EMAT10007 – Introduction to Computer Programming

Part 2: Variables & Types

[N.B. Questions marked with an asterisk (*) are optional and designed to be more challenging.]

Exercise 1 - Variables

1. Variables can be assigned values and (may) change during the execution of your program. In many languages, to assign a value to a variable, we just use a single "equals" sign: `=`. Try:

   ```
   x = 10
   y = 5
   ```

   What is returned now when you `print x` and `y`?
2. What happens if you now assign `x = 4`?
3. Variable names should start with a letter, and may contain letters or numbers. Be aware that some `keywords` are reserved by the Python language and cannot be used as variable names. Try the following:

   ```
   True = 1
   ```

   What happens here, and why? Now try:

   ```
   true = 1
   ```

   Note: Python is case-sensitive, so be careful when naming and using your variables!
   For a full list of keywords reserved by Python, enter the following:

   ```
   import keyword
   print(keyword.kwlist)
   ```

4. (*)
   We can also assign multiple variables on the same line.

   ```
   x, y = 5, 10
   ```

   How would you extend this to also assign 15 to the variable `z`?

Exercise 2 - Numbers and Operators

As discussed in the lectures Python can be used as a calculator. You can input operations, assign values to variables, and store the results of operations for use in additional calculations.

1. Create two variables called "A" and "B", and assign a value of your choice to each of these variables.
2. To calculate the addition of `A` and `B`, enter `A+B`.
3. Now enter `A*(A+B)`.
4. What happens if you enter `A*A+B` instead? Python follows the same ordering of mathematical operations as any other calculator.

5. Set `A = 10` and `B = 3`. Now enter `A/B` to calculate the division of A and B. What happens if you enter `A//B` instead?
6. Now try `A%B`.
   Note: `%` is the modulus operator, and calculates the remainder of a division.
7. (*) Can you calculate the circumference of a circle with a 5cm radius? How about a 12.5cm radius?
   Hint: Recall in the lectures we used the `math` module to gain extra functionality. It also gives us constant values such as `math.e`.

Exercise 3 - Booleans

1. Create two variables called "A" and "B", and assign some numeric values to each of them.
2. Print the results of:
   - `A < B`
   - `A > B`
   - `A == B` *(Note: Two equals signs)*

   You can also use `<=` for $\leq$ and `>=` for $\geq$ comparisons.
3. What happens if you do use `A=B` with only a single equals sign?
4. What happens if you write `not` in front of one of the lines above?
5. You can also set `A` and `B` to be Boolean truth values themselves, such as: `A = True` and `B = False`.

   Now try some logical (Boolean) operations such as:
   - `A and B`
   - `A or B`
   - `A and not B`
7. (*) You can also use the `bool()` function to evaluate any value as either `True` or `False`. What numbers evaluate as `True`? Do any evaluate as `False`?

Part 2: Lists & Strings

Exercise 4 - Strings

1. Create two variables called "A" and "B".
2. Assign `A` to "Hello" and `B` to "world". The quotation marks indicate that these are strings. You can check this with `type(A)` and `type(B)`.
3. Join these variables together by *adding* `A` and `B`: `A + B`
4. What happens when you try `A * B`?
   Note: Not all operators are defined for all variable types.
5. Notice how the string is missing a space between "Hello" and "world". Combine `A` and `B` with a new space " " in the middle, and assign this new string to a new variable `C`.

6. Print the length of this new string by entering `len(C)`.
7. `len()` is a built-in function in Python which simply returns the *length* of something. What types of variables does `len()` work on?

   Hint: Look at the Python 3 Documentation: `https://docs.python.org/3/library/functions.html`.
8. (*) You can also check if a certain phrase or character is present in a string by using the keywords `in` or `not in`, which return a boolean.
   Using the string `C` from above, try:
   - `"w" in C`
   - `"hello" in C`
   - `"Hello" in C`
   - `"world" not in C`
   - `A in C`

Exercises – Week 2. More on Variables, Types and Conditionals

Part 1. Variables & Types

Exercise 1 - Overwriting Variables

1. In the lectures we saw that you can also overwrite variables as functions of themselves. Assign `x = 5` and reassign it:

   ```
   x = x + 7
   x = x - 3
   ```

   What is `x` now?
2. The above question involved a lot of typing and so Python has the assignment operators `+=` (increment assignment) and `-=` (decrement assignment). Assign `y = 2` and then:

   ```
   y = y - 2
   y += 2
   ```

   What is `y` now? What do `+=` and `-=` do?
3. Rewrite Q1 using these new operators.
4. How would you add the value of the variable `y` to the value of the variable `y` using this shorter notation?
5. What happens if you try to assign `z += 1`? Why?
6. (*) How would we reassign a variable to itself multiplied by 10? What about itself divided by 5?
   Hint: Check the documentation here: `https://python-reference.readthedocs.io/en/latest/docs/operators/index.html#assignment-operators`

7. (*) What is the answer to $8 \div 2(2+2)$? Calculate this using Python. Is it what you expected? To learn more about this behaviour research the order and precedence of operators in Python.

