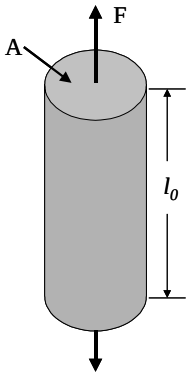
MATLAB Workshop 11 - Information Flow in Scripts

Objectives: Learn about how identifying information flow through a script can greatly simplify script design.

- **Materials Science problem**

A common activity in Materials Science (a branch of engineering that deals with development and characterization of materials for various needs) is to characterize the stress-strain behavior of a new material. Consider a cylinder of the new material with cross-sectional area A and length l_0 subjected to a force F in the axial direction, as depicted at the right. The applied force induces a *stress*, S, defined as



$$S = \frac{F}{A}$$

(what are the *dimensions* of S?). In response to the applied stress, the cylinder will stretch slightly to a new length ($l_0 + \Delta l$). The *strain*, e, is a measure of the fractional elongation (stretch) and is given by

$$e = \frac{\Delta l}{l_0}$$

(what are the *dimensions* of e?). For small applied forces, the stress and strain are linearly related as

$$S = Ee$$

where the proportionality constant, E, is known as Young's modulus or the modulus of elasticity (what are the *dimensions* of E?). Young's modulus is a characteristic (defining) property of the material and helps to define how the material deforms under stress.

Your materials science instructor wants you to develop a MATLAB script that will (1) estimate the stress, strain, and elastic modulus given F, A, l_0, and Δl, (2) determine the stress, strain, and applied force given A, l_0, Δl, and E, or (3) determine the stress, strain, and stretch given F, A, l_0, and E. The input units for values can be either all engineering units or all SI units. The answers must be reported in both SI units and the original input units. What is the difference between *units* and *dimensions*?

- **Developing a plan of attack**

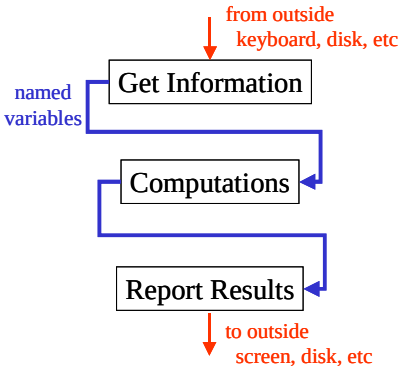
You should develop a coherent plan of attack (a program or script outline) before trying to create any new script. This requires that you do some background thinking and scratching on a piece of paper before you touch the computer keyboard. A small amount of time invested at this stage will save hours of frustration at the computer keyboard. You need to remember: **computers are stupid!** They will do exactly what you tell them to do. But, if you cannot tell them exactly what to do (step-by-step), they will complain and fuss at you and frustrate you.

A general paradigm that can be employed in designing scripts is to break the complex problem into smaller, more easily manageable tasks. Tasks are activities that are designed to achieve a single, well defined goal and, therefore, make ideal candidates for functions. A general paradigm that can be employed for most programs is

(1) get information or values needed by the script
(2) use the values to perform computations
(3) report results.

- **Information flow**

One of the most important aspects of script design is information flow. That is, how does information (values) move through the script: What information is required for any one task to perform its job? What information will result when as task executes successfully? A generic information flow diagram is shown at the right. Information that is required from the user or that is reported back to the user is shown in red. Information then flows through the script by means of named variables listed in the function input and output variable lists (shown in blue). Named variables are internal to the computer. Note that information flows into the computer only in tasks designed to get information from an external source. Likewise, information flows out of the computer only from functions designed for that purpose. Functions that are designed to perform computations should not also be asking for input values. That is improper mixing of tasks.



Carefully thinking about the information flow through the script will help greatly in the design of the individual functions. For example, with respect to the present problem, the information needs for the computation section can be identified without knowing the details of the script algorithm. Thus, a preliminary design for the computation section might be:

```
function  stress_strain_calc
```

(1) Function to (1) estimate the stress (S), strain (e), and elastic modulus (E) given F, A, l_0, and Δl, (2) determine the stress, strain, and applied force (F) given A, l_0, Δl, and E, or (3) determine the stress, strain, and stretch (Δl) given F, A, l_0, and E, depending upon user choice.

