Python Numbers, Type Conversion and Mathematics

The number data types are used to store the numeric values.

Python supports integers, floating-point numbers and complex numbers. They are defined as `int`, `float`, and `complex` classes in Python.

- `int` - holds signed integers of non-limited length.
- `float` - holds floating decimal points and it's accurate up to **15** decimal places.
- `complex` - holds complex numbers.

Python Numeric Data Type

Integers and floating points are separated by the presence or absence of a decimal point. For instance,

- **5** is an integer
- **5.42** is a floating-point number.

Complex numbers are written in the form, `x + yj`, where `x` is the real part and `y` is the imaginary part.

We can use the type() function to know which class a variable or a value belongs to.

Let's see an example,

```
num1 = 5
print(num1, 'is of type', type(num1))

num2 = 5.42
print(num2, 'is of type', type(num2))

num3 = 8+2j
print(num3, 'is of type', type(num3))
```

Output

```
5 is of type <class 'int'>
5.42 is of type <class 'float'>
(8+2j) is of type <class 'complex'>
```

In the above example, we have created three variables named *num1*, *num2* and *num3* with values **5**, **5.42**, and `8+2j` respectively.

We have also used the `type()` function to know which class a certain variable belongs to. Since,

- **5** is an integer value, `type()` returns `int` as the class of *num1* i.e `<class 'int'>`
- **5.42** is a floating value, `type()` returns float as the class of *num2* i.e `<class 'float'>`
- `1 + 2j` is a complex number, `type()` returns complex as the class of *num3* i.e `<class 'complex'>`

Number Systems

The numbers we deal with every day are of the decimal **(base 10)** number system.

But computer programmers need to work with binary **(base 2)**, hexadecimal **(base 16)** and octal **(base 8)** number systems.

In Python, we can represent these numbers by appropriately placing a prefix before that number. The following table lists these prefixes.

Number System	Prefix
Binary	`0b` or `0B`
Octal	`0o` or `0O`
Hexadecimal	`0x` or `0X`

Here are some examples

```
print(0b1101011)  # prints 107

print(0xFB + 0b10)  # prints 253

print(0o15)  # prints 13
```

Type Conversion in Python

In programming, type conversion is the process of converting one type of number into another.

Operations like addition, subtraction convert integers to float implicitly (automatically), if one of the operands is float. For example,

```
print(1 + 2.0) # prints 3.0
```

Here, we can see above that **1** (integer) is converted into **1.0** (float) for addition and the result is also a floating point number.

Explicit Type Conversion

We can also use built-in functions like int(), float() and complex() to convert between types explicitly. These functions can even convert from strings.

```
num1 = int(2.3)
print(num1)  # prints 2

num2 = int(-2.8)
print(num2)  # prints -2

num3 = float(5)
print(num3) # prints 5.0

num4 = complex('3+5j')
print(num4)  # prints (3 + 5j)
```

Here, when converting from float to integer, the number gets truncated (decimal parts are removed).

Similarly when converting from integer to float, `.0` is postfixed to the number.

To learn more about type conversion in Python, visit Python Type Conversion.

Python Random Module

Python offers the `random` module to generate random numbers or to pick a random item from an iterator.

First we need to import the `random` module. For example,

```
import random

print(random.randrange(10, 20))

list1 = ['a', 'b', 'c', 'd', 'e']

# get random item from list1
print(random.choice(list1))

# Shuffle list1
random.shuffle(list1)

# Print the shuffled list1
print(list1)

# Print random element
print(random.random())
```

Output

```
15
a
['d', 'b', 'c', 'e', 'a']
0.6716121217631744
```

To learn more about the `random` module, visit Python Random Module.

Python Mathematics

Python offers the `math` module to carry out different mathematics like trigonometry, logarithms, probability and statistics, etc. For example,

```
import math

print(math.pi)

print(math.cos(math.pi))

print(math.exp(10))

print(math.log10(1000))

print(math.sinh(1))

print(math.factorial(6))
```

Output

```
3.141592653589793
-1.0
22026.465794806718
3.0
1.1752011936438014
720
```

Here is the full list of functions and attributes available in the Python math module.

Table of Contents