Data Types

- Dr. Junjie Zhang
- Huaxia Chinese School at Mason, OH

Data Types

A variable can store data, which has a specific type.

- Built-In Data Types:
  - Known by the Python robot by default.
- Customized Data Types:
  - Designed by software engineers like you.

Built-In Data Types

- Simple Data Types
  - integer, float, boolean, string, and etc.
- Complex Data Types
  - list, set, dict, and etc.

type()

If you want to know the type of data or data stored in a variable, use `type()`.

```
type("here")
type(29)
type(1.3)
type(True)
```

Let me know your outputs.

type()

If you want to know the type of data or data stored in a variable, use `type()`.

```
a = "here"
type(a)
a = 29
type(a)
a = 1.3
type(a)
a = True
type(a)
```

Let me know your outputs.

Types

- **Int**, or integer, is a whole number, positive or negative, without decimals, of unlimited length.
- **Float**, or "floating point number" is a number, positive or negative, containing one or more decimals.
- **Booleans** represent one of two values: True or False.
- **Strings** in python are surrounded by either single quotation marks, or double quotation marks.

Two Important Questions

- What data types matter?
  - The same operation may have different effectiveness with different data types.
  - Certain operations make sense only if they are applied on data with specific types.
- Can we convert one data type to another one?
  - Yes to some of them.

Data Types Matter for Operators

Let's give a look at `+`

```
a = 1
b = 2
c = a + b
print(c)
```

```
a = "1"
b = "2"
c = a + b
print(c)
```

Data Types Matter for Operators

Let's give a look at `*`

```
a = 1
b = 2
c = a * b
print(c)
```

```
a = "1"
b = "2"
c = a * b
print(c)
```

Convert Data Types

A more professional way to say "data type converting" is **casting**!

Python Casting

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Python Casting

```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

Python Casting

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

More Discussion

- Boolean
- String

Boolean Values

Question: What operations lead to boolean values?
Answer: Many, but there are some commonly used ones.

">", "<", ">=", "<=", "=="

Boolean Values

```
result = 5 > 3
#result = 5 < 3
#result = 5 == 3
print(type(result))
print(result)
```

There will be more operators and functions that lead to boolean values. We will learn them as this course proceeds.

Logic Operators: Operators Using Boolean Values

- not
  - output = not input
- and
  - output = input1 and input2
- or
  - output = input1 and input2

Typically, we expect input, input1, and input2 are with the boolean type.

Logic Operators

Q = A and B
Q = A or B
Q = not A

AND

A	B	Q
F	F	F
F	T	F
T	F	F
T	T	T

OR

A	B	Q
F	F	F
F	T	T
T	F	T
T	T	T

NOT

A	Q
F	T
T	F

Logic Operators

```
a = 3
b = 4
resutl1 = a > b
result2 = (a > b) and (a < b)
result3 = (a > b) or (a < b)
result4 = not (a > b)
result5 = (not (a > b)) and (a == b)
result6 = not ((a > b) and (a == b))
result7 = (not ((a > b) and (a == b))) or True
result8 = (not ((a > b) and (a == b))) or False
```

Strings

You will use strings a lot to output the results

```
a = "I "
b = "Love "
c = "Hate "
d = "Cilantro"
r1 = a + b + d
r2 = a + c + d
print(type(r1))
print(r1)
print(type(r2))
print(r2)
```

Strings

```
a = 5
b = 6
c = a + b
r = "the result is " + c
print(r)
```

How to fix it?

End