(2) Produces **S** (stress, SI units)
e (strain, dimensionless)
E (elastic modulus, SI units - case 1)
F (applied force, SI units - case 2)
delta_l (stretch, SI units - case 3)

(3) Needs **choice** (user choice for calculation)
S (stress, SI units)
e (strain, dimensionless)
E (elastic modulus, SI units - cases 2&3)
F (applied force, SI units - cases 1&3)
delta_l (stretch, SI units - cases 1&2)
init_l (initial length, SI units - all cases)
area (cross-sectional area, SI units - all cases)

(4) Algorithm
Calculations depending upon **choice**

Clearly identifying the purpose of the task, i.e., what will result from what information, lends itself to identifying the information flow. Indeed, both the information produced and information required should be able to be extracted from the purpose statement. One thing to note for **function stress_strain_calc** is that the same variable names can appear on both the input and output side of the function definition.

Looking at information flow for the output task produces

function stress_strain_results

(1) Function to display (1) stress (S), strain (e), and elastic modulus (E) given F, A, l_0, and Δl, (2) stress, strain, and applied force (F) given A, l_0, Δl, and E, or (3) stress, strain, and stretch (Δl) given F, A, l_0, and E, in SI units (case emphasis on user choice) and engineering units if original input in engineering units.

(2) Produces **Nothing** (this is a display function)!!!

(3) Needs **choice** (user choice for calculation)
units (original input units: 1 = SI, 2 = engr)
S (stress, SI units)
e (strain, dimensionless)
E (elastic modulus, SI units - cases 2&3)
F (applied force, SI units - cases 1&3)
delta_l (stretch, SI units - cases 1&2)
init_l (initial length, SI units - all cases)
area (cross-sectional area, SI units - all cases)

(4) Algorithm
Display SI unit results with language depending upon **choice**
If original input is engr units, display engr unit results in same manner as SI

Note that the requested script was to display results in both SI units (mandatory) and in engineering units if the original input units were engineering. Information required by this function is, thus, the SI values, the choice for computation (since this will affect the text associated with the display), and the original input units. Since we are not interested in producing and keeping values in engineering units, the output displays for engineering units can be handled by doing arithmetic conversions within the **disp** command. For example

```
disp('   initial length, ft')
disp( init_l/ft_to_m )
```

where the existing unit conversion function, **ft_to_m**, designed in Workshop 7 is being used to directly convert from SI to engineering units inside the **disp** function call.

Identifying the information flow requirements for the information input section of the script is rather straightforward once the information flow for the computation and display sections of the script is known. In particular, preliminary design for the input section might look like

function stress_strain_getinfo

(1) Function to (a) get user choice for calculations to perform, (b) input units, (c) depending upon choice get values for (1) F, A, l_0, and Δl, (2) A, l_0, Δl, and E, or (3) F, A, l_0, and E, and (d) if input units are engineering, convert all outgoing values to SI.

(2) Produces **choice** (user choice for calculation)
units (original input units: 1 = SI, 2 = engr)
E (elastic modulus, SI units - cases 2&3)
F (applied force, SI units - cases 1&3)
delta_l (stretch, SI units - cases 1&2)
init_l (initial length, SI units - all cases)
area (cross-sectional area, SI units - all cases)

(3) Needs **Nothing** (all input comes from outside the computer)!!!

(4) Algorithm

Get user **choice**
Get input units
Ask for input values depending upon choice. Set noninput value to 0.
If engineering units, convert all outgoing values to SI

The third step in the algorithm will need to be expanded to either an **if/elseif/else** construct or **switch/case** construct. The fourth step simply converts all values to SI regardless of choice if engineering units are used as input.

- **Algorithm design**

Once the information flow has been designed, you can go back and augment (add detail to) the algorithm design. The amount of detail to add depends upon your skill as a programmer. But even experienced programmers find more detail is better than less.

Go back and add sufficient algorithm detail to the function designs to allow you to convert the algorithm outline directly into MATLAB code.

(1) Create the individual MATLAB functions

You should have enough information to create the individual MATLAB functions **stress_strain_getinfo**, **stress_strain_calc**, and **stress_strain_results**. Go ahead and do so. Remember to test your functions as you create them with values for which you know the answers.

(2) Create the script **wkshp11**.

You now have the requisite functions to create a script that puts it all together. Create a script, **wkshp11.m**, that does the following

- displays script header information (call to **wkshp11_header**)
- gets the necessary input information (call to **stress_strain_getinfo**)
- performs the pressure calculation (call to **stress_strain_calc**)
- displays the results (call to **stress_strain_results**)

Be sure your script has an appropriate header section and variable dictionary.

Bonus: Add a **while** loop to your script so that the script asks the user whether to repeat for another set of stress-strain calculations and, if the answer is yes, repeats the input, calculation, display portion of the script.

Recap: You should have learned

- Rudimentary aspects of information flow through a script.
- Attention to information flow details is crucial to script design.
- The function/script design algorithm.
- The most important part of the function/script design algorithm is the purpose statement.
- Scripts do not need to be designed in “linear order”, that is start to finish.