Values

Literals:

- integer literals (e.g., `42`)
- *floating-point* (real-valued) literals (e.g., `3.14`)
- *string* literals (e.g., `'hello'`)
- *Boolean* (logical) literals (e.g., `True`)

Variables

Variables

Named values

Actually, it's slightly more complicated than that, but...

```
>>> from math import *
>>> pi
3.141592653589793
>>> 2j * pi
6.283185307179586j
```

```
r1 + r2
```


We'll talk more about variables in later lectures when we talk about how to _____ them. For now we will just focus on _____ them.

Operations

Arithmetic operators

- addition, substraction, multiplication, division (both kinds)
- exponentiation

Today: more operators!

Values

- literals ✅
- variables ✅

Operators

- arithmetic ✅
- function calls
- logical operators (**next week**)

Function calls

There's lots we *will* say about functions, but for now...

What are mathematical functions?

e.g., $\sin(x)$

Calling Python functions:

```
>>> from math import *
>>> sin(pi)
1.2246467991473532e-16
```


Again, this is _____ to 0 but not _____ so!

Math vs programming

Functions are like math functions, but also not!

Recall: $e^{jx} = \cos x + j \sin x$

Given:

$$f(t) = 8x + j \sin \frac{t}{2} + 1 + \cos \frac{t}{2}$$

Can we simplify $f(t)$?
Can we solve for x?

We can simplify to:

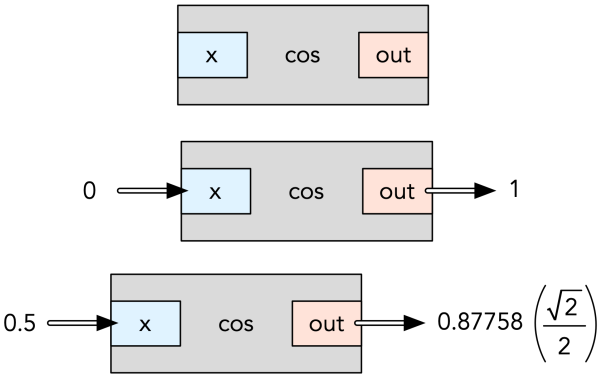
$$f(t) = 8x + e^{\frac{t}{2}j} + 1$$

In _____, we can re-arrange things pretty arbitrarily, e.g.,

$$x = \frac{f(t) - j \sin \frac{t}{2} - 1 - \cos \frac{t}{2}}{8}$$

But not in _____!

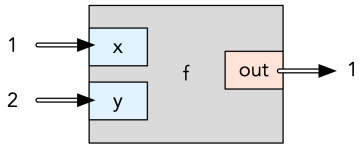
Machine model



In programming, we should instead think of a function as a machine for producing output when given input.

Multiple inputs

Can have more than one input:



Corresponding Python code:

```
Python 3.9.1

>>> f(1, 2)
1
```

We'll talk about how to *make* functions later

For now, this will let us _____ functions.

Some functions

Some functions that we can use right now:

General	Trigonometry*		Other math*	
`help`	`sin`	`asin`	`ceil`	`floor`
`print`	`cos`	`acos`	`degrees`	`radians`
	`tan`	`atan`	`factorial`	
			`gcd`	`lcm`

* Type `from math import *` before using mathematical functions

Using these simple functions, you should now be able to write Python expressions to address math and science problems in your other Engineering One courses. So, when calculating something by hand, why not *also* try writing the answer as a Python expression to help practice programming too? Then you can put your other course work to work for you in multiple ways!

More operator precedence

Operator	Description
( )	Parentheses
`x(args...)`	Function call
**	Expon.
+x, –x, ~x	Unary
*, /, //, %	Mult.
+, –	Add.

... and more to come next week