Data Types

- *Every **value***, or piece of data, has a specific ***type***

- In Python there are five *primitive data types*
  1. **`int`** - numerical integers
  2. **`float`** - numerical data with decimal points ("floating point")
  3. **`complex`** - numerical data with imaginary numbers *for science*
  4. **`str`** - short for "string" and textual data
  5. **`boolean`** - logical true or false data

- **Primitive data types** are the *simplest* types built into a language
  - Later you will compose primitive data types together to form composite data types

- A **value's type informs its capabilities**
  - If you have two `number` values, then you can add them, for example.

Numerical Types

`int`

- **Integers** are generally useful for ***counting***
- **`int` Literal** examples: **`0, 1, 2, 100`**

`float`

- **Floating Points** have decimals useful for ***calculations***
- **`float` Literal** examples: **`0.0, 1.0, 2.99, 100.001`**
- ***Warning:*** Floating points values cannot represent *every* decimal value. Many values can only be *approximated*. For example, open the REPL and evaluate **`0.2 + 0.1`**

Numerical Operators

	Name	Operator Symbol	Example
1	Exponentiation	**	`2 ** 8` (is the same as 2^8)
2	Multiplication	*	`10 * 3`
	Division	/	`7 / 5` (result is 1.4)
	Integer Division	//	`7 // 5` (result is 1)
	Remainder	%	`7 % 5` (result is 2)
3	Addition	+	`1 + 1`
	Subtraction	-	`111 - 1`

- Complex expressions can be formed of multiple operators and parenthesis:

  `4 // ((1 + 1) ** 2) is 1`

- Rules of precedence determine the order each operator is evaluated

  `4 // 1 + 1 ** 2 is 5`

- The groupings show operators tiered from high-to-low precedence.
  - A sequence of operators of the same tier will be evaluated left-to-right.
  - For example, **`8 // 4 * 2`** is **`4`**.

- Standard division operator results in a **`float`** value, even **`4 / 2`**.

Textual Type - `str`

- `str` is short for **"string of characters"**

- Literal examples: `"abc"`, `"123"`, `"~() @#z2"`
  - Characters surrounded by double quotes.

- Useful for **textual data**.

- **Docstrings** are a special kind of string used for *documenting* code
  - `"""Docstring literals are surrounded by sets of three quotes."""`
  - Unlike normal strings, docstrings can span multiple lines of code.

- Python has other useful variations on string values you'll learn in time.

Concatenation

- **Concatenation** is when you "add" two **`str`** values together

```
"Hello" + "World"  # evaluates to "HelloWorld"
```

- Concatenation means, simply, to "join" one string to another. When the program *concatenates* two strings, the result is a *single* String value.

- To concatenate a value that is *not* a str *to* a str, you must convert it:
  - **`"1 + 1 is " + str(1 + 1)`** **`# Evaluates to "1 + 1 is 2"`**
  - In the Python REPL, try: **`"1 + 1 is " + 2`**
    - TypeError: can only concatenate str (not "int") to str

Concatenation Gotcha

- Be careful **concatenating** two strings containing numbers together!
- What is:

```
"1" + "2"
```

- ***It is "12"!***

Logical Type - `bool`

- Literal examples: `True`, `False`
- A `bool`, short for Boolean, can only be **one of two values**, *either* `True` or `False`.
- An upcoming lesson will focus on `bool` operators:
  - `not`
  - `and`
  - `or`