PYTHON SNIPPETS

Formatting Integers in Different Bases in Python

Phil Best

3 min read · Sep 16

In the last few Python snippet posts, we've been on a bit of a formatting binge. First we looked at a number of special formatting options for numbers using the Formatting Specification Mini Language, and then we looked at nested string interpolation. This week we're tackling printing numbers in different bases, and we're going to build a little colour converter at the end.

As with all of our special formatting options, we start with a string that includes some curly braces for string interpolation. We add a colon inside the curly braces, and then our options come after. In our case, we just have to add a single letter, and which letter depends on how which base we want to use to represent our number:

```
base_10 = 231

print(f"This is the number in binary: {base_10 :b}")
# This is the number in binary: 11100111
```

Here we added `:b` inside the curly braces, which formatted our

`base_10` number as a binary representation of that number.

Let's look at a few more examples:

```
base_10 = 231

print(f"This is the number in octal: {base_10 :o}")
# This is the number in octal: 347

print(f"This is the number in hexadecimal: {base_10 :x}")
# This is the number in hexadecimal: e7

print(f"This is the number in uppercase hexadecimal: {base_10 :X}")
# This is the number in uppercase hexadecimal: E7
```

We can also convert a number from another base to decimal this way:

```
base_16 = int("E7", base=16)

print(f"This is the number in decimal: {base_16 :d}")
# This is the number in decimal: 231
```

You can find tables listing the various presentation types here: https://docs.python.org/3/library/string.html#format-specification-mini-language

An RGB to Hex Colour Converter

Now that we've taken a look at some of the options, let's put them to use! We're going to create a quick RGB to hex colour converter. RGB values look like this: `(11, 255, 68)`. The first value represents the amount of red in the colour, the second value represents the amount of green, and the third value represents the amount of blue.

Hexadecimal colours work in a similar way, but instead of using three comma separated values, hex colours use pairs of hexadecimal values that together form a 6 character string. Usually this string of characters has a `#` in front of it. An example might be `#e3f102`, which can also be written as `#E3F102` with capital letters. They're absolutely identical. Once again, the colours are broken up into red, green, and blue, with each pair of values representing the amount of a given colour.

In order to convert from RGB to hexadecimal we have to convert each channel separately, ensuring that each hexadecimal value has two numerals. Then we have to glue all of the values together and prepend the `#` symbol.

One case we have to look out for is when an RGB channel value is less than 16. Because hexadecimal is base-16, numbers less than 16 can be represented with a single numeral. For example, the hexadecimal value for `12` is just `c`. In our code, we need to represent this as `0c` instead, so we're going to use Python's formatting language to add a zero before any hexadecimal value with only one numeral.

The syntax for this is `:02x`, which means represent the number as hexadecimal (`x`), display the number with two figures (`2`), and pad the number with zeroes if we have fewer than two figures (`0`).

```
red = 12
green = 205
blue = 81

hex_red = f"{red:02x}"
# 0c

hex_green = f"{green:02x}"
# cd

hex_blue = f"{blue:02x}"
# 51
```

That's a lot of code repetition, so let's turn our RGB values into a tuple and iterate over it to convert each value. We can use a list comprehension for this:

```
rgb = (12, 205, 81)
hex_colours = [f"{channel:02x}" for channel in rgb]

# ['0c', 'cd', '51']
```

Now that we have this list of hex colour pairs, we can join them using the `join` method (see documentation), to turn them into a single connected string. We can then attach this string to `"#"`.

```
rgb = (12, 205, 81)
hex_color = "#" + "".join([f"{channel:02x}" for channel in rgb])

print(f"Converted {rgb} to {hex_color}")
# Converted (12, 205, 81) to #0ccd51
```

Wrapping Up

That's it for this week! I hope you enjoyed that mini project and I hope you learnt something new about formatting numbers.

If you're interested in improving your Python skills, we'd love to have you on our Complete Python Course. We think it's an awesome course, but we have a 30 day money back guarantee just in case the course isn't for you.

You can also sign up to our mailing list below to stay up to date with all our content, and to get regular discount coupons for all of our

courses.

Phil Best
I'm a freelance developer, mostly working in web development. I also create course content for Teclado!

Read More

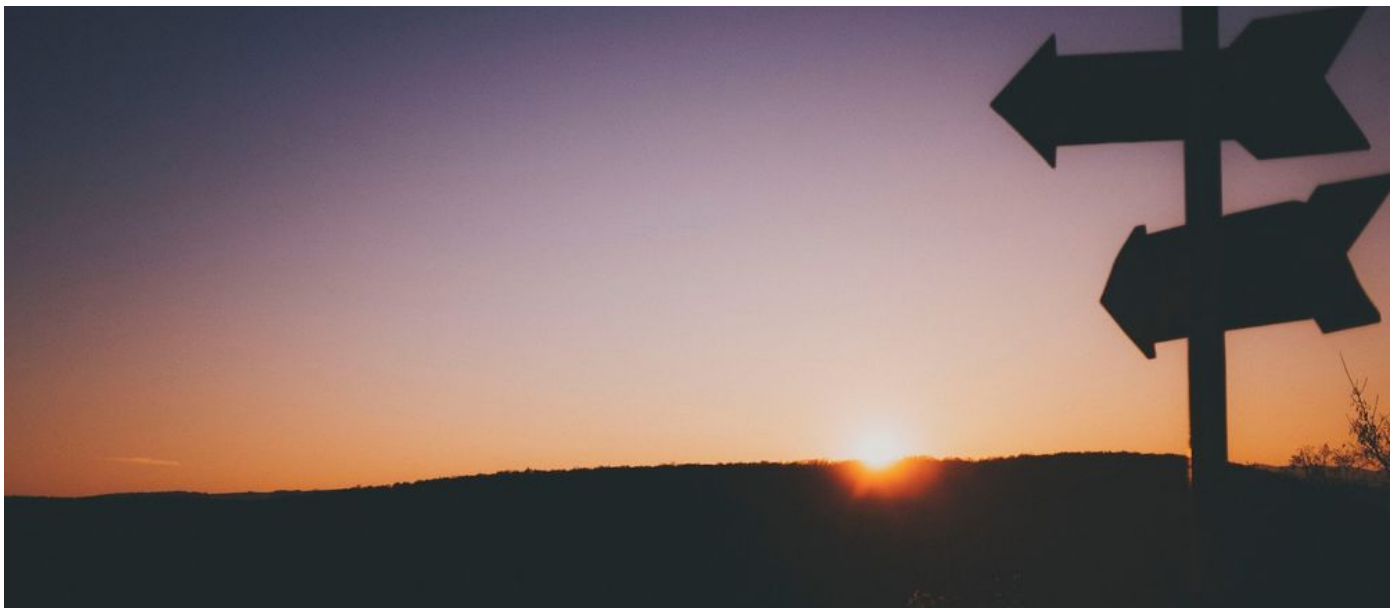
Python Snippets

Assignment expressions in Python →

Python's namedtuples →

Python's divmod Function →

→ See all 26 posts



FLASK

Flashing messages with Flask

Learn how to temporarily show messages in your websites when using Flask, by using the built-in message flashing mechanism. It's flexible yet straightforward!

Jose Salvatierra
5 min read · Sep 19