

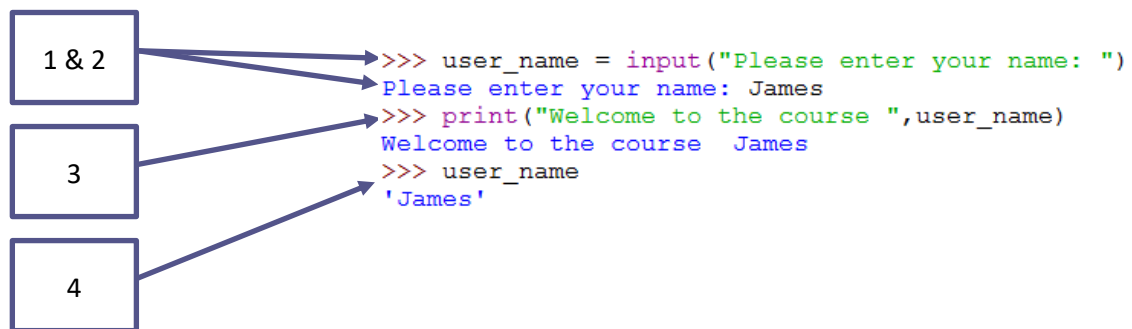
2. VARIABLES AND ASSIGNMENT STATEMENTS

In computer programming, a LITERAL is any type of data that represents a VALUE, e.g. a number or text. Variables are reserved memory locations to store values. **This means that when you create a variable you reserve some space in memory.**

The data used in a program **must be stored** in main memory while the program is running. We use names for these areas in memory so we can refer to them in our programs.

- When we create a variable, the name we use refers to the location of the data item in memory.
- Each variable will have a data type (see Data Types). The data type determines what we can do with the data and how much space it needs in memory.

We can create a variable, store the data entered and use that data again. Look at the example below:



1. A variable 'user_name' is created and the value entered from the input() function is ASSIGNED to the variable.
2. The value of 'user_name' is now James.
3. The print() statement uses the variable to output a message to screen.
4. The value of the variable can be checked by typing it and pressing the enter/return key.

2.1 VARIABLE NAMES

When we create variables and choose names for them there are some basic rules to follow:

1. Variable names must NOT have spaces
 - a. e.g. my name ☐ myName ☒ my_name ☒ my_Name ☒
2. Variable names must not use any reserved keywords (see below)
3. Choose a consistent naming style (and stick to it!)
4. Must begin with a letter (a–z, A–Z) or underscore (_)
5. Other characters can be letters, numbers or _
6. Variable names are case-sensitive (this means you must use the same CAPITALS or lower case wherever you use the variable name)
7. Use a name that makes sense (in the context of the program)
 - a. e.g. student_Name **NOT** sn

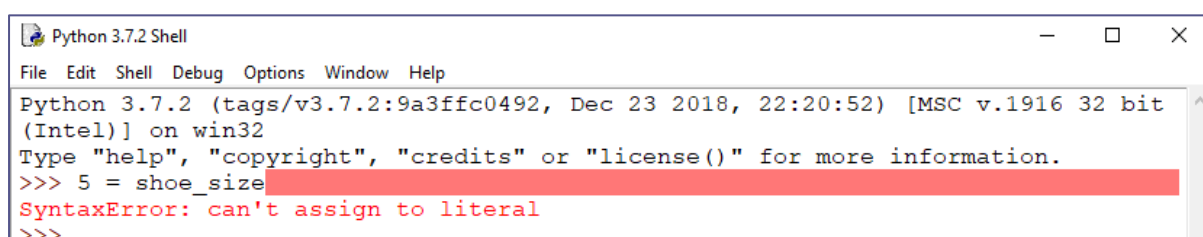
The reserved keywords have a special meaning and purpose in Python. You can check what they are by typing help ('keywords') in the Python interpreter window.

```
>>> help('keywords')

Here is a list of the Python keywords. Enter any keyword to get more help.

False          def           if            raise
None           del           import        return
True           elif          in            try
and            else          is            while
as             except        lambda        with
assert         finally       nonlocal      yield
break          for           not
class          from          or
continue       global        pass
```

When we ASSIGN a value to a variable it is important that the variable name is created BEFORE the value is assigned:



The screenshot shows a Python 3.7.2 Shell window. The title bar says 'Python 3.7.2 Shell'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following code and error message:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 5 = shoe_size
SyntaxError: can't assign to literal
>>>
```

Python returns a syntax error as the rules about variable names have not been followed. You will encounter this error message regularly as you develop your skills and knowledge using Python.

When we assign values to variables, we can use the variable names in calculations instead of the values. As you develop more programming skills you should try to follow this method in your work.

```
>>> #Calculate the area of a rectangle
>>> length = 13.6
>>> width = 5.7
>>> area = length * width
>>> print(area)
77.52
>>>
```

2.2 CONSTANTS

A constant is a value that once declared in the code remains the same and cannot be changed. Unlike many other programming languages like C++ or Java, there is no easy feature in Python to create a constant. At GCSE level we can use the PEP 8 guidelines and write the name of a variable to be used as a constant in all capitals.

For example:

```
PI = 3.14159

radius = float(input("Enter circle radius: "))
area = PI * radius**2

print(area)
```

ASSIGNING MULTIPLE VALUES

Python also allows us to make our code more **efficient** by allowing multiple assignments of values to variables:

```
>>>
>>> shoe_size,height = 5,157.8
>>> print("My shoe size is ",shoe_size,"and I am ",height,"cms tall",)
My shoe size is 5 and I am 157.8 cms tall
>>>
>>> shoe_size,height = height,shoe_size
>>> print("My shoe size is ",shoe_size,"and I am ",height,"cms tall",)
My shoe size is 157.8 and I am 5 cms tall
>>>
```

The variable names are separated by commas; as you can see they print in the correct order. In the second **assignment statement** these have been swapped around.

There are three different DATA TYPES shown in the example above:

1. The variable **shoe_size** is an INTEGER data type. This is a whole number.
2. The variable **height** is a REAL or FLOAT data type. This is a number with a fractional part.
3. The text surrounded by speech marks inside the brackets of the print function is known as a STRING.

Look at [Chapter 3: Data Types](#) for more information.

EXERCISE 02: VARIABLES AND ASSIGNMENT

1. Which of these variable names and assignment statements will NOT cause a syntax error?

```
# variable names and assignment statements

continue_list = 15          #(a)
red Counter = True          #(b)
154.2 = length              #(c)
_1234 = 'password'          #(d)
my_best_friend = "Thomas"   #(e)
```

2. Write a program to calculate the area of a circle with a radius of 6.7 cm and print the result of the calculation.
 - a. Set the value of $\pi = 3.142$
 - b. The area of a circle is $\pi \times \text{radius squared}$
3. Write a program to store the following variables:
 - a. Forename
 - b. Surname
 - c. Age
 - d. Height

Use one print function to display the information in a sentence.