COMPLEX NUMBERS

The `complex` class

Constructor: `complex(x, y)`

`x` → real part

`y` → imaginary part

(rectangular coordinates)

Literals

`x + yJ`

`x + yj`

Example:

```
a = complex(1, 2)
b = 1 + 2j
a == b → True
```

`x` and `y` (the real and imaginary parts) are stored as floats

Some instance properties and methods

`.real` → returns the real part

`.imag` → returns the imaginary part

`.conjugate()` → returns the complex conjugate

```
d = 2 – 3j
d.real → 2
d.imag → -3
d.conjugate() → 2 + 3j
```

Arithmetic Operators

The standard arithmetic operators (+, -, / , *, **) work as expected with complex numbers

```
(1 + 2j) + (3 + 4j) → 4 + 6j
```

```
(1 + 2j) * (3 + 4j) → 5 + 10j
```

Real and Complex numbers can be mixed:

```
(1 + 2j) + 3  → 4 + 2j
```

```
(1 + 2j) * 3 → 3 + 6j
```

// and % operators are not supported

Other operations

The `==` and `!=` operators are supported

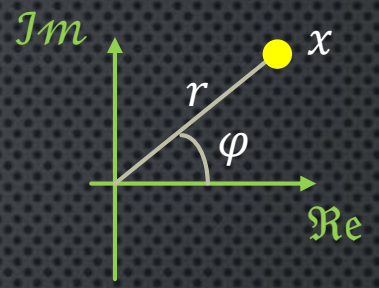
Comparison operators such as `<`, `>`, `<=` and `>=` are not supported

Functions in the `math` module will not work

Use the `cmath` module instead

- exponentials
- logs
- trigs and inverse trigs
- hyperbolics and inverse hyperbolics
- polar / rectangular conversions
- isclose

Rectangular to Polar




```
import cmath
```

`cmath.phase(x)` Returns the argument (phase) φ of the complex number x

$\varphi \in [-\pi, \pi]$ measured counter-clockwise from the real axis

`abs(x)` Returns the magnitude (r) of x

```
a = -1 + 0j
```
`cmath.phase(a)` → `3.1415…` (π) `abs(a)` → `1`

```
a = -1j
```
`cmath.phase(a)` → `-1.570…` $(-\frac{\pi}{2})$ `abs(a)` → `1`

```
a = 1 + 1j
```
`cmath.phase(a)` → `0.785…` $(\frac{\pi}{4})$ `abs(a)` → `1.414…` $(\sqrt{2})$

Polar to Rectangular

```
import cmath
```

`cmath.rect(r, phi)` Returns a complex number (rectangular coordinates) equivalent to the complex number defined by (r, phi) in polar coordinates

```
cmath.rect(math.sqrt(2), math.pi/4)  → 1 + 1j
                                       1.0000000000000002+1.0000000000000002j
```

Euler's Identity

$$e^{i\pi} + 1 = 0$$

```
cmath.exp(cmath.pi * 1j) + 1
```

→ `1.2246467991473532e-16j` binary floats tend to spoil the effect!

So, the next best thing:

```
cmath.isclose(cmath.exp(cmath.pi*1j) + 1, 0, abs_tol=0.0001)
```

→ True

Do note however the same issue with `isclose()` as we discussed in the float videos:

```
cmath.isclose(cmath.exp(cmath.pi*1j) + 1, 0)
```

→ False

Code