Python Types I: Numbers and strings

Jean Mark Gawron

Linguistics 572
San Diego State University

September 2, 2020

Python tutorials

1. Older version of Python tutorial, better for beginners Actually written by Guido van Rossum.
2. Current Python docs tutorial
3. Google's Python class

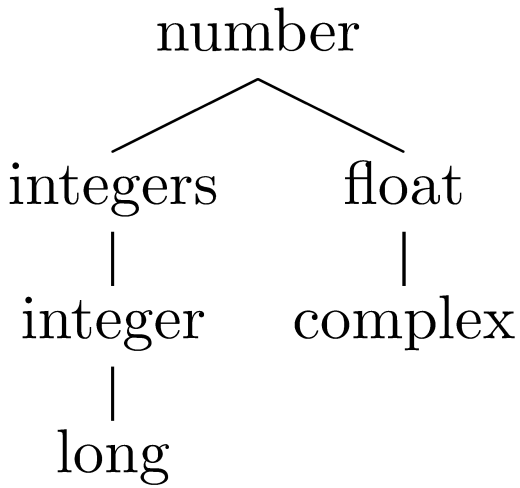
Numbers

```
>>> X = 3
>>> X
3
>>> type(3)
<class 'int'>
>>> type(X)
<class 'int'>
>>> X = 1.2
>>> type(1.2)
<class 'float'>
>>> type(X)
<class 'float'>
```

Type casting

```
>>> (X, Y) = (3, 1.2)
>>> Z = 3 + 1.2
>>> type(X), type(Y)
(<class 'int'>, <class 'float'>)
>>> type(Z)
<class 'float'>
```

Python type hierarchy



Python number types

Complex numbers

```
>>> type(0j)
<class 'complex'>
>>> 0 == 0j
True
>>> X = 3j+2
>>> type(X)
<class 'complex'>
>>> from numpy import log
>>> log(-1 + 0j)    # Works for complex number
3.141592653589793j
>>> log(-1)         # Fails for real number
nan
```

String literals

```
>>> X = "frog"
>>> type(X)
<class 'str'>
>>> Y = 'frog'
>>> type(Y)
<class 'str'>
>>> X == Y
True
```

When entering string literals, delimiters (' ' or " " or """ """) are necessary.

String literals with quotes marks

```
>>> X = "The big dog laughed and said, 'Hello, Jeremy.'"
>>> Y = 'The big dog laughed and said, "Hello, Jeremy."'
>>> X == Y
False
```

Multiline strings

```
>>> X = """
... Beautiful is better than ugly.
... Explicit is better than implicit.
... Simple is better than complex.
... Complex is better than complicated.
... """
```

Special characters in strings

```
>>> Z = "x\ty"
>>> print(Z)
x	y
>>> X = "\n Beautiful is better than ugly.\n Explicit is better
>>> print(X)

 Beautiful is better than ugly.
 Explicit is better than implicit.
 Simple is betterthan complex.
 Complex is better than complicated.
```

\t tab
\n new line

Concatenating strings

```
>>> X = "The dog"
>>> Y = " barked"
>>> X + Y              # X + Y is a new string
'The dog barked'
>>> X                  # X unchanged
'The dog'
>>> Y                  # Y unchanged; strings are immutable
' barked'
```

Casting limitations

```
>>> X = "The dog"
>>> Y = 3
>>> X + Y
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

Takeaways

1. All python data has a type
2. In these slides we looked at two data types: numbers and strings
3. Builtin python operations (like "+") work only on certain types
4. The result of every python operation has to have a type, and when the opearyion has arguments of different types, python has to chose a type for the result. This is called **casting**.