**String formatting in Python [Insert this as the first paragraph on page 179]**

Prior to Python 3.6, there were two ways to embed Python expressions inside string literals for formatting: %-formatting and str.format (). String objects have a built-in operation that uses the% operator to format strings. In practice, this looks like this:

**Using % formatting**

Here is the simplest form of % formatting

first\_name="Timothy"

print("my name is %s" % first\_name)

We can eve use more than  one variable, but they have to be closed in a brackets (tuple) as in the example below:

first\_name="Timothy"

surname="Mpofu"

Age=19

Level="NCV2"

#using %s to format output

print("My name is %s, and my surname is %s" % (first\_name, surname))

print("I am %s years old, and I am in %s" % (Age, Level))

**eLinks**

For more on formatting strings, refer to the Python Docs link:

<https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting>

**Using the str.format()**

We briefly introduced this in module 5 but did not go in depth. *str.format()* is an improvement on %-formatting. With *str.format(),* the replacement fields are marked by curly braces { }. Take a look at the code below which is similar to the one we did using the % formatting.

first\_name="Timothy"

surname="Mpofu"

Age=19

Level="NCV2"

#using str.format to format output

print("My name is {}, and my surname is {}." .format(first\_name, surname))

print("I am {} years old, and I am in {}." .format(Age,Level))

You can use str.format to refer to variables in any order by referencing their index:

first\_name="Timothy"

surname="Mpofu"

Age=19

Level="NCV2"

#using str.format

print("My name is {2}, and my surname is {1}." .format(Age, surname, first\_name,Level))

first\_name is at index 2 and surname is at index 1.

Although str.format() code is much more readable than %-formatting code, it can still be quite verbose when dealing with multiple parameters and longer strings. A newer way which addresses the weaknesses of % and *str,format*() is the f string.

**Using f string**

f-strings are string literals that begin with a f and contain curly braces that contain expressions that will be replaced with their values. The f string syntax is similar to  the *str.format()* syntax but less verbose. Take a look at how easy this is to read:

first\_name="Timothy"

surname="Mpofu"

Age=19

Level="NCV2"

#using f-string to format output

print(f"My name is {first\_name}, and my surname is {surname}." )

Moving forward, the f-string is the most recent technique for formatting output and we recommend you to make use of it.

**datetime module**

continue from the first paragraph on page 179.