String Interpolation and Concatenation

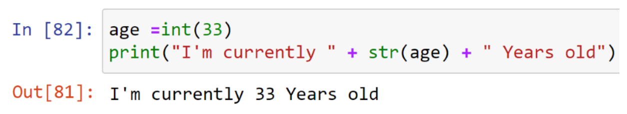
Monday, December 13, 2021 3:16 PM

BASIC CONCATENATION

String objects have a built in operation which is the concatenation operator +. Using this operator is known as String interpolation or string formatting.

When you build a string in Python, you concatenate a lot of values, especially when those values are not just strings, for example, when concatenating a integer into a string, you have to convert the integer into a string.

Concatenation can be done by simply using an concatenation + operator after the string and before any string that appears after. See example:



```
In [82]: age =int(33)
         print("I'm currently " + str(age) + " Years old")

Out[81]: I'm currently 33 Years old
```

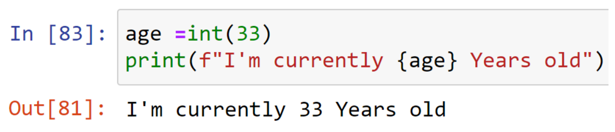
Note, that we use both single and double quotes because we are using a word that contains an apostrophe which we explored earlier. Because the variable age was casted as an integer, we have to convert it into a string by casting the age variable as a string.

FORMAT STRING

Python has a special type of string called a format string, also known as an f-string. This doesn't give you any new capabilities over regular concatenation, but is something called "syntax sugar", making the code easier to read.

To write an f string we place an f f before the opening quotation of the string literal and within this string we can surround expressions in curly brackets.

Using the same code example as before, we can rewrite it but using an f-string, see example:



```
In [83]: age =int(33)
         print(f"I'm currently {age} Years old")

Out[81]: I'm currently 33 Years old
```

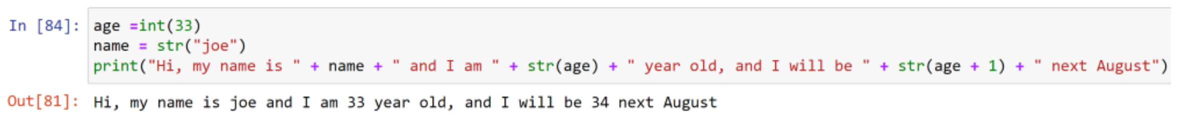
The curly brackets are used to tell the Python interpreter that whatever is inside them is a Python code expression. The interpreter then converts it into a string and concatenates it inside of the string without the need to do it ourselves like in normal concatenation.

So we can see that when a f-string literal is evaluated, each curly bracket expression is converted to a string then concatenated into the string literal and evaluated like a normal expression.

Both examples generate the same result, the f-string is just shorter and easier to work with.

CONCATENATION WITH DIFFERENT DATA TYPES

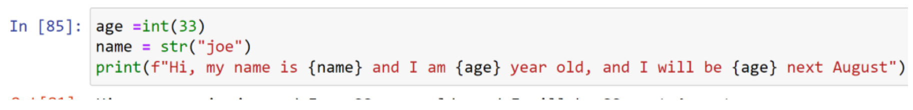
Different data types can be used in string concatenation, the first example shows the string using regular concatenation and shows how much more work it is:



```
In [84]: age =int(33)
         name = str("joe")
         print("Hi, my name is " + name + " and I am " + str(age) + " year old, and I will be " + str(age + 1) + " next August")

Out[81]: Hi, my name is joe and I am 33 year old, and I will be 34 next August
```

Now see the next example with the same code but written in a f-string:



```
In [85]: age =int(33)
         name = str("joe")
         print(f"Hi, my name is {name} and I am {age} year old, and I will be {age} next August")
```

```
Out[81]: Hi, my name is joe and I am 33 year old, and I will be 33 next August
```

Note that we can use two expressions in an f-string literal.
Like before, the result is the same.