Week 8: Formatting and tabulation

Intended learning outcomes

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

- use the string format method to tailor outputs;
- copy lists and strings;
- define and use anonymous functions in Python.

1 The string format method

We saw a little of the `str.format()` method at the end of the last lab. This is a useful part of Python. Try the following:

```
>>> function = "sin"
>>> derivative = "cos"
>>> template = "The derivative of {0} is {1}."
>>> print(template.format(function, derivative))
```

Here, the `format()` method has taken its first argument, in this case the string `function`, and inserted it where the `{0}` was. Remember that the first item has index 0. The `format()` method has taken the second argument, which has index 1 and in this case is the string `derivative`, and inserted it where the `{1}` was in the string `template`. Since they are numbered, the items don't have to appear in the template in the same order that they appear in the list.

```
>>> template = "An integral of {1} is {0}."
>>> print(template.format(function, derivative))
```

When the parameters are substituted into the template, you can control how they appear, by using a **format specification**. A format specification is given by putting a colon after the index. The simplest part of the specification is the width: this is just an integer. Compare the outputs of the following:

```
>>> print("={0}=".format("parrot"))
>>> print("={0:20}=".format("parrot"))
>>> print("={0:6}=".format("parrot"))
>>> print("={0:5}=".format("parrot"))
```

In the first of these, the string `"parrot"` is placed between the two equals signs. In the rest, the string `"parrot"` is placed between the equals signs but padded out with extra spaces, if necessary, to make the total width equal to the given number. Note that if the specified width is less than the length of `"parrot"`, then no spaces are added but no attempt is made to shorten the string.

When dealing with floats you will often want to specify only a fixed number of decimal places. The `format()` method allows you control of this: for an integer n, using `.nf` in the format specification will cause the float to be rounded to n decimal places.

```
>>> import math
>>> print("Approximately, pi is {0}.".format(math.pi))
>>> print("Approximately, pi is {0:.5f}.".format(math.pi))
>>> print("Approximately, pi is {0:.1f}.".format(math.pi))
```

Exercise 8.1. Use the `format()` method with `math.pi` and `math.e` to produce the following output:

```
Roughly, pi is 3.14 and e is 2.72.
```

You can use the width specification with floats as well. You need to put them before the decimal point, e.g. `{0:10.5f}` would give you 5 decimal places with the entire number padded out with spaces to be 10 characters wide.

2 Looking at the limit of a function

In the section on limits in MAS110, you have to tabulate the values of various mathematical functions to see what the limits appear to be. We can write a Python function to do that.

Suppose you have a function $f(x)$ and you want to look at the limit of the function at $x = a$, where the function might not be defined. You can tabulate the values of $f(a + 10^{-i})$ for $i = 1, \ldots, n$. For large n you can often get a sense of whether $\lim_{x \to a} f(x)$ exists.

Consider the following program. Do *not* type it in yet – look at it and see if you can understand what it will do.

```
import math


def limit_table(f, a, n):
    """
    Print out a table of n values of the function f(x)
    close to x = a.
    """
    horizontal_line = "-" * 31

    print(horizontal_line)
    print("   {0:14}   {1:14}".format("x", "f(x)"))
    print(horizontal_line)

    for i in range(1, n+1):
        print("{0:14.10f}   {1:14.10f}".format
              (a + 10**(-i), f(a + 10**(-i))))

    print(horizontal_line)


def f_1(x):
    """Calculate a simple rational function of x."""
    return (x**4 - x**3 + x**2 - 1)/(x - 1)


def f_2(x):
    """Calculate the function sin(7x)/x."""
    return math.sin(7*x)/x


print("\nf(x) = (x**4 - x**3 + x**2 - 1)/(x - 1)")
limit_table(f_1, 1, 10)

print("\n\nf(x) = sin(7x)/x")
limit_table(f_2, 0, 10)
```

There are several things to note here.

