Data Types

Data Types

You would like to see the rest of the paper, or know where it came from. Without knowing the context, it is hard to say what MIX means. It *could* be 1009 in Roman numerals, or it could be the English word "mix" (which itself has several meanings), or it could be the last name of the old-time radio hero Tom Mix. It could be part of a label on a music CD, "MTV Dance MIX", or part of a label on a bottle, "BLOODY MARY MIX". Or maybe you are looking at it upside down and it should be "XIW". Of course, it might not be English at all. Without knowing the context, a string of letters has little meaning.

Computer memory stores arbitrary bit patterns. As with a string of letters, the meaning of a string of bits depends on how it is used. The particular scheme that is being used for a particular string of bits is a **data type.**

A **data type**

- Is a scheme for using bits to represent values.
- Values are not just numbers, but any kind of data that a computer can process.
- All values in a computer are represented using one data type or another.

For example

```
0000000001100111
```

is a pattern of 16 bits that might be found somewhere in computer memory. What does it represent?

Without knowing more about how the above pattern is being used, it is impossible to say what it represents. The type `short` is one of Java's data types. If the pattern is of data type `short`, then it represents the value 103 (one hundred and three).

There are Many Data Types

You might be tempted say that the pattern 0000000000000000 represents "zero". But it doesn't necessarily. Even such an obvious pattern has no automatic meaning.

If you were told that the above pattern were of type short, then you would know that it represents the integer zero. Here is another pattern:

1111111110011001	As a unsigned int, this pattern is 65433
1111111110011001	As a signed int, this pattern is -103

There are uncountably many types of data that can be represented in the memory of a computer. If specific patterns always had specific meanings, then only a few types of data could be represented. This would be much too restrictive. Many data types are *invented* by programmers as they write programs. Without this flexibility computers would be much less useful.

Not all machines use memory this way. A simple electronic calculator, for example, uses memory for one purpose only: to store floating point numbers. It uses only one data type, and can do only those few things with that data type that it has been wired to do. The engineers who designed the calculator decided how to represent numbers with bit strings, and then designed the electronics to work with just those strings. This is too restrictive for a general purpose computer.

Data Type

Primitive data type

- Numbers
  - int, float, complex
- String
- Byte
- List
- Tuple
- Set
- Dictionary

