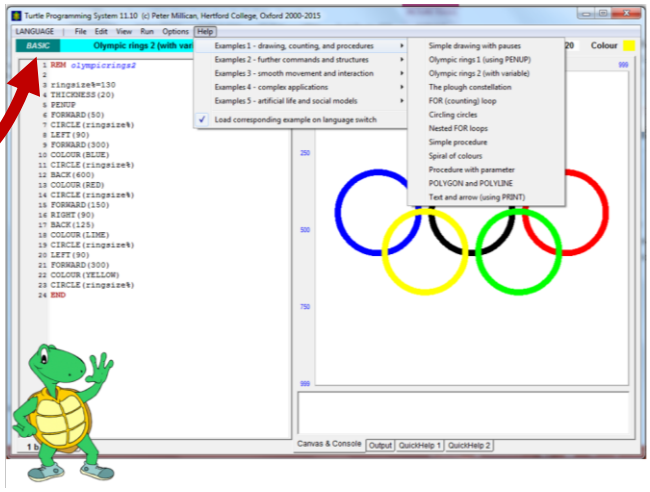
Investigating Turtle System

Turtle System lets you code in different languages. Make sure you are using the right programming language for your class.



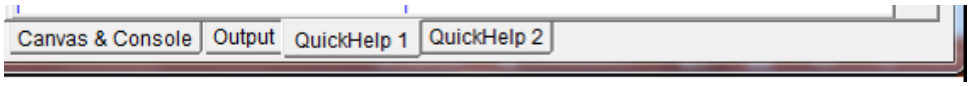
Challenge 1: Nested FOR Loops

From *Help - Examples 1*, open ***Nested FOR Loops***. Do not run this program, but study it carefully. Notice that it uses a counter controlled loop. Try to read the code and work out what it will do. ONLY when you have a prediction about what will happen should you run it.

Did it do what you thought it would?Don't worry if it didn't. Predicting the outcome is an important part of learning to code. You try to get 'inside the head' of the computer and think like it does. Often you will be surprised by the outcome. When something doesn't do what you expect, that is when real learning starts. You now have to investigate why you thought the code would behave differently. One tool in your tookit is to add pauses to the program to slow it down. Add `PAUSE(100)` commands inside and outside the inner loop. Is it clearer to understand now? Explain in your own words, how the program works.

Challenge 2: Using A Procedure

From *Help - Examples 1*, open ***Simple Procedure***. Again, study it carefully and try to work out what it will do. Line 5 has a procedure 'call'. Each time it executes this line it will run the code defined in the procedure. How often will it run the procedure? Notice it is in a condition controlled loop. It will keep repeating until `TURTD = 0`. But what is `TURTD`? It is a variable. You can find out what it represents by looking at *QuickHelp1* (select the *Basics* tab). When you start writing your own programs, *QuickHelp2* lists all the commands you can use.



When you have a prediction about what will happen run the program. Was your prediction correct? Probably not. But if not, why not? Explain in your own words what happens below.

Challenge 3: Spiral Of Colours

From *Help - Examples 1*, open ***Spiral Of Colours***. The name of the program gives you an idea of what it will do. Before running it, predict how many sides will there be in the spiral. Can you alter the code so it draws a 'squiral' instead?

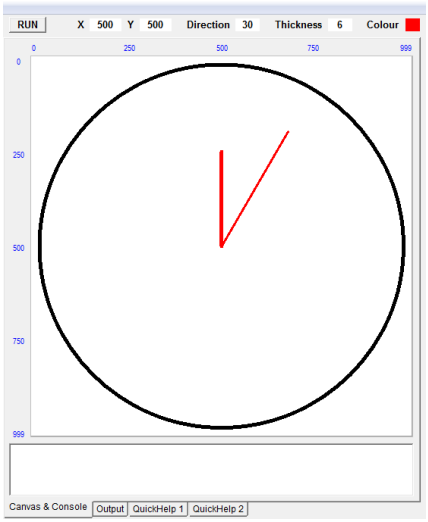
Challenge 4: A Procedure With A Parameter

The next file from *Help – Examples1* is ***Procedure With Parameter***. Like Challenge 2, it uses a procedure called Prong, but this time the definition includes a parameter. This is `len`, `LEN%` or `length` depending on the language. When the procedure is called the value of `count+100` is passed to the procedure and assigned to the parameter. The value will change on each loop.

This is really challenging. Can you predict what will happen? Try tracing out one call of the procedure first.

When you run it, what happened probably surprised you. Add pause commands so you can see how the image builds up. Can you explain what happens below?

Challenge 5: Telling The Time



In the challenges so far we've investigated some difficult programs. The purpose was to establish a method for analyzing code:

- Read the code
- Predict what will happen
- Run the code
- Reason logically about what you observe
- Use Help, or insert tests and pauses to confirm your theories

Few people write before they can read. Now it is your turn to write code, but you need to follow the same systematic method. When you write a line of code, read it, predict and observe. Fix it before moving on.

Your challenge is to write the code to display a clock face with two hands. It should tick round, moving the minute (long) hand at a regular interval. It doesn't have to wait a minute! After sixty minute ticks, the hour (short) hand should move to the next hour position.

You do not need to use procedures to accomplish this goal initially. All the clues you need to figure it out are in the sample programs we have looked at. Remember, a program is built by assembling commands using the 'Big 3' constructs.

When you have a working program, think about ways you could make the main program clearer by using procedure calls. Develop a second version to achieve this.