Python Bootcamp & Masterclass

complex numbers

gknxt

Python automatically creates complex numbers whenever an expression of the form n j is encountered where n is a real number and j is equal to $\sqrt{(-1)}$

Python puts the complex numbers in parentheses when both real and imaginary parts are present. Pure imaginary numbers (complex numbers with no real part) are shown without the parentheses (example: 8 j)

Examples:

- (3 + 2 j)
- 8 j
- (2 – 2 j)

Complex numbers are immutable (they can't be modified after they've been created)

A complex number is a number of the form a + b j , where a (real part) and b (imaginary part) are real numbers, and j is equal to √(-1)

Either the real part (a) and/or the imaginary part (b) of a complex number can be zero.

```
z = 2 + 3j
print("z = {} and its real part is {} and imaginary part is {}".format(z, z.real, z.imag))
```

```
z = (2+3j) and its real part is 2.0 and imaginary part is 3.0
```

```
z = 3j
print("z = {} and its real part is {} and imaginary part is {}".format(z, z.real, z.imag))
```

```
z = 3j and its real part is 0.0 and imaginary part is 3.0
```

```
z = 2 + 0j
print("z = {} and its real part is {} and imaginary part is {}".format(z, z.real, z.imag))
```

```
z = (2+0j) and its real part is 2.0 and imaginary part is 0.0
```

```
z = 0j
print("z = {} and its real part is {} and imaginary part is {}".format(z, z.real, z.imag))
```

```
z = 0j and its real part is 0.0 and imaginary part is 0.0
```

Python has a built-in function, **`complex( )`**, that can create complex numbers. It accepts two numeric parameters - first one represents the real part and the second one represents the imaginary part.

```
a = complex(2, 3)
print("a = {} and its real part is {} and imaginary part is {}".format(a, a.real, a.imag))
```

```
a = (2+3j) and its real part is 2.0 and imaginary part is 3.0
```

```
b = complex(0,3)
print("b = {} and its real part is {} and imaginary part is {}".format(b, b.real, b.imag))
```

```
b = 3j and its real part is 0.0 and imaginary part is 3.0
```

```
c = complex(3)
print("c = {} and its real part is {} and imaginary part is {}".format(c, c.real, c.imag))
```

```
c = (3+0j) and its real part is 3.0 and imaginary part is 0.0
```

```
d = complex()
print("d = {} and its real part is {} and imaginary part is {}".format(d, d.real, d.imag))
```

```
d = 0j and its real part is 0.0 and imaginary part is 0.0
```

The `complex( )` function can be called with zero or one or two arguments. In general, the arguments will be real numbers (integers or floats). Python also accepts boolean (`True` / `False`) as an argument as it can translate `True` as **1** and `False` as **zero**.

```
e = complex(True, False)
print("e = {} and its real part is {} and imaginary part is {}".format(e, e.real, e.imag))
```

```
e = (1+0j) and its real part is 1.0 and imaginary part is 0.0
```

```
f = complex(False, True)
print("e = {} and its real part is {} and imaginary part is {}".format(f, f.real, f.imag))
```

```
e = 1j and its real part is 0.0 and imaginary part is 1.0
```

```
g = complex(True)
print("g = {} and its real part is {} and imaginary part is {}".format(g, g.real, g.imag))
```

```
g = (1+0j) and its real part is 1.0 and imaginary part is 0.0
```

```
h = complex(False)
print("h = {} and its real part is {} and imaginary part is {}".format(h, h.real, h.imag))
```

```
h = 0j and its real part is 0.0 and imaginary part is 0.0
```

The **`complex( )`** function can take even complex numbers as its arguments. For example, complex (1, (3 + 2 j)) will be evaluated as (1 + (3 + 2 j) j) which is equal to (1 + 3j + 2 j^2)

By definition, j^2 equal to -1, so we get the result (1 + 3 j - 2) which is equal to (-1 + 3 j)

```
m = complex(1, (3+2j))
print("m = {} and its real part is {} and imaginary part is {}".format(m, m.real, m.imag))
```

```
m = (-1+3j) and its real part is -1.0 and imaginary part is 3.0
```

```
n = complex((3+2j), 1)
print("n = {} and its real part is {} and imaginary part is {}".format(n, n.real, n.imag))
```

```
n = (3+3j) and its real part is 3.0 and imaginary part is 3.0
```

```
p = complex((3+2j), (2+3j))
print("p = {} and its real part is {} and imaginary part is {}".format(p, p.real, p.imag))
```

```
p = 4j and its real part is 0.0 and imaginary part is 4.0
```

Python's built-in method, `conjugate( )`, flips the sign of the imaginary part (positive to negative or vice versa)

```
r = 2 + 3j
r.conjugate()
```

```
(2-3j)
```

```
s = 3 + 2j
s.conjugate().conjugate()
```

```
(3+2j)
```

None of Python's math module functions support complex numbers. So, Python has provided **cmath** module that has complex number versions of most of the common math functions and some complex number-specific functions such as **cmath.phase()**, **cmath.polar()**, and **cmath.rect()**, and also the **cmath.pi** and **cmath.e** constants which hold the same float values as their math module counterparts.

```
import math
math.sqrt(-1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-44-8956a835fc63> in <module>
      1 import math
----> 2 math.sqrt(-1)

ValueError: math domain error
```

```
import cmath
cmath.sqrt(-1)
```

```
1j
```

In Python, `nan` (not a number) and `inf` (infinity) have special significance. For example, `nan` is never equal to anything else, including itself!

The `cmath` module provides two complex counterparts for `nan` (not a number) and `inf` (infinity), with both having zero real parts:

```
from cmath import nanj
nanj.real, nanj.imag
```

```
(0.0, nan)
```

```
from cmath import infj
infj.real, infj.imag
```

```
(0.0, inf)
```

Online Resources

For best python resources, please visit:

gknxt.com/python/

gknxt

Python
Bootcamp
& Masterclass

Thank You

for your Rating & Review

gknxt