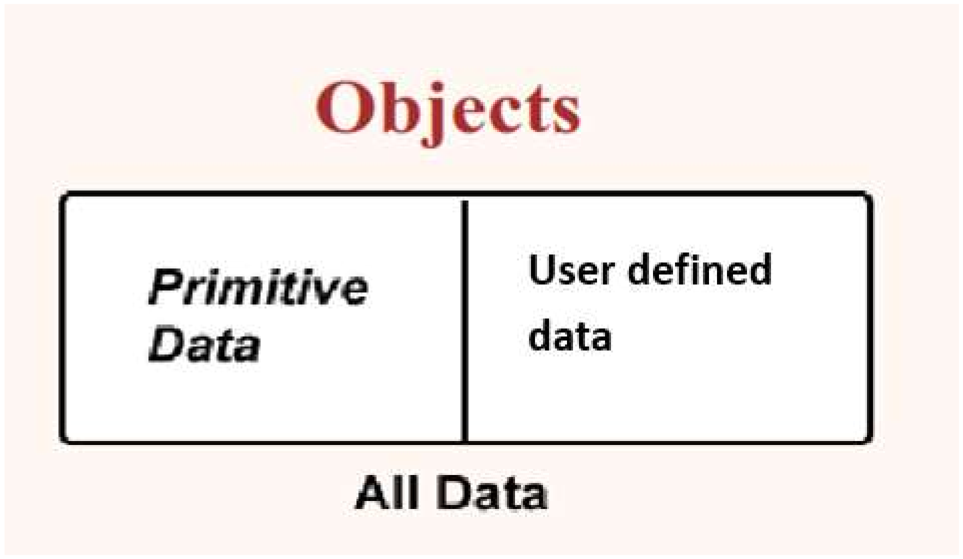
User Defined data type

- Class

Data can be single or sequence

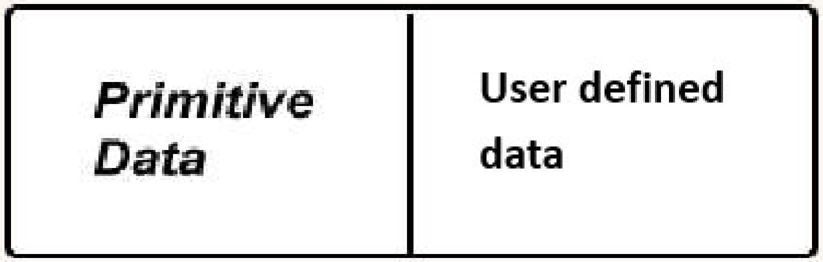
Data

- Data in python created from the data types
- All data in python falls into one of two categories: primitive data and user defined data
- These data refer as object



Objects

All data in Python falls into one of two categories: **primitive data** and **objects**. There are only eight primitive data types. However, Python has *many* types of objects, and you can invent as many others as you need. Any data type you invent will be a type of object.



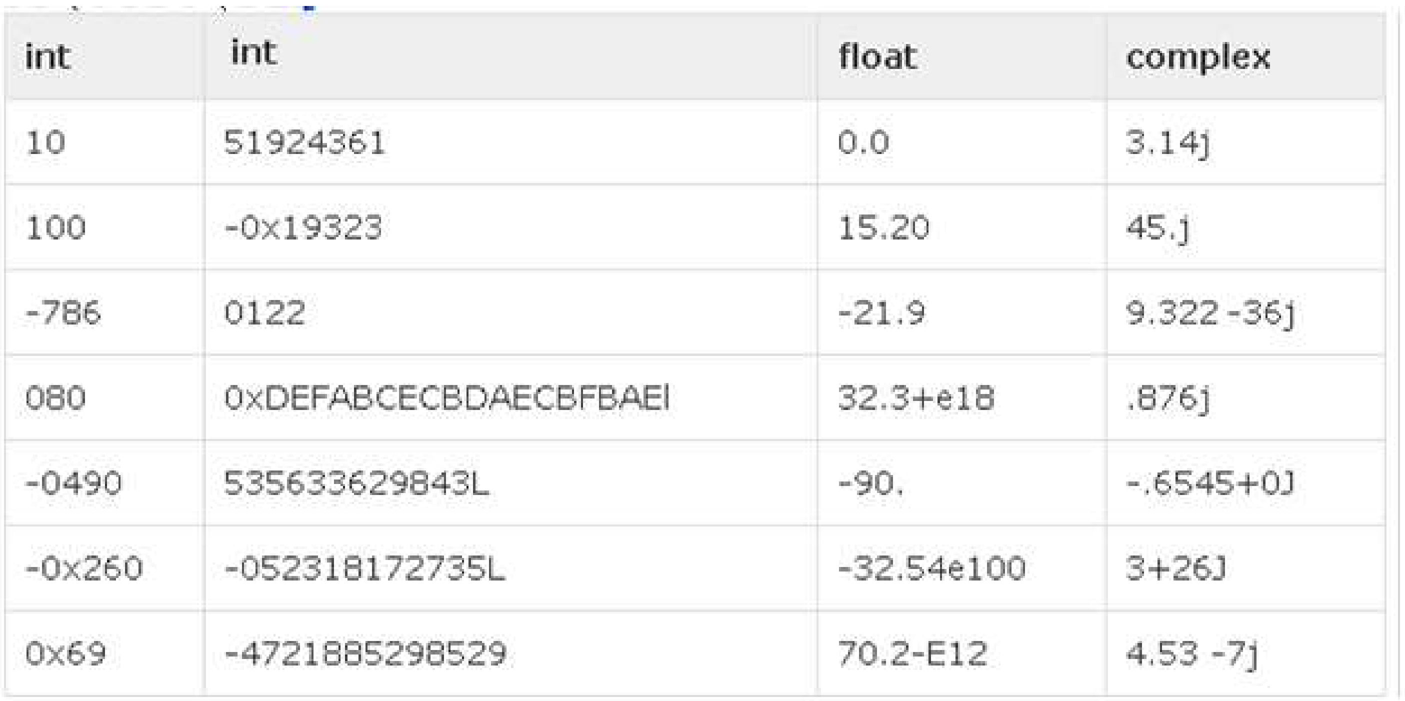
All Data

Much more will be said about objects in future chapters (since Java is a *object oriented* programming language). The following is all you need to know, for now:

