Complex

Austin Bingham
COFOUNDER - SIXTY NORTH

@austin_bingham

Robert Smallshire
COFOUNDER - SIXTY NORTH

@robsmallshire

Overview

`complex` for numbers with imaginary components

Constructing `complex` from other types

Accessing the components of `complex`

The standard `cmath` module

Using `complex` in a realistic example

Complex

complex

Built-in type for working with numbers with an imaginary component

Each has a real and an imaginary component

Python uses the "j" suffix to represent the square root of negative one

complex

```
>>> 2j
2j
>>> type(2j)
<class 'complex'>
>>> 3 + 4j
(3+4j)
>>> type(3 + 4j)
<class 'complex'>
>>> complex(3)
(3+0j)
>>> complex(-2, 3)
(-2+3j)
>>> complex('(-2+3j)')
(-2+3j)
>>> complex('-2+3j')
(-2+3j)
>>> complex('-2 + 3j')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: complex() arg is a malformed string
>>>
```

Complex Construction

`complex` can be constructed from any numeric type

This is much like `int` and `float`

The real and imaginary components are stored as `float` instances internally

`complex` Construction

```
>>> c = 3 + 5j
>>> c.real
3.0
>>> c.imag
5.0
>>> c.conjugate()
(3-5j)
>>>
```

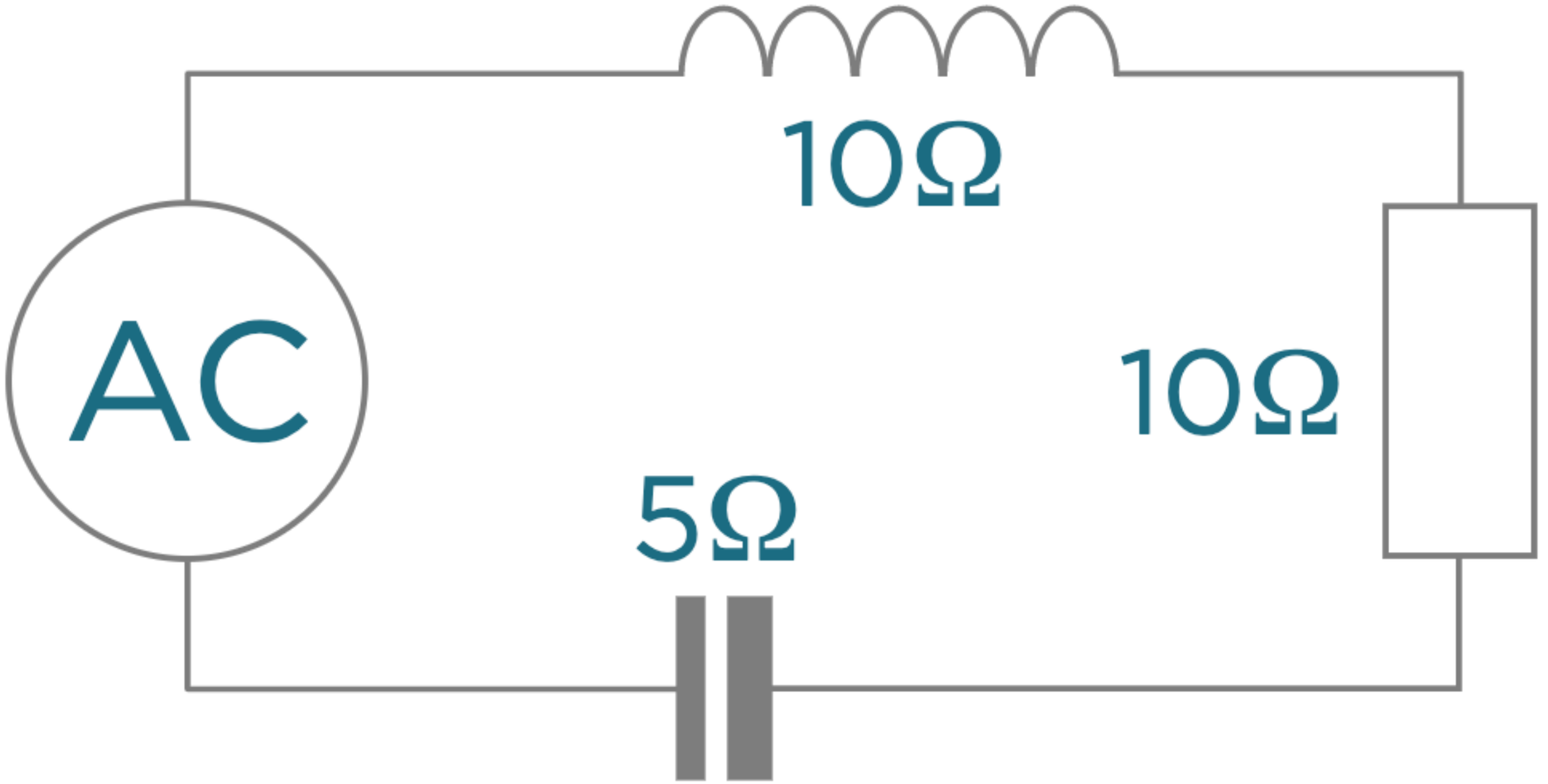
The cmath Module

cmath

```
>>> import math
>>> math.sqrt(-1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: math domain error
>>> import cmath
>>> cmath.sqrt(-1)
1j
>>> cmath.phase(1+1j)
0.7853981633974483
>>> abs(1+1j)
1.4142135623730951
>>> cmath.polar(1+1j)
(1.4142135623730951, 0.7853981633974483)
>>> modulus, phase = cmath.polar(1+1j)
>>> modulus
1.4142135623730951
>>> phase
0.7853981633974483
>>> cmath.rect(modulus, phase)
(1.0000000000000002+1j)
>>>
```

A Practical Example

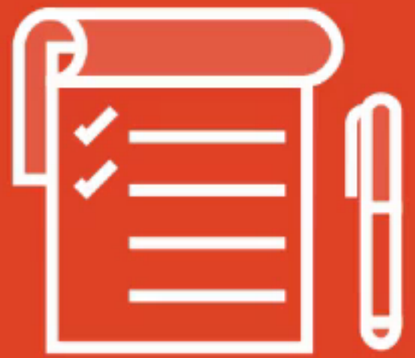
Circuit Analysis



Electrical Engineering

```
>>> def inductive(ohms):
...     return complex(0.0, ohms)
...
>>> def capacitive(ohms):
...     return complex(0.0, -ohms)
...
>>> def resistive(ohms):
...     return complex(ohms)
...
>>> def impedance(components):
...     z = sum(components)
...     return z
...
>>> impedance([inductive(10), resistive(10), capacitive(5)])
(10+5j)
>>> import cmath
>>> cmath.phase(_)
0.4636476090008061
>>> import math
>>> math.degrees(_)
26.56505117707799
>>>
```

Summary



`complex` models numbers with imaginary components

`complex` can be constructed from other numeric types or strings

Real and imaginary components stored as `float`

Functions from `math` do not work on `complex`

`cmath` provides these functions for `complex`