

Python 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 categories:

| | |
|------------------------|-------------------------------------|
| 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 |
| None Type: | NoneType |

Getting the Data Type

We can get the data type of any object by using the **type() function**:

Example

Print the data type of the variable x:

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

Output:

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

Setting the Data Type

In Python, the data type is set when we assign a value to a variable:

| Example | Data Type | |
|-------------------|---|--|
| x = "Hello World" | str | <pre>x = "Hello World" #display x: print(x) #display the data type of x: print(type(x)) Output: Hello World <class 'str'></pre> |
| x = 20 | <pre>int x = 20 #display x: print(x) #display the data type of x:</pre> | |

| | | |
|---------------------|---|---|
| | <pre>print(type(x))</pre> <p><u>Output:</u></p> <pre>20</pre> <pre><class 'int'></pre> | |
| <pre>x = 20.5</pre> | <pre>float</pre> | <pre>x = 20.5</pre> <pre>#display x:</pre> <pre>print(x)</pre> <pre>#display the data type of x:</pre> <pre>print(type(x))</pre> <p><u>Output:</u></p> <pre>20.5</pre> <pre><class 'float'></pre> |
| <pre>x = 1j</pre> | <p>Complex</p> <pre>x = 1j</pre> <pre>#display x:</pre> <pre>print(x)</pre> <pre>#display the data type</pre> <pre>of x:</pre> <pre>print(type(x))</pre> <p><u>Output</u></p> | |

| | | |
|--|---|--|
| | <pre>l <class 'complex'></pre> | |
| <pre>x = ["apple", "banana", "cherry"]</pre> | <pre>List x = ["apple", "banana", "cherry"] #display x: print(x) #display the data type of x: print(type(x)) Output: ['apple', 'banana', 'cherry'] <class 'list'></pre> | |
| <pre>x = ("apple", "banana", "cherry")</pre> | <pre>tuple</pre> | |
| <pre>x = range(6)</pre> | <pre>range</pre> | |

| | | |
|---|--|--|
| <code>x = {"name" : "John", "age" : 36}</code> | dict | |
| <code>x = {"apple", "banana", "cherry"}</code> | set | |
| <code>x = frozenset({"apple", "banana", "cherry"})</code> | frozenset | |
| <code>x = True</code> | <p>bool</p> <p>x = True</p> <p>#display x:</p> <p>print(x)</p> <p>#display the data type of x:</p> <p>print(type(x))</p> <p>Output:</p> <p>True</p> <p><class 'bool'></p> | |
| <code>x = b"Hello"</code> | bytes | |
| <code>x = bytearray(5)</code> | bytearray | |

| | | |
|---|------------|--|
| <code>x = memoryview(bytes(5))</code> | memoryview | |
| <code>x = None</code> | NoneType | |

Setting the Specific Data Type

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

| Example | Data Type | Try it |
|--|-----------|--------|
| <code>x = str("Hello World")</code> | str | |
| <code>x = int(20)</code> | int | |
| <code>x = float(20.5)</code> | float | |
| <code>x = complex(1j)</code> | complex | |
| <code>x = list(("apple", "banana", "cherry"))</code> | list | |

Dr. Uma

```
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

Dr. Uma

Python Numbers

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:

Example

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

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

Example

```
x = 1
y = 2.8
z = 1j

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

Output:

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

Int

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

Example

Integers:

```
x = 1
```

```
y = 35656222554887711
```

```
z = -3255522
```

```
print(type(x))
```

```
print(type(y))
```

```
print(type(z))
```

Output:

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

Dr. Uma

Float

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

Example

Floats:

```
x = 1.10
```

```
y = 1.0
```

```
z = -35.59
```

```
print(type(x))
```

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

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

Example

Floats:

```
x = 35e3
y = 12E4
z = -87.7e100
```

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

Output:

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

Dr. Uma

Complex

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

Example

Complex:

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

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

output:

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

Type Conversion

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

Example

Convert from one type to another:

#convert from int to float:

```
x = float(1)
```

#convert from float to int:

```
y = int(2.8)
```

#convert from int to complex:

```
z = complex(1)
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

```
print(type(x))
```

```
print(type(y))
```

```
print(type(z))
```

Dr. Uma

Output:

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:

Example

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

```
import random
```

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

Output:

1

Dr. Uma