Data Types

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categorie.

- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview

Getting the Data Type

You can get the data type of any object by using the type() function.

```
# Print the data type of the variable x:

x = 5
print(type(x))
```

```
<class 'int'>
```

Setting the Data Type

In Python, the data type is set when you assign a value to a variable.

Example	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions.

Example	Data Type
`x = str("Hello World")`	str
`x = int(20)`	int
`x = float(20.5)`	float
`x = complex(1j)`	complex
`x = list(("apple", "banana", "cherry"))`	list
`x = tuple(("apple", "banana", "cherry"))`	tuple
`x = range(6)`	range
`x = dict(name="John", age=36)`	dict
`x = set(("apple", "banana", "cherry"))`	set
`x = frozenset(("apple", "banana", "cherry"))`	frozenset
`x = bool(5)`	bool
`x = bytes(5)`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

Python Numbers

There are three numeric types in Python.

- int
- float
- complex Variables of numeric types are created when you assign a value to them.

```
x = 1    # int
```

```
y = 2.8  # float
z = 1j   # complex
```

To verify the type of any object in Python, use the type() function.

```
print(type(x))
print(type(y))
print(type(z))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

Int

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

```
x = 1
y = 35656222554887711
z = -3255522

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'int'>
<class 'int'>
<class 'int'>
```

Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

```
x = 1.10
y = 1.0
z = -35.59

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
```

Float can also be scientific numbers with an "e" to indicate the power of 10.

```
x = 35e3
```

```
y = 12E4
z = -87.7e100

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
```

Complex

Complex numbers are written with a "j" as the imaginary part.

```
x = 3+5j
y = 5j
z = -5j

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'complex'>
<class 'complex'>
<class 'complex'>
```

Type Conversion

You can convert from one type to another with the int(), float(), and complex() methods.

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex

#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

#convert from int to complex:
c = complex(x)

print(a)
print(b)
print(c)

print(type(a))
print(type(b))
print(type(c))
```

```
1.0
2
(1+0j)
<class 'float'>
<class 'int'>
<class 'complex'>
```

Note: You cannot convert complex numbers into another number type.

Random Number

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers.

```
# Import the random module, and display a random number between 1 and 9:

import random

print(random.randrange(1, 10))
```

```
2
```

Boolean Values

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer.

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

```
True
False
False
```

Evaluate Values and Variables

The bool() function allows you to evaluate any value, and give you True or False in return.

```
print(bool("Hello"))
print(bool(15))
```

```
True
```

```
True
```

```
# Evaluate two variables:

x = "Hello"
y = 15

print(bool(x))
print(bool(y))
```

```
True
True
```

Most Values are True

Almost any value is evaluated to True if it has some sort of content.

- Any string is True, except empty strings.
- Any number is True, except 0.
- Any list, tuple, set, and dictionary are True, except empty ones.

```
bool("abc")
bool(123)
bool(["apple", "cherry", "banana"])
```

```
True
```

Some Values are False

In fact, there are not many values that evaluate to False, except empty values, such as (), [], {}, "", the number 0, and the value None. And of course the value False evaluates to False.

```
  bool(False)
  bool(None)
  bool(0)
  bool("")
  bool(())
  bool([])
  bool({})
```

```
False
```