Exercise 1: Getting Started

We will enter and execute this program. The assumption is that you will make typing errors and receive error messages from the Python interpreter. You will have to find and correct these errors. It is important to get used to messages produced by the interpreter. You will get a lot of them. We will discuss the various facets of this program after you get it entered and it is working.

```
# This program demonstrates assignment statements, comments,
# variable names, the print function, math operators and how
# to continue a long statement

gas_in_tank = 4.5  # We have 4.5 gallons in the tank
miles_to_destination = 400
my_money = 12.37
your_money = 14.63
our_money = my_money + your_money
gas_price = 2.50  # Gas costs $2.50 per gallon
gallons_bought = our_money / gas_price
miles_per_gallon = 27.5
distance_we_can_drive = (gallons_bought + gas_in_tank) * miles_per_gallon

print(miles_to_destination, 'miles to drive')
print('We can cover', distance_we_can_drive,  # Continue a long statement
      'miles with the gas we have')
# print('Can we make it?')
```

Once you are finished and the program is working, remove the hash tag/pound sign from the last line and run the program again. Observe the change.

Exercise 2: Printing and Strings

The print function is your friend. It is usually the first tool you use in solving problems with your program. This exercise gets you to use the print function in a number of different ways. It also uses a number of control characters to indent or go to a new line. It shows different ways to enclose a string and how the characters used to enclose a string can be used in the string itself.

```
""" Printing

The print function has  a number of options shown in this program
"""

anum = 3.45
bnum = 6
print('Print variables', anum, bnum)
print('Print an expression', anum + bnum)

print('Print variables', anum, bnum, end = ' ')
print('Print an expression', anum + bnum)

print('Print variables', anum, bnum, sep = '->')
print('Print an expression', anum + bnum, sep = ' *-* ')

print('Print variables\n', anum, bnum)
print('Print an expression\n', anum + bnum)

print('Print variables\n', '\t', anum, '\t',  bnum)
print('Print an expression\n', '\t', anum + bnum)

print('I am 5\' 10" tall')
print("I am 5' 10\" tall")
print("""I am 5' 10" tall""")
print("""
I am a python
programmer in
training
""")
```

You should see the following results:

```
Print variables 3.45 6
Print an expression 9.45
Print variables 3.45 6 Print an expression 9.45
Print variables->3.45->6
Print an expression *-* 9.45
Print variables
 3.45 6
Print an expression
 9.45
Print variables
         3.45    6
Print an expression
         9.45
I am 5' 10" tall
```

```
I am 5' 10" tall
I am 5' 10" tall

I am a python
programmer in
training
```

Included below are some of the more common escape sequences. The first three are used to include characters in a string that otherwise might cause a different result. Unix and OS/X systems produce a \n to cause a line feed (LF). Windows systems tend to produce a \r\n to cause a linefeed. Either one works fine in Python. For a more complete list of Python escape sequences with examples of what they do, see:
https://www.quackit.com/python/reference/python_3_escape_sequences.cfm

Collectively, all of these control characters plus the space are referred to as whitespace.

Escape	What it does.
`\\`	**Backslash (\)**
`\'`	**Single-quote (')**
`\"`	**Double-quote (")**
`\f`	**ASCII formfeed (FF)**
`\n`	**ASCII linefeed (LF) or newline**
`\r`	**Carriage Return (CR)**
`\t`	**Horizontal Tab (TAB)**

Exercise 3: Keyboard Input – Strings and Numbers

The input() function will get data from the keyboard. If you put a string inside the parentheses, It will be written to the operators screen before it accepts input from the keyboard. This allows you to direct the operator as to what they should enter.

```
""" Keyboard Input

The input() function will get data from the keyboard.
"""

name = input('Please enter your name: ')
age = input('Please enter your age in years: ')
# The age entered must be converted before being used in calculations.
age_in_years = int(age)
print(name, 'has been eligible to drive an automobile for',
      age_in_years - 16, 'years') # Note the continuation method used
```

An example of the expected output is as follows:

```
Please enter your name: Bob Smith
Please enter your age in years: 42
Bob Smith has been eligible to drive an automobile for 26 years
```

There are two functions used to convert numbers received as strings; int() and float(). The difference between the two is that float() allows a decimal point in addition to digits while int() doesn't. Also, notice the continuation used in the print() function. It follows the rules described in PEP 8. The following extract is from that document:
The preferred way of wrapping long lines is by using Python's implied line continuation inside parentheses, brackets and braces. Long lines can be broken over multiple lines by wrapping expressions in parentheses. These should be used in preference to using a backslash for line continuation.

You should read the details in PEP8 and any other topic in the contents that covers an area with which you have familiarity.