Week 3: Variable types and `while` loops

Intended learning outcomes

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

- interrogate and convert between Python variable types;
- construct Boolean expressions using comparison operators;
- write while loops to perform repeated commands.

Remember that when you see a line of code with `>>>` 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.

1 Python variable types

Each variable in Python has a *type* (also called a *class*) that affects what you can do with it. Here are some examples of types; there are many more.

Integer: whole number, e.g. `7`
Float: decimal approximations to real number, e.g. `175.38`
Boolean: truth value, e.g. `True`
String: piece of text, e.g. `"parrot"`
List: list of other variables, e.g. `[7, 175.38, "parrot"]`

You can use `type()` to find out the type of a variable.

```
>>> approximation = 175.38
>>> type(approximation)
>>> secret_password = "dead parrot"
>>> type(secret_password)
```

Different types have different operations that can be performed with them. This week you will look at the integer, float and Boolean types. You will see strings and lists in future weeks.

2 The integer type: `int`

Integers are whole numbers and are stored exactly. You can do the usual operations with integers: for multiplication, you use `*`; for raising to a power, you use `**`; and for the maximum and minimum of a set of numbers, you use `max()` and `min()`.

```
>>> a = 7
>>> b = 2
>>> print(a * b, a + b, a - b, -a, a**b)
>>> print(max(a, b), min(a, b))
```

In general, if you divide two integers you will not get an integer. For example, 7/3 does not give a whole number answer. Python provides an integer division operator `//` and an integer remainder operator `%`.

```
>>> print("7/3 is", 7 // 3, "with remainder", 7 % 3)
```

Exercise 3.1. Write down the answers to the following questions, using Python to help you find the answer.

- How many digits are there in 2^{100}? (You can count the digits by hand; we will see a better way of doing this next week.)
- What is the remainder when 3^{11} is divided by 11? (This is an example of Fermat's Little Theorem; google it if you are interested.)

3 The floating-point type: `float`

Floating-point numbers are approximations to real numbers. These are not stored exactly like integer variables. You will notice this if you do high-precision calculations. Python has other variable types that can handle high precision better, but floating-point numbers are generally fine to use as long as you remember they are not exact.

```
>>> a = 7.2
>>> b = 2.5
>>> print(a * b, a + b, a - b, -a, a**b)
>>> print(max(a, b), min(a,b))
```

You can often convert between between different types. For instance `int()` will turn a float into an integer and `float()` will turn an integer into a float.

```
>>> a = 3
>>> b = float(a)
>>> c = int(b)
>>> print(a, b, c)
>>> print(int(5.2), int(5.99))
```

Exercise 3.2. (i) How can you tell from the output that `b` is a float and not an integer? (ii) How does `int()` round a number? Does it round down? Round up? Round to the nearest? Forget the decimal part? What do you think `int(-7.8)` will be? Check your answer.

Here's a program that converts a temperature in Celsius to a temperature in Fahrenheit, except that I've missed out the way to convert from one to another. Type in the following program and put the correct expression where `correct_expression` is.

```
temperature_in_C = float(input("Temperature in Celsius: "))
temperature_in_F = correct_expression
print("Temperature in Fahrenheit is", temperature_in_F)
```

Run your program and check that it gives the correct answer of roughly 98.6 °F when you input the temperature of 37 °C.

Exercise 3.3. Write a program that asks for your height in metres and your weight in kilograms and prints out your body mass index (BMI). Check that your program gives a BMI of about 23.7 for a height of 1.78 m and a weight of 75 kg.

We can show the difference between `int` being exact and `float` being an approximation by calculating 12^{285} either as an integer or as a float.

```
>>> print(12 ** 285)
>>> print(12.0 ** 285)
```

The output for the second statement is `3.686847070674066e+307`, which is Python's way of writing $3.686847070674066 \times 10^{307}$. This is only an approximation to the exact, first answer. Floating-point numbers are stored to about 17 significant figures of decimal.

4 The Boolean type: `bool`

We understand what it means for a statement to be true or false: "Paris is the capital of France" is true; whereas "$7 > 21$" is false. The terms 'true' and 'false' are known as truth values and you will study these in the logic part of MAS114.