- A primitive data value uses a small, fixed number of bytes.
- There are only eight primitive data types.
- A programmer can not create new primitive data types.

- An object is a big block of data. An object may use many bytes of memory.
- An object usually consists of many internal pieces.
- The data type of an object is called its **class**.
- Many classes are already defined in Java.
- A programmer can invent new classes to meet the particular needs of a program.

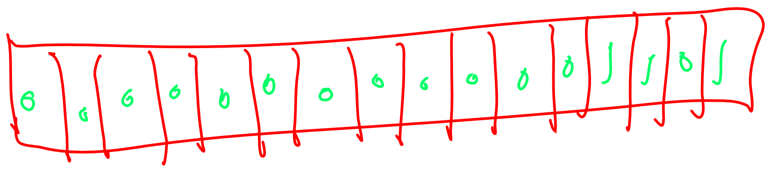
Numerical Data



int	int	float	complex
10	51924361	0.0	3.14j
100	-0x19323	15.20	45.j
-786	0122	-21.9	9.322 -36j
080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3+26J
0x69	-4721885298529	70.2-E12	4.53 -7j

Integer Numbers

- Just is represented as binary
- 13 == 1101



34.125 101011.11011 0.10101111011×2^6

- IEEE numbers are stored using a kind of scientific notation.

$$\pm \text{mantissa} * 2^{\text{exponent}}$$

- We can represent floating-point numbers with three binary fields: a sign bit s, an exponent field e, and a fraction field f.

s	e	f

- The IEEE 754 standard defines several different precisions.
  - Single precision numbers include an 8-bit exponent field and a 23-bit fraction, for a total of 32 bits.
  - Double precision numbers have an 11-bit exponent field and a 52-bit fraction, for a total of 64 bits.

floating point

639.6875

639	1001111111
.6875	.1011

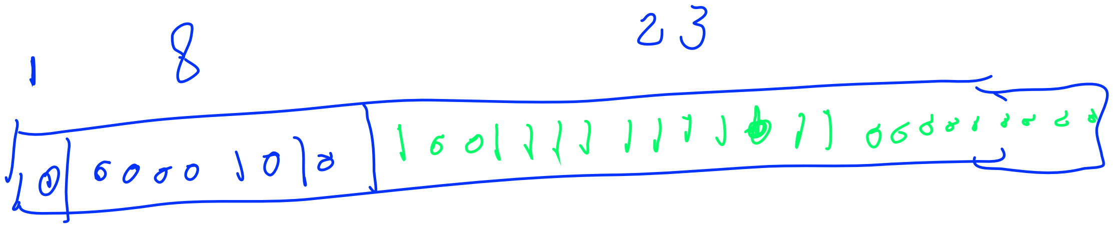
$$1001111111.1011 = 0.10011111111011 * 2^{10}$$

