Week 2: Variables and programs

Intended learning outcomes

By the end of this class, you will be able to:

- define, assign, and print variables;
- write and edit basic Python programs.

Remember that when you see a line of code with >>> below, you should type in the code and run it immediately. When you see a block of code without >>>, you should type in all the code before running it. If you can't remember how to access Google Colab, please check the Week 1 worksheet.

1 Variables

Last week, we used Python as a calculator to perform simple arithmetic. This week, we begin seeing how to use Python as a programming language.

A fundamental notion in programming is that of a *variable*.

Concept 1. Roughly speaking, a **variable** in Python is a name or label that refers to a value stored in the computer.

You can assign values to variables by typing something that looks like `variable_name = expression`. Python first works out what the expression is and stores it, then uses `variable_name` to refer to that stored value. You can reveal the value of a variable using the **`print`** command. Guess what will happen in the following each time before you press 'Enter'.

```
>>> number = 2
>>> print(number)
>>> new_number = number**2 + number/10
>>> print("The new number is", new_number)
```

The command `number = 2` stores the value 2 and uses the label `number` to refer to it.

Variables don't have to refer to numbers. They can be text, and also other things as we will see in the next few weeks. Text needs to be enclosed in inverted commas.

```
>>> breakfast_option = "spam and eggs"
>>> print("You have ordered", breakfast_option)
```

When writing code it is useful to remember the following basic maxim of programming:

Code is written once but read many times.

This means that you will spend more time reading code than writing it, so you should aim to make your code easy to read. There are various conventions which try to help with that. You are expected to follow these conventions.

Style convention 1. When assigning variables, put a *single* space either side of the symbol `=`. So, showing each space explicitly, you should type `x␣=␣2` rather than `x=2`.

The Python symbol `=` is *not* the same as a mathematical equality. It is an *assignment operator*. In some languages, such as R, you would type `x <- 2` and in some, such as Maple, `x := 2` to show the two sides have different roles.

You cannot swap the left and right hand sides over: for instance, you cannot enter `x**2 + x = y` without Python complaining. Try it and see what

happens! The left hand side must be just the name of a variable (or several names).

One thing that Python beginners sometimes find strange is that you can type something like `x = x + 1`. This wouldn't make sense if it were a mathematical equation, but it is not a mathematical equation. Remember, Python first evaluates the thing on the right and stores it, and then uses the name on the left to refer to it (forgetting, if necessary, any previous use for that name). In this case, Python takes the current value of `x` and adds `1` to it, stores the result, and then uses `x` to refer to this new value.

```
>>> count = 3
>>> count = count + 1
>>> print(count)
```

Sometimes you will want to switch two variables round. You might try to switch the values of `x` and `y` with the following code. Write down what you think the output will be before typing it in.

```
>>> x = 5
>>> y = 8
>>> x = y
>>> y = x
>>> print("x =", x, " and y =", y)
```

Is the output as you expected? If not, it is important for you to figure out why not. In that case, try inserting some print statements at each stage so you understand what the computer is actually doing: type in the following. (On the fifth and seventh lines you can use the up arrow on the keyboard to repeat the third line without retyping it.)

```
>>> x = 5
>>> y = 8
>>> print("x =", x, " and y =", y)
>>> x = y
>>> print("x =", x, " and y =", y)
>>> y = x
>>> print("x =", x, " and y =", y)
```

Hopefully, that should make things clearer. If you still don't understand, then ask a demonstrator.

Exercise 2.1. Can you see how to swap the values of `x` and `y` using a third, temporary variable `z`? (It may help to think how you could swap items from two boxes if you're only allowed to move a single thing into another box; you might need a third box.)

We can swap two variables by using a nice feature of Python called **parallel assignment**, also called **multiple assignment**. Parallel assignment allows you to define two things at once.

```
>>> x, y = 5, 8
>>> print(x, y)
```

Style convention 2. Just as in normal text, in Python a comma should have no space in front of it and a single space after it, so type "`x, y = y, x`" and not "`x,y = y,x`" nor "`x , y = y , x`".

In parallel assignment, Python evaluates *all* of the right hand side *before* assigning the labels on the left hand side . This means that you can swap two variables without the use of any extra variables.

```
>>> x, y = y, x
>>> print(x, y)
```

You can do more interesting things with parallel assignment. Write down what the values of `x` and `y` will be when the following commands have been executed.

```
>>> x, y = 5, 8
>>> x, y = y, x + y
```

Use a **`print`** command to check your answer.

2 Variable names

Variable names can contain

- letters (a–z and A–Z),
- numbers (0–9),
- the underscore (_).