In Python, truth values are written capitalized as `True` and `False`. A Boolean variable is one whose value is either `True` or `False`. (They are named after George Boole, one of the founders of modern logic.)

```
>>> a = 10 > 3
>>> print(a)
>>> type(a)
>>> print(7 > 21)
```

The assignment operation above looks a bit strange! It is saying "let a be the truth value of the statement `10 < 3`", which is clearly false. Note the difference between `print(7 > 21)` and `print("7 > 21")`.

Boolean expressions are often built up using comparison operators. Here are the standard ones.

Symbol	Meaning	Notes
==	is equal to	This is *not* the same as =
!=	is not equal to	
>	is greater than	
>=	is greater than or equal to	This is *not* the same as =>
<	is less than	
<=	is less than or equal to	This is *not* the same as =<

It is important to note that `a = 3` and `a == 3` mean different things and should not be mixed up. The first statement is *assigning* the value 3 to a, while the second is *checking if* a has the value 3. Type them into the console and compare.

Style convention 1. Comparison operators have a single space either side of them. Type a␣<=␣b and not a<=b.

In MAS114 you will learn about the logical operations 'and', 'or' and 'not'. You can use these in Python as `and`, `or` and `not`.

```
>>> x = 25
>>> print((x > 20) and (x < 30))
```

Now edit the statement `x = 25` to `x = 35`, re-run the `print` statement, and compare the answers.

Exercise 3.4. Write a short program that inputs an integer and prints either "It is True that your number is non-zero" or "It is False that your number is non-zero", whichever is correct. (Note that the capitalization will be forced on you.)

Boolean expressions are key in `while` loops, as you will now see. Next week you will see their role in conditional statements.

5 The `while` loop

It is often the case that we perform processes repeatedly. For instance, when walking somewhere, you repeatedly put one foot in front of another until you have reached your destination.

Concept 1. A **loop** in a computer program is a sequence of commands that are repeatedly executed a certain number of times or until some condition is met.

We used `while` loops last week without looking at them in depth. In English we have a while-type construction. In the morning you might follow the instructions "While your bowl has cereal in it, keep lifting out cereal with your spoon and putting it in your mouth." Python's `while` loop works in a similar way.

The general form of a `while` loop is as follows.

```
while boolean_expression:
    do_this
    then_this
    then_this
after_all_that_carry_on_here
```

If the given Boolean expression is `True`, the computer will perform the indented steps in order. Once it has completed the indented steps it will go back and check the Boolean expression again, repeating the steps in order, until it checks and finds that the Boolean expression is false.

Here's an example. Write down *exactly* what you think the following program will display. Now type it in and run it.

```
x = 3
while x < 8:
    print(x)
    x = x + 1
print("We've finished.")
```

Exercise 3.5. What will the following program do and why? What happens if you replace `1 < 2` with `1 > 2`? Write down your answer and then type in and run the program.

```
while 1 < 2:
    print("Spam")
    print("Spam and eggs")
print("Lobster thermidor")
```

Homework

1. Finish working through the sheet.

2. (*Quick review.*) Do the following without typing anything into Python. You can then check your answers using Python.

   (i) What is the output of `type(3.1415)`?
   (ii) How do you write $x \geq y$ in Python?
   (iii) What is the output of `print((3 < 2) or (10 == 5))`?
   (iv) Write down the exact output of the following program.

   ```
   total = 0
   i = 10
   while i > 5:
       total = total + i**2
       print(i, total)
       i = i - 2
   print("We finished with i =", i)
   ```

3. (*Assessed homework.*) This week's homework involves writing a program that asks for two positive integers and finds their greatest common divisor using a specified algorithm. To access the homework, you must navigate to the following webpage:

   `https://aim.shef.ac.uk/moodle/mod/quiz/view.php?id=424`

   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 4 (18th October).

4. (*Optional.*) Read the Wikipedia article on the **Gauss-Legendre formula** for calculating an approximation to π. Write a program to implement the algorithm.