Numeric Literals

Integer literals can be written as:

- A *decimal* number, with no prefix
- A *binary* number, with a `0b` prefix
- An *octal* number, with a `0o` prefix
- A *hexadecimal* number, with a `0x` prefix

All of these integer literals have a decimal value of `17`:

```
let decimalInteger = 17
let binaryInteger = 0b10001       // 17 in binary notation
let octalInteger = 0o21           // 17 in octal notation
let hexadecimalInteger = 0x11     // 17 in hexadecimal notation
```

Floating-point literals can be decimal (with no prefix), or hexadecimal (with a `0x` prefix). They must always have a number (or hexadecimal number) on both sides of the decimal point. Decimal floats can also have an optional *exponent*, indicated by an uppercase or lowercase `e`; hexadecimal floats must have an exponent, indicated by an uppercase or lowercase `p`.

For decimal numbers with an exponent of `exp`, the base number is multiplied by 10^{exp}:

- `1.25e2` means 1.25 x 10^{2}, or `125.0`.
- `1.25e-2` means 1.25 x 10^{-2}, or `0.0125`.

For hexadecimal numbers with an exponent of `exp`, the base number is multiplied by 2^{exp}:

- `0xFp2` means 15 x 2^{2}, or `60.0`.
- `0xFp-2` means 15 x 2^{-2}, or `3.75`.

All of these floating-point literals have a decimal value of `12.1875`:

```
let decimalDouble = 12.1875
let exponentDouble = 1.21875e1
let hexadecimalDouble = 0xC.3p0
```

Numeric literals can contain extra formatting to make them easier to read. Both integers and floats can be padded with extra zeros and can contain underscores to help with readability. Neither type of formatting affects the underlying value of the literal:

```
let paddedDouble = 000123.456
let oneMillion = 1_000_000
let justOverOneMillion = 1_000_000.000_000_1
```