hexagonal cells. This is where you're going to visualise the data sets. It also has some instructions for you which you should replace with various elements of your visualisation.

Hexagonal grids are good for simulations because they have the advantage that any point in a cell is the same distance from the corresponding point in all six of its neighbours. This is good for certain kinds of simulations, such as the way bushfires spread, but in our case we're going to be interested in the way an object moves around in the area defined by the hexagonal grid.

You'll also see that some of the cells have been marked as "special" with dots, but you don't need to worry about that until Task 1B. All you need to do for now is *design* and *draw* the object that will move around inside the grid. And, of course, because it's a hexagonal grid, every time the object makes a move, it can move in any one of six directions.

You have a completely free choice of what object you want to draw, but it must be something capable of moving under its own power, so it could be

- a person,

- a bird,
- an insect,
- a reptile,
- a fish,
- a motor car,
- an aircraft,
- a boat,
- a spaceship,
- a monster,

or whatever you like. The only constraints are that it must be non-trivial, easily recognisable by our data analysts, and it must be obvious in which of the six directions it's facing at any time.

To illustrate the idea I've asked our back-room boffins to create an example but, of course, we want you to produce your own unique solution. They're very strange people and they've based their example around the old nursery rhyme, "Ladybird, ladybird, fly away home! Your house is on fire and your children are gone!" Very creepy stuff! But all they've actually done is draw a picture of a ladybird as their symbol.

They've done this using Python Turtle graphics, exactly the same thing I know you've been learning about in our training courses. To show the idea they've created a very large-scale version of their design, and I'll show that to you now, but you *don't* need to do this. They're just showing off their Turtle drawing skills!

[*Shows the large-scale demo of drawing a symbol using Turtle*]

Most importantly, though, they can draw the same image pointing in each of the six possible directions we can move in the hexagonal grid.

[*Shows the same drawing pointing in another direction*]

But you don't have to draw it at a large size like this. We want you to draw your object at the scale of the cells in the hexagonal grid that we saw before, and draw all six variations of the object so that we can easily see which direction it's facing at any time when we get around to doing the full visualisation.

So that's our boffins' design for their sample solution, and you should do something at around the same level of complexity. But we don't want you to draw the object on its own like that. We want you to draw it on the canvas that's provided by our Python template. Let's have a closer look at that.

> [*Points out the various features of the canvas, including the window's title, the left-hand space for drawing three versions of the symbol and the right-hand space for the remaining three*]

It also says up the top

Replace this with your final message

but we don't need to worry about that until Task 1B.

To add your symbols to the canvas, you need to modify the `assignment_1.py` file by putting all of your code in the area marked "Student's Solution".

[*Shows the relevant parts of the Python code file*]

Put your code in the function called `visualise_data` in the place marked by the **`pass`** statement. That's where your code goes. You can also change the arguments to the function calls shown down here but, essentially, all of your drawing code goes in function `visualise_data`.

When we run our boffins' sample solution to Task 1A, this is what happens.

[*Shows the sample solution to Task 1A, pointing out its various features*]

And that's all we need to do for Task 1A.

We know that you've been practising using Turtle graphics, so drawing a symbol of that complexity shouldn't be too hard for you. But, since you need to draw it six times, facing in six different directions, it would be smart not to have to duplicate your code six times. So before you begin, think hard about how you can draw the same symbol in different places on the screen, facing different ways, but without rewriting all of your code six times.

In summary, your solution to Task 1A must have all of the following properties:

- You must describe the theme of your solution in the title on the drawing canvas.
- You must draw your object in six different orientations, in place of the instructions that were there before.
- You should label each of the six versions so that we know exactly what each one represents.
- The symbols should be a neat fit in the cells of the grid. We've put a hexagonal background on the symbols. You don't need to do that. However, the object itself should be not too big for the cells and not too small.
- Whatever symbol you design it must be: non-trivial (meaning it's composed of multiple shapes and lines); easily recognisable (so our data analysts can recognise at a glance what the object is); and it must be clear in which direction it's heading in all of its six different orientations. It looks best if your object is viewed from above, but that's not essential.
- But something that *is* essential is that for portability of your program code, the symbol must be drawn using standard Turtle graphics primitives only, with no additional imported modules beyond those available in a standard Python 3 installation, and no separate image files. You can submit a single Python file only.

After you've submitted your solution to Task 1A, at the end of Week 4, I'll give you some more instructions about what we need done for Task 1B. But I can tell you now that it will involve drawing your symbol multiple times in this hexagonal grid. So, for that reason, you don't want to hardwire your code to absolute coordinates on the screen. Before you start writing *any* Python code, think carefully about how you can draw an image at any location on the screen and pointing in any possible direction.

I know that as well as Turtle graphics you've been learning about function definitions and parameter passing, and I think that will be a big help to you in designing some *reusable* code for drawing your object in different places and pointing in different ways.

Our team is looking forward to seeing your solution at the end of Week 4, and I'll give you some more instructions after that. Good luck!