Exercise 2 - Naming Variables

1. Python is quite flexible when it comes to naming variables but there are some rules that you need to keep in mind. Say we are trying to assign the string `"Whiskers"` to a cat name variable. Try the following:
   - `cat name = "Whiskers"`
   - `cat-name = "Whiskers"`
   - `1catname = "Whiskers"`
   - `$catname = "Whiskers"`
   - `cat1 name = "Whiskers"`
   - `catname = "Whiskers"`

   Which ones raise a `SyntaxError`? The errors are because variable names can only contain letters, numbers and underscores (_). They cannot start with a number or contain spaces. Which rule(s) are the above breaking?
2. Python is also case sensitive. Let us try to change the cat's name to `"Felix"` by typing:

   ```
   CATNAME = "Felix"
   ```

   What happens now when we call `catname`? Has it been overwritten?
3. When writing code it is good practice to use a consistent naming style. In this course we will be using `CamelCase` (also referred to as `UpperCamelCase` or `Pascal case`). This is where each word in the variable starts with an uppercase letter.
   How would you apply this style to the cat name variable? If the cat is six years old how would you represent this with a variable?
   Note: `CamelCase` is named after the "humps" of its proceeding capital letter - much like the hump of common camels. Find out more about other naming styles here (Kebab case is not valid in Python - can you see why? Can you see an example of where Kebab case is commonly used?): `https://medium.com/better-programming/string-case-styles-camel-pascal-snake-and-kebab-case-981407998841`
4. Recall from Exercise 1 that variables can be overwritten and often are; however, sometimes we have variables that should never be changed. These are known as constant variables and are often used in equations for fixed values e.g. gravity of earth $g = 9.8$. To signify that a variable is a constant and so should not be changed, the convention is to capitalise it, e.g. `GRAVITY = 9.8`
   How would you add a constant variable for species assigned to `"cat"`?
5. Recall from Exercise 1 that some `keywords` are reserved by Python and so cannot be used as variables name. However, we have to be careful as not all `keywords` restricted. Does assigning `len = 10` produce an error?
   Now try to call the `len()` function on a string. You should see the error:

   ```
   TypeError: 'int' object is not callable
   ```

This is because we have now redefined `len` as the integer 10. Check you can undo this using the `del` keyword:

```
del len
```

Note: Sypder helps us recognise these `keywords` by colour coding them.

Exercise 3 - Number Types

1. What `type` are the following variables?
   - `a = 5`
   - `b = 1.618`
   - `c = 1 + 2j`
   - `d = 9j`
   - `e = 0`
   - `f = 2.0 + 0j`
2. Use the `isinstance` function to find out which of the following variables are of type `int`:
   - `z = 1 / 2`
   - `y = 5 * 20`
   - `x = 5.0 * 20.0`
   - `w = (2.0+1j) * (2.0-1j)`

   Hint: Remember you can use `help()` for more information on a function like `type`.
3. Sometimes we might want to force a variable to be a particular type. This is called explicit type conversion, or type casting. We use the built-in functions `int()`, `float()`, `complex()`, `bool()` and `str()`.
   Try converting the variables in Q2 to each of the different number types and observe what happens. When do we lose information about the number?
4. What happens when we combine different types? What types are the following:
   - `e = 1.0 + 5`
   - `f = 3j + 4.6`
   - `g = 4 * 2.5`
   - `h = 4 * 2.7`

   This is the result of implicit type conversion, where Python automatically converts one type into another. This requires no input from us and is designed to avoid the loss of information.
   Note: This also occurs when you calculate $1 \div 2$. Although you are dividing two `ints`, a `float` is returned.
5. (*) Explore the conversion functions `oct()`, `hex()` and `bin()`. Research for what purposes these might be used.
   Hint: Remember the Python Documentation for built-in functions: `https://docs.python.org/3/library/functions.html`

Exercise 4 - Boolean Operators

1. Boolean values are the two constants `False` and `True`. Recall that Python is case-sensitive and so these values must begin with a capital letter. What happens if not?
   Once again Spyder helps us recognise that `True` and `False` are special **`keywords`** by highlighting them using a different colour.
2. Assign three variables $A = 5$, $B = 10$ and $C = 0$. Find the results of the following:
   - `A < B < C`
   - `(A <= B) and (A > C)`
   - `(2*A) == B`
   - `A and B and C`
   - `(A != B) or C`
   - `(A > 0) and (B > 0) or not (C == 0)`

   Hint: It can be helpful to work out the truth tables for more complicated expressions. Have a look here: `https://www.digitalocean.com/community/tutorials/understanding-boolean-logic-in-python-3`
3. (*) You might have noticed above that some operators are binary (e.g. `and` and `or`) and some are unary (e.g. `not`). Another binary operator is $\wedge$ the `xor` operator. Can you figure out what this operator does?