$0.6875 \times 2 = \underline{1}.3750$

$0.3750 \times 2 = \underline{0}.750$

$0.75 \times 2 = \underline{1}.5$

$0.5 \times 2 = \underline{1}.0$



13.34126

$13 = 1101$

0	$0.34126 \times 2 = 0.68252$
1	$0.68252 \times 2 = 1.36504$
2	$0.36504 \times 2 = 0.73008$
3	$0.73008 \times 2 = 1.46016$
4	$0.46016 \times 2 = 0.92032$
5	$0.92032 \times 2 = 1.84064$
6	$0.84064 \times 2 = 1.68128$
7	$0.68128 \times 2 = 1.36256$
8	$0.36256 \times 2 = 0.72512$
9	$0.72512 \times 2 = 1.45024$
10	$0.45024 \times 2 = 0.90048$
11	$0.90048 \times 2 = 1.80096$
12	$0.80096 \times 2 = 1.60192$
13	$0.60192 \times 2 = 1.20384$
14	$0.20384 \times 2 = 0.40768$
15	$0.40768 \times 2 = 0.81536$
16	$0.81536 \times 2 = 1.63072$
17	$0.63072 \times 2 = 1.26144$
18	$0.26144 \times 2 = 0.52288$
19	$0.52288 \times 2 = 1.04576$
20	$0.04576 \times 2 = 0.09152$
21	$0.09152 \times 2 = 0.18304$
22	$0.18304 \times 2 = 0.36608$
23	$0.36608 \times 2 = 0.73216$

24	$0.73216 \times 2 = 1.46432$
25	$0.46432 \times 2 = 0.92864$
26	$0.92864 \times 2 = 1.85728$
27	$0.85728 \times 2 = 1.71456$
28	$0.71456 \times 2 = 1.42912$
29	$0.42912 \times 2 = 0.85824$
30	$0.85824 \times 2 = 1.71648$
31	$0.71648 \times 2 = 1.43296$
32	$0.43296 \times 2 = 0.86592$
33	$0.86592 \times 2 = 1.73184$
34	$0.73184 \times 2 = 1.46368$
35	$0.46368 \times 2 = 0.92736$
36	$0.92736 \times 2 = 1.85472$
37	$0.85472 \times 2 = 1.70944$
38	$0.70944 \times 2 = 1.41888$
39	$0.41888 \times 2 = 0.83776$
40	$0.83776 \times 2 = 1.67552$
41	$0.67552 \times 2 = 1.35104$
42	$0.35104 \times 2 = 0.70208$
43	$0.70208 \times 2 = 1.40416$
44	$0.40416 \times 2 = 0.80832$
45	$0.80832 \times 2 = 1.61664$
46	$0.61664 \times 2 = 1.23328$
47	$0.23328 \times 2 = 0.46656$
48	$0.46656 \times 2 = 0.93312$
49	$0.93312 \times 2 = 1.86624$
50	$0.86624 \times 2 = 1.73248$
51	$0.73248 \times 2 = 1.46496$
52	$0.46496 \times 2 = 0.92992$
53	$0.92992 \times 2 = 1.85984$
54	$0.85984 \times 2 = 1.71962$

$0.34126 = 0101\ 0111\ 010111\ 0011010000\ 101110\ 11011011\ 10110101\ 100111011$

floating point

13.34126

13 = 1101

0.34126 = 0101 0111 010111 0011010000 101110 11011011 10110101 100111011

13.34126 = 1101. 0101 0111 010111 0011010000 101110 11011011 10110101 100111011

$$13.34126 = 0.\underbrace{\overbrace{1101\ 0101\ 0111\ 0101\ 1100\ 110}\quad 1000\ 0101\ 1101\ 1011\ 01110110\ 10110}\quad 011\ 1011 \times 2^4$$

