Day 26: String Formatting

Where to see all the latest updates in python:
https://docs.python.org/3/whatsnew/index.html

F-Strings

F-string allows you to format selected parts of a string.

To specify a string as an f-string, simply put an `f` in front of the string literal, like this:

```
txt = f"The price is 49 dollars"
print(txt)
```

Placeholders and Modifiers

To format values in an f-string, add placeholders `{}`, a placeholder can contain variables, operations, functions, and modifiers to format the value.

```
price = 59
txt = f"The price is {price} dollars"
print(txt)
```

A placeholder can also include a *modifier* to format the value.

A modifier is included by adding a colon `:` followed by a legal formatting type, like `.2f` which means fixed point number with 2 decimals:

```
price = 59
txt = f"The price is {price:.2f} dollars"
print(txt)
```

You can also format a value directly without keeping it in a variable:

```
txt = f"The price is {95:.2f} dollars"
print(txt)
```

Perform Operations in F-Strings

You can perform Python operations inside the placeholders.

You can do math operations:

```
txt = f"The price is {20 * 59} dollars"
print(txt)
```

You can perform math operations on variables:

```
# Add taxes before displaying the price

price = 59
tax = 0.25
txt = f"The price is {price + (price * tax)} dollars"
print(txt)
```

You can perform `if...else` statements inside the placeholders:

```
price = 49
txt = f"It is very {'Expensive' if price>50 else 'Cheap'}"

print(txt)
```

Execute Functions in F-Strings

You can execute functions inside the placeholder:

```
# Use the string method upper()to convert a value into upper case letters
```

```
fruit = "apples"
txt = f"I love {fruit.upper()}"
print(txt)
```

The function does not have to be a built-in Python method, you can create your own functions and use them:

```
# Create a function that converts feet into meters:


def myconverter(x):
  return x * 0.3048

txt = f"The plane is flying at a {myconverter(30000)} meter altitude"
print(txt)
```

More Modifiers

At the beginning of this chapter we explained how to use the `.2f` modifier to format a number into a fixed point number with 2 decimals.

There are several other modifiers that can be used to format values:

```
# Use a comma as a thousand separator:

price = 59000
txt = f"The price is {price:,} dollars"
print(txt)
```

Formatting Types	
`:<`	Left aligns the result (within the available space)
`:>`	Right aligns the result (within the available space)
`:^`	Center aligns the result (within the available space)

`:=`	Places the sign to the left most position
`:+`	Use a plus sign to indicate if the result is positive or negative
`:-`	Use a minus sign for negative values only
`: `	Use a space to insert an extra space before positive numbers (and a minus sign before negative numbers)
`:,`	Use a comma as a thousand separator
`:_`	Use a underscore as a thousand separator
`:b`	Binary format
`:c`	Converts the value into the corresponding Unicode character
`:d`	Decimal format
`:e`	Scientific format, with a lower case e
`:E`	Scientific format, with an upper case E
`:f`	Fix point number format
`:F`	Fix point number format, in uppercase format (show `inf` and `nan` as `INF` and `NAN`)
`:g`	General format
`:G`	General format (using a upper case E for scientific notations)
`:o`	Octal format
`:x`	Hex format, lower case
`:X`	Hex format, upper case
`:n`	Number format
`:%`	Percentage format