Part 2. Lists & Strings

Exercise 5 - Further Strings

1. Python also supports type casting to strings using the `str()` function. Turn the following into strings.
   - 4
   - 3.14159
   - 2+9j
2. What happens when you assign a variable to just `str()`, i.e `w = str()`?
3. Unlike for numbers, Python **cannot** use implicit type conversion with strings. For example, try:

   ```
   a = "The meaning of life is"
   a = a + " " + 42
   ```

   You should see you have a `TypeError`. This is because the `+` operator means addition for numbers, but concatenation for strings. We need to explicitly convert our int into a string. How would you do this?

4. Strings are a sequence of characters and we can call elements of them using square brackets (`[]`). How would I access the first and last elements of the string `a`? Is there another way to index the last element?
   Hint: Remember the `len()` function
5. There are lots of built-in string operators (also called methods) available. Use `help(str)` to read about them. Can you find what `replace()` does? Can you use it to change the `42` in the string `a` to something else?
   Note: You can also see everything from the `help()` function in the documentation here: `https://docs.python.org/3/library/stdtypes.html#str`
6. (*) What does the string method `join()` do? Try it out - does it work how you expected?

Exercise 6 - Making a .py file

1. For larger programs you will want to save your work as a python file.
   - Create a new python file by clicking on "File" and then "New File".
   - In your new file add the line `NumberInFile=5`.
   - Now click "File" and then "Save".
   - You will be prompted to choose a file name and location. Name your file and save it somewhere appropriate.
   - Next time you open Spyder, you can access the file by clicking on "File" then "Open", and then navigating to where your file is stored.

Part 3. Control flow

Exercise 7 - Conditionals

1. We can use Boolean operators to write conditional statements which control the flow of our program. In the interactive shell, type:

   ```
   AgeOfWhiskers = 5
   if AgeOfWhiskers <= 1.5:
       print("Whiskers is a kitten")
   ```

   What do you think will happen?
2. We can also use the `else` keyword which triggers only when all the above conditions are false. Use an `else` statement to print the string `"Whiskers is an adult cat"`?
3. Wasn't it annoying retyping everything? Move your code to a file to make changing it easier.
4. Add an `elif` statement so the string `"Whiskers is an old cat"` prints when Whiskers is older than 12.
5. Now we might want to reuse this code to test the ages of many different cats. Generalise your code to use the variable `CatAge`.

6. (*) Suppose we have three circles in the xy-plane. Circle C_1 is centred at $(0, 0)$ with radius of length 5. Circle C_2 is centred at $(2, 1)$ and has radius of length 2. Circle C_3 is centred at $(-5, 0)$ and has a radius of length 3. (See Figure 1)
   Using conditional statements write a program which takes in the variables `x` and `y` and tests which circles the point (x, y) is in. How can you make this as concise as possible? Are there any conditions you do not have to test?

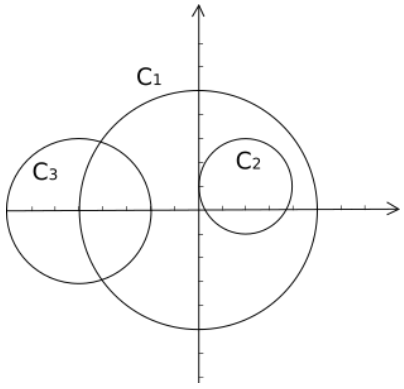

Figure 1: Overlapping circles C_1, C_2 and C_3.

Checklist

- Check that you understand the basics: variables, different types of variables (integers, floats, complex, Booleans, strings), the different built-in operators, and how these work with both numbers and strings.
- Practice with using Spyder as a useful calculator with access to some powerful abilities provided in "modules".
- Check you are familiar with variable naming styles and conventions.
- Know how to save and open a python file script in Spyder.
- Learn about conditional statements, and how these are used to control the flow of the program.

Additional resource

- There are many online tutorials for Python 3, so if you're wondering what more you can do, please do search online or ask your TAs for pointers.