0	0000 0100	1101 0101 0111 0101 1100 110

13.3412599861621856689453125

0	000 0000 0100	1101 0101 0111 0101 1100 110 1000 0101 1101 1011 0111 0110 10110

13.341260000000000007780442956573

floating point

0.0859372

$0.0859372 = 0.0001011$ 0 0 0 0 0 0 1 1

$0.0859372 = 0.1011 \times 2^{-3}$

0	1111 1101	1011 0000 0000 0000 0000 000

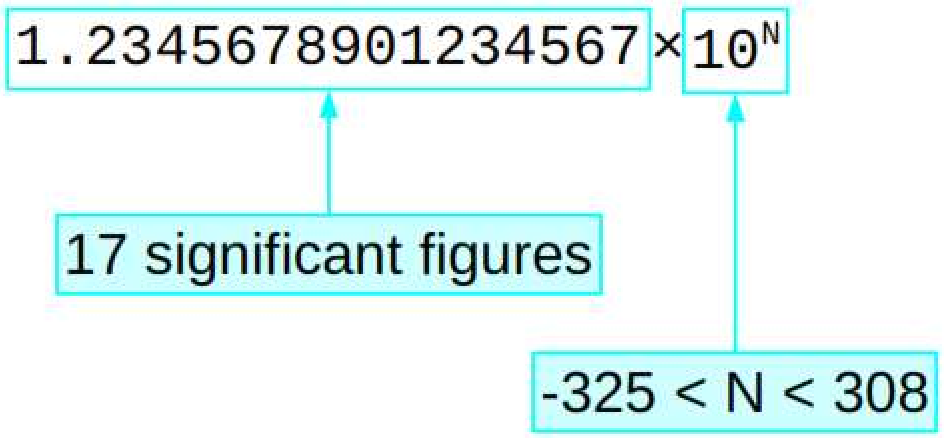
24	$0.0859375 \times 2 = 0.1718750$
25	$0.1718750 \times 2 = 0.3437500$
26	$0.3437500 \times 2 = 0.6870000$
27	$0.6870000 \times 2 = 1.3750000$
28	$0.3750000 \times 2 = 0.7500000$
29	$0.7500000 \times 2 = 1.5000000$
30	$0.5000000 \times 2 = 1.0000000$

floating point scientific notation

1.8446744073709552 e+19

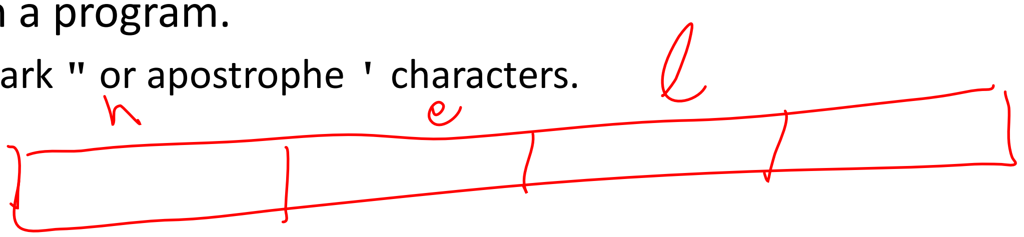
$1.8446744073709552 \times 10^{19}$

Floating point limits



Strings

- **string**: A sequence of text characters in a program.
  - Strings start and end with quotation mark `"` or apostrophe `'` characters.
  - Examples:

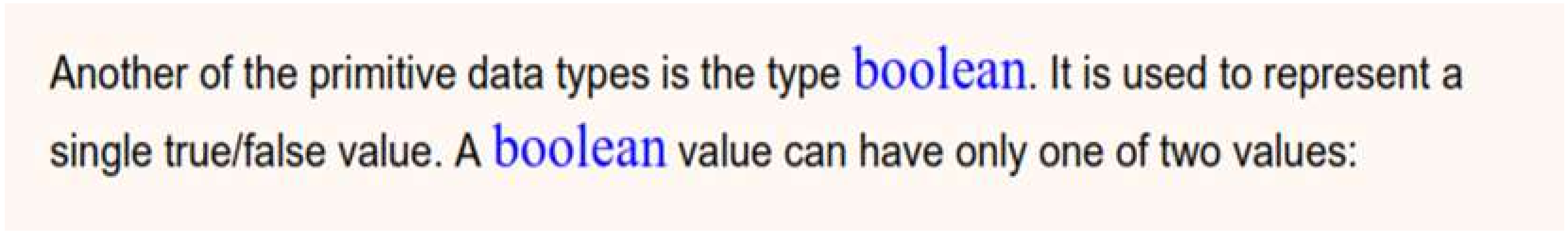


```
"hello"
"This is a string"
"This, too, is a string.   It can be very long!"
```

- A string may not span across multiple lines or contain a " character.

```
"This is not
a legal String."
"This is not a "legal" String either."
```

Boolean


Another of the primitive data types is the type boolean. It is used to represent a single true/false value. A boolean value can have only one of two values:

True False