Numbers

Reference: https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex.

```
100 #> 100
-100 #> -100
0.45 #> 0.45
```

Numeric Operations

Numeric functions include the usual arithmetic operators:

```
100 + 5 #> 105
100 - 5 #> 95
100 * 5 #> 500
100 / 5 #> 20
100 + 5 * 2 #> 110
(100 + 5) * 2 #> 210
```

Boolean equality operators also apply:

```
100 == 100 #> True
100 == 100.0 #> True
100 == 99 #> False
100 == (99 + 1) #> True
True == 1 #> True
False == 0 #> True
```

Also reference the built-in `round()` function: https://docs.python.org/3/library/functions.html#round.

```
round(4.5) #> 5.0
round(4.49) #> 4.0

round(4.49, 0) #> 4.0
round(4.49, 1) #> 4.5
round(4.49, 2) #> 4.49
```

Formatting as Currency

Use string formatting to control how numbers will display when printed:

```
# using the format function:
"the price is ${0:.2f}".format(6.5) #> 'the price is $6.50'
"the price is ${0:,.2f}".format(1234567890.12345678) #> 'the price is
$1,234,567,890.12'

# alternatively using a format string:
price = 6.5
f"the price is ${price:,.2f}" #> 'the price is $6.50'

price = 1234567890.12345678
f"the price is ${price:,.2f}" #> 'the price is $1,234,567,890.12'
```

Feel free to use (copy-paste) this function definition into your projects:

```
def to_usd(my_price):
    """
    Converts a numeric value to usd-formatted string, for printing and
display purposes.
    Source: https://github.com/prof-rossetti/intro-to-
python/blob/master/notes/python/datatypes/numbers.md#formatting-as-currency
    Param: my_price (int or float) like 4000.444444
    Example: to_usd(4000.444444)
    Returns: $4,000.44
    """
    return f"${my_price:,.2f}" #> $12,000.71
```

Advanced Operations

Also reference the numeric functionality of these built-in Python modules:

- `math`
- `random`
- `statistics`