However,

- variable names should not start with a number,
- variable names starting with an underscore have a special meaning.,
- variable names should not be one of the 33 *Python keywords* such as **`if`**, **`for`**, **`while`**, etc. (Search the internet "python keywords" for a complete list: you should make sure that the list is for Python 3 and not Python 2.)

Style convention 3. Variable names should be in lower case with each word separated by an underscore.

Variable names should also be meaningful.

```
lucky_number = 7
height_of_statue = 175.38
secret_password = "dead parrot"
```

You are advised to adopt this convention for your files on this course as well: e.g. `factorial.py`, `big_primes.html`, `prime_numbers.tex`. Note that putting spaces in file names of programs can cause problems.

Exercise 2.2. In a code cell, define a variable whose value is your first name and a variable whose value is your last name. Make sure that your variable names are meaningful and adhere to the above conventions. Issue a print command using these two variables, which greets you, for instance saying `Hello Joan Smith`.

3 Editing programs

Now you will type in some simple programs. To make it easier to describe particular lines in a program, we will refer to line numbers, which you can show in Google Colab by pressing `Ctrl+M+L`. Note that the following code snippets don't include the symbols `>>>`, so you should type in all the lines before pressing `Ctrl+Enter` to run the code.

In a code cell, type the following:

```
1 name = input("What is your name? ")
2 count = 1
3 while count <= 10:
4     print("Hello", name)
5     count = count + 1
```

The editor should automatically insert indentation of four spaces on lines 4 and 5. (These will be numbered differently if you started typing on line 8!) This indentation is part of the program. Before you continue, have a think about what this program will do. Write down what you think will happen when you run the program. Now run the program. Is the output what you predicted?

Exercise 2.3. Modify your program so that it also asks you for a number and greets you that number of times. (To input an integer you need to use the `int()` function, as in the programs in the lecture notes.)

Enter the following script, which was demonstrated in the lecture, in a code cell and run it.

```
# A program to calculate factorial of number entered
n = int(input("Enter the number you want "
              "to calculate the factorial of: "))
factorial = 1
i = 1
while i <= n:
    factorial = factorial * i
    i = i + 1
print(n, "! = ", factorial)
```

Check that it gives the correct answer for 4! and 5!. What is 200! ?

Exercise 2.4. The nth *triangular* number is defined to be the sum of the first n integers: $1 + 2 + \cdots + n$. Calculate the second and third triangular numbers by hand. Edit the above program so that it calculates the nth triangular number rather than the nth factorial. Run the program and check that it gives the right answer for $n = 2$ and $n = 3$. What is the 200th triangular number? There is a simple formula for the nth triangular number (find it on the internet if you don't know it): check that it gives the same answer as your program for $n = 200$.

Homework

If you have any problems with the homework then you can seek help in the following ways, both accessible via the module's Blackboard site:

- ask on the discussion board;
- book an appointment during Dr Fletcher's office hour.

1. You should finish off this worksheet before the next lab.

2. (*Quick review: a good test of understanding!*) Look at the following two pieces of code and write down what you think the output will be.

```
>>> x = 13
>>> y = 21
>>> x = y + 10
>>> y = x + 10
>>> print("x =", x, " and y =", y)
```

```
>>> x = 13
>>> y = 21
>>> x, y = y + 10, x + 10
>>> print("x =", x, " and y =", y)
```

Now type in the code and check your answer.

3. (*Assessed homework.*) This week's homework involves modifying your triangular number program so that it asks for a number n and calculates the sum of the first n reciprocal squares. To access the homework, you must navigate to the following webpage:

`https://aim.shef.ac.uk/moodle/mod/quiz/view.php?id=423`

Your code will be run with some test values. If it gives the correct answer you will get a mark, if it does not then you will get no mark. You can submit your homework anytime from now til 2pm on Monday of Week 3 (11th October).

4. (*Optional, but recommended if you found the previous questions easy.*) The Fibonacci sequence is a sequence of numbers which starts

$$1, 1, 2, 3, 5, 8, 13, \ldots$$

and is defined in the following way. The first two numbers are both 1; each of the following terms is defined to be the sum of the previous two terms.

Calculate the first ten terms in the Fibonacci sequence by hand. This helps you to understand the steps in the algorithm.

Write a program to print out all terms in the Fibonacci series that are less than 1000. My program is five lines long, but don't worry if yours is quite a bit longer.

If you found that straightforward, then also do the following. Make a copy of your program and modify it so that it prints out the ratio of consecutive terms of the series: instead of printing out $1, 1, 2, 3, 5, \ldots$, it should print out $1, 2, 1.5, 1.666666, \ldots$. These ratios should look like they are tending to a single number. Use the internet to find out what this number should be.