Python - Number Types

Python includes three numeric types to represent numbers: integer, float, and complex.

Integer:

Zero, positive and negative whole numbers without a fractional part and having unlimited precision, e.g. 1234, 0, -456.

A number having **0o** or **0O** as prefix represents an **octal** number.

For example: 0O12: equivalent to 10 (ten) in the decimal number system.

A number with **0x** or **0X** as prefix represents **hexadecimal** number.

For example: 0x12: equivalent to 18 (Eighteen) in the decimal number system.

Float:

Positive and negative real numbers with a fractional part denoted by the decimal symbol or the scientific notation using E or e, e.g. 1234.56, 3.142, -1.55, 0.23.

Scientific notation is used as a short representation to express floats having many digits.

For example:

345600000000 is represented as 3.456e11 or 3.456E11

345.56789 is represented as 3.4556789e2 or 3.4556789E2

Complex:

A complex number is a number with real and imaginary components. For example, 5 + 6j is a complex number where 5 is the real component and 6 multiplied by j is an imaginary component.

Examples: 1+2j, 10-5.5J, 5.55+2.33j, 3.11e-6+4j

Arithmetic Operators

Operator	Description	Example

+ (Addition)	Adds operands on either side of the operator.	>>> a=21
		>>> b=10
		>>> c=a+b
		>>> c
		31

- (Subtraction)	Subtracts the right-hand operand from the left-hand operand.	>>> a=21
		>>> b=10
		>>> c=a-b
		>>> c
		-11

* (Multiplication)	Multiplies values on either side of the operator.	>>> a=21
		>>> b=10
		>>> c=a*b
		>>> c
		210

/ (Division)	Divides the left-hand operand by the right-hand operand.	>>> a=21
		>>> b=10
		>>> c=a/b
		>>> c
		2.1

% (Modulus)	Returns the remainder of the division of the left-hand operand by right-hand operand.	>>> a=21
		>>> b=10
		>>> c=a%b
		>>> c
		1

** (Exponent)	Calculates the value of the left-operand raised to the right-operand.	>>> a=21
		>>> b=10
		>>> c=a**b
		>>> c
		16679880978201

// (Floor Division)	The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	>>> a=9
		>>> b=2
		>>> c=a//b
		>>> c
		4

Arithmetic Operations on Complex Numbers

Addition and subtraction of complex numbers is straightforward. Real and imaginary parts are added/subtracted to get the result.

```
>>> a=6+4j
>>> b=3+2j
>>>a+b
(9+6j)
>>>a-b
(3+2j)
```

The process of multiplying these two complex numbers is very similar to multiplying two binomials. Multiply each term in the first number by each term in the second number.

```
a=6+4j
b=3+2j
c=a*b
c=(6+4j)*(3+2j)
c=(18+12j+12j+8*-1)
c=10+24j
```

Verify this result using the Python interpreter.

```
>>> a=6+4j
>>> b=3+2j
>>>a*b
(10+24j)
```

To obtain the division of two complex numbers, multiply both sides by the **conjugate** of the denominator, which is a number with the same real part and the opposite imaginary part.

```
a=6+4j
b=3+2j
c=a/b
c=(6+4j)*(3-2j)/(3+2j)(3-2j)
c=(18-12j+12j-8*-1)/(9-6j+6j-4*-1)
c=26/13
c=2+0j
```

Verify this using the Python interpreter.

```
>>> a=6+4j
>>> b=3+2j
```

```
>>>a/b
(2+0j)
```

Built-in Functions

A numeric object of one type can be converted in another type using the following functions:

Built-in Function	Description
int	Returns the integer object from a float or a string containing digits.
float	Returns a floating-point number object from a number or string containing digits with decimal point or scientific notation.
complex	Returns a complex number with real and imaginary components.
hex	Converts a decimal integer into a hexadecimal number with 0x prefix.
oct	Converts a decimal integer in an octal representation with 0o prefix.
pow	Returns the power of the specified numbers.
abs	Returns the absolute value of a number without considering its sign.
round	Returns the rounded number.