6. Nice formatting Numbers

1. Basic Number Formatting

The `format()` function takes a string as its first argument, and you can use replacement fields within the string to specify where the formatted values should be inserted. Here's an example:

```
num = 12345.6789
print("The number is: {}".format(num))
```

Output:

```
The number is: 12345.6789
```

In this example, the `{}` is a replacement field, and `num` is inserted into that field.

2. Specifying the Field Width

You can control the width of the field by adding a number between the braces. If the value is shorter than the specified width, it will be padded with spaces on the left:

```
num = 12345.6789
print("The number is: {10}".format(num))
```

Output:

```
The number is:  12345.6789
```

Here, `{10}` specifies that the field should be at least 10 characters wide.

3. Formatting Floating-Point Numbers

By default, `format()` prints floating-point numbers with a fixed precision of 6 decimal places. You can change this by adding a precision specifier after the field width:

```
num = 12345.6789
```

```
print("The number is: {10.2f}".format(num))
```

Output:

```
The number is:   12345.68
```

In this example, `{10.2f}` specifies a field width of 10 characters and a precision of 2 decimal places for floating-point numbers.

4. Formatting Integers

You can also format integers using the `format()` function. For example, you can add commas as thousands separators:

```
num = 12345678
print("The number is: {:,}".format(num))
```

Output:

```
The number is: 12,345,678
```

Here, `{:,}` specifies that commas should be used as thousands separators.

5. Formatting Currency

Python also provides a way to format numbers as currency values:

```
num = 12345.6789
print("The number is: ${:,.2f}".format(num))
```

Output:

```
The number is: $12,345.68
```

In this example, `${:,.2f}` specifies that the number should be formatted as a currency value with a dollar sign, two decimal places, and commas as thousands separators.

See Also