- The main part of the program is the function `limit_table()`, which takes three parameters: a function, `f`; a float, `a`; and an integer, `n`. The weird thing here is that the function takes another function as a parameter! However, Python is okay with that: the call `limit_table(f_1, 1, 10)` causes Python to make the assignments `f = f_1`, `a = 1` and `n = 10` before executing the function.
- The Python functions above take two different forms: `limit_table()` actually prints out something and looks like a mini-program, while `f_1()`

and `f_2()` are really just like mathematical functions. However, Python treats these all the same.
- In this program we have not used tab characters to create the table, but rather have used the `str.format()` method to produce columns of 14 characters in width, with three spaces between the columns. Look at lines 12 and 16.
- On lines 5 to 8, the docstring takes up more than one line, so it is set out differently to the docstrings on lines 23 and 28: the triple quotation marks go on lines by themselves.
- Lines 16 and 17 are really one line, but split into two to fit the width of the screen. Python knows that line 16 doesn't really end because there is a bracket which is still open at the end of the line.

Now type in the program above and run it.

Exercise 8.2.

- Modify the program so that `limit_table()` also prints out the values just below $x = a$, i.e. also tabulates $f(a - 10^{-i})$ for $i = 1, \ldots, n$.
- Get the program to also tabulate values for $\dfrac{x^2 - 1}{x - 1}$ near $x = 1$.

3 Mutability of lists

This is a more technical section on the so-called *mutability* of lists. The list type behaves differently in some respects to the float, integer, boolean and string types. This is something you should be aware of, but not worry too much about now. It is a useful feature of Python but can occasionally cause some headaches and result in bugs that are hard to pin down.

Strings and lists are different in that you can change the elements of a list but you cannot change the elements of a string. Compare the following two sets of commands:

```
>>> menu = ["spam", "egg", "beans"]
>>> print(menu[0])
>>> menu[0] = "sausage"
>>> print(menu)
>>>
>>> prop = "parrot"
>>> print(prop[0])
>>> prop[0] = "c"
```

Python will not let you *change* a string. You can, of course, replace it with a *new* string.

```
>>> prop = "carrot"
>>> print(prop)
```

We say that lists are *mutable* and strings are *immutable*. This difference has some consequences that can cause confusion. For example, try the following:

```
>>> food = menu
>>> print(food, menu)
>>> food[2] = "fried aubergine"
>>> print(food, menu)
```

Changing `food` has changed `menu`! This is because they referred to the same list, changing it in one place changed it in another. If we had made `food` a *new* list then this wouldn't be a problem – it is the changing that is the problem.

```
>>> food = ["toast", "butter"]
>>> print(food, menu)
```

The important bit. This leads to the question of how you can copy a list. Suppose you want to make a copy of `menu` and then do some things to the copy without altering the original. There are various ways to do this, but a good way is to use the `list()` command. This will make an independent copy.

```
>>> food = list(menu)
>>> print(food, menu)
>>> food[2] = "jam"
>>> print(food, menu)
```

Note this will not work with lists of lists. To do this you need to use the `deepcopy()` command in the `copy` module, but we won't go into this here. You can look online for that if necessary.

The idea of mutability of lists can be confusing, but you can look at the chapter on lists in the freely available Think Python (`https://greenteapress.com/wp/think-python`) for more information.

4 (Advanced) Anonymous functions

In mathematics we can talk about functions without giving them names: so-called *anonymous* functions. For example, we can talk about the function

$$x \mapsto \frac{x^2 - 1}{x - 1},$$

which sends a number x to the expression on the right. Note that the arrow is a barred arrow $\mapsto$ and not an ordinary function arrow $\to$. You can do a similar thing in Python with **anonymous** or **lambda** functions. The syntax would be as follows:

```
lambda x : (x**2 - 1)/(x - 1)
```

This is only used when you want a simple function for a short time and it does not need a name. This means that in the above program you could get away with not defining `f_1` and `f_2`, i.e. you could delete lines 22–29, and replace lines 33 and 36 with the following lines:

```
limit_table(lambda x : (x**4 - x**3 + x**2 -1)/(x-1), 1, 10)
limit_table(lambda x : math.sin(7*x)/x, 0, 10)
```

Homework

1. Finish off this sheet.

2. (*Assessed homework.*) This week's homework involves writing a program that tabulates the values of a specified function $f(x)$ for increasingly large values of x. To access the homework, you must navigate to the following webpage:

   `https://aim.shef.ac.uk/moodle/mod/quiz/view.php?id=429`

   Your code will be run with some test functions. 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 next Friday (27th November).