Introductory Project

This is really aimed at people who have no Python programming experience. Before you try to complete these programs you really need to have a look at a basic Python programming tutorial.

The projects are named like 'project1a*partial.py*', '*project1b*partial.py'. The 'a' project comes before the 'b' project etc. and each successive project usually involves extending the previous one in some way.

Once you have been given access to the working project code, you will find the programs called 'project1a_working.py'. There will be one 'working' program to go with each 'partial' one. The idea is that you have a go first at get the partial code to run properly and then have a look at the 'working' code to get some alternative ideas, or if you get stuck (which I'd say is likely, especially at first).

There is NO BENEFIT to spending a lot of time *stuck*, especially when just starting out. In the introductory projects, if you hit a roadblock, give it 10 minutes of research or brain time, then ask someone or failing that just look at the 'working' code file. Make sure you can follow the working code. Using the debugger (called pdb or python debugger) to step through the code one line at a time is helpful and you'll be shown how to do this.

In every '_partial.py' file parts of the code that you need to work on are marked with a #* symbol. This means the are just 'comments', but you need to replace them with python instructions.

Python programmers use a lot of comments to remind them and others what the code does. Python uses the # sybol for this. Comments can also be given inside a triple double quote. These are 'docstrings' and help python to create an automated help or documentation system.

Project 1a: project1a_partial.py

The first project is a super basic calculator for a piece of physics called Ohm's law. Ohm's law has three numerical quantities called voltage, current and resistance. They are give the symbols V, I and R respectively. (Don't ask why I). Ohm's law says that V = I*R. The * asterisk symbol means multiplication.

You need to edit lines 9, 16 and 18.

Delete the comment on line 9 and replace it setting the resistance to say 100. The resistance will be in ohms, but you just need the number here.

Lines 16 and 18 are started for you. You need to delete from the #* and replace with some python code.

When you run the code it should just print `V = 10 volts`

Things to note.

- Inside strings for printing you will see \n and \t. These are little text controls, so \n forces a new line. \t forces a tab and is for lining things up a bit like a table.

Project 1b: project1b_partial.py

The previous program was pathetically pointless. Let's upgrade to something merely almost pointless.

Previously we 'hard-coded' the resistance and current values. So to calculate different values you had to edit, save and reload the program. Our upgraded program is going to ask the user to type in some numbers when it runs.

```
number = input("Hi please type a number ")
```

The above python code will wait for you to type and hit ENTER. Whatever you type will be returned as a string and stored in variable 'number'. You can type numbers, letters or even a dollar sign. Number will be a python *string*, which is not a number at all!

To force python to turn this in to an actual number we must use the float() function: `number = float(input("Hi please type a number ")`

Float refers to floating point or decimal numbers. If you ever want a whole number try the int() function.

You need to edit lines 10 and 12. Replace the comment with python code. Line 10 should look very similar to line 9. Line 12 should start with 'voltage = '

Things to note.

- In python a single equals sign means "store in a variable named on the left"
- The input() function lets the user type in something. It could be anything! Numbers, letters, symbols or a mixture.
- The float() function tries to interpret a string as if it was a decimal. If in doubt it will opt for 0.0

Project 1c: project1c_partial.py

This next upgrade creates something almost believably useful. (A spreadsheet could do this more nicely though).

We will ask for a resistance value and then create a little table of selected voltages and currents using Ohm's law.

You will need to edit lines 14 and 15 first. Then also lines 30 and 31. Lines 14 and 15 should look very similar to 12 and 13. The idea is to get two values

for current and create a table over these values. An example should make it clearer.

```
Please enter a resistance in ohms: 200 Please enter a lower
value for current in amps eg. 0.1: Please enter an upper value
for current in amps eg. 1.0: .5 Please say how many rows you
want in the table 5 Current/A Resistance/ohms Voltage/V
--------- --------------- --------- 0.1 200.0 20.0 0.2 200.0
40.0 0.3 200.0 60.0 0.4 200.0 80.0 0.5 200.0 100.0
```

On line 30, the current value will be `current = current_start + step_size*i` The for loop on line on line 29 supplies a sequence of values for i, starting from 0 and increasing 'number*of*values'. On line 31, work out the voltageOn line 30, the current value will be `current = current_start + step_size*i` The for loop on line on line 29 supplies a sequence of values for i, starting from 0 and increasing 'number*of*values'. On line 31, work out the voltage.

Things to note

- you can see on line 16 that this time we *need* a whole number, hence the use of the int() function wrapped around the input().
- On line 19 and 22 we have examples of *conditional execution* or the *if* statement. The tabbed code only happens IF the if statement condition is TRUE.
- Line 29 has TWO things of interest. First is the range() function. What this does is create a list of numbers starting from 0 and running up to number*of*values. Secondly you'll notice the *for* loop. This repeatedly executes the tabbed in code one time for each value of produced by the range() function. We have told the for loop to give each value the symbol *i* as it carries out the tabbed code.

Project 1d: project1d_partial.py