MATH1401

Fall 2021

Lecture 4

Data Types

Review

Lecture 3 Checklist

- Use operators +,-,*,/, **
- Assign values to a name
- Call functions min, max, abs
- Select columns of a table
- Sort columns of a table
- Filter rows of a table by a condition

Review: Table Operations

- `t.select(label)` - constructs a new table with just the specified columns
- `t.drop(label)` - constructs a new table in which the specified columns are omitted
- `t.sort(label)` - constructs a new table with rows sorted by the specified column
- `t.where(label, condition)` - constructs a new table with just the rows that match the condition

(Demo)

Overview

Class Checklist

- **Lab 1 – Due Date** : Tuesday 8/31 – 5 PM
  - Graded Questions : 3.1.2, 3.3.1, 3.3.2, 4.1.1, 5.1, 5.1.1
- **Homework 1 – Due Date** : Friday 9/3 – 5 PM
  - Graded Questions : 1.1, 2.1,2.1.1, 2.2, 3.1-3.3, 4.1-4.3, 5.1-5.4, 7.1
- **Quiz 2** – Tuesday: 8/31 – Covers Chapter 3
- **Quiz 3** – Thursday: 9/2 – Covers Chapter 4

Lecture 4 Checklist – Chapter 4

- int/floats
- strings
- bools
- comparisons

Lecture 4 Checklist - Programming

- Use strings, integers, floats, booleans
- Be able to use the string methods:

  s.upper, s.replace(s1,s2), str(),concatention
- Make comparisons between numerical values and strings
- Be able to use the type functions:

  type(),str(),int(),float()

Numbers

Ints and Floats

Python has two real number types

- `int`: an integer of any size
- `float`: a number with an optional fractional part

An `int` never has a decimal point; a **float** always does

Ints and Floats

Python has two real number types

- **`int`**: an integer of any size
- **`float`**: a number with an optional fractional part

An **`int`** never has a decimal point; a **float** always does

A **`float`** might be printed using scientific notation

Ints and Floats

Python has two real number types

- `int`: an integer of any size
- `float`: a number with an optional fractional part

An `int` never has a decimal point; a **float** always does

A `float` might be printed using scientific notation

Three limitations of float values:

- They have limited size (but the limit is huge)
- They have limited precision of 15-16 decimal places
- After arithmetic, the final few decimal places can be wrong

(Demo)

Strings

Text and Strings

A string value is a snippet of text of any length

- `'a'`
- `'word'`
- `"there can be 2 sentences. Here's the second!"`

Text and Strings

A string value is a snippet of text of any length

- `'a'`
- `'word'`
- `"there can be 2 sentences. Here's the second!"`

Strings consisting of numbers can be converted to numbers

- `int('12')`
- `float('1.2')`

Text and Strings

A string value is a snippet of text of any length

- `'a'`
- `'word'`
- `"there can be 2 sentences. Here's the second!"`

Strings consisting of numbers can be converted to numbers

- `int('12')`
- `float('1.2')`

Any value can be converted to a string

- `str(5)`

Text and Strings

A string value is a snippet of text of any length

- `'a'`
- `'word'`
- `"there can be 2 sentences. Here's the second!"`

Strings consisting of numbers can be converted to numbers

- `int('12')`
- `float('1.2')`

Any value can be converted to a string

- `str(5)`

String Methods

String_name.upper()

- `"there can!".upper() -> “THERE CAN!”`

String Methods

String_name.upper()

- `"there can!".upper() -> “THERE CAN!”`

String_name.replace(s1,s2)

- `"there can!".replace(‘there’,’cannoli’) -> “cannoli can!”`

(Demo)

Discussion Question

Assume you have run the following statements:

```
x = 3
y = '4'
z = '5.6'
```

What's the source of the error in each example?

A. `x + y`
B. `x + int(y + z)`
C. `str(x) + int(y)`
D. `y + float(z)`

Types

Every value has a type

We've seen 5 types so far:

- `int: 2`
- `float: 2.2`
- `str: 'Red fish, blue fish'`
- `builtin_function_or_method: abs`
- `Table`

Every value has a type

We've seen 5 types so far:

- `int: 2`
- `float: 2.2`
- `str: 'Red fish, blue fish'`
- `builtin_function_or_method: abs`
- `Table`

The `type` function can tell you the type of a value

- `type(2)`
- `type(2 + 2)`

Every value has a type

We've seen 5 types so far:

- `int: 2`
- `float: 2.2`
- `str: 'Red fish, blue fish'`
- `builtin_function_or_method: abs`
- `Table`

The `type` function can tell you the type of a value

- `type(2)`
- `type(2 + 2)`

An expression's "type" is based on its value, not how it looks

- `x = 2`
- `type(x)`

Conversions

Strings that contain numbers can be converted to numbers

- `int('12')`
- `float('1.2')`
- ~~`float('one point two')`~~ `# Not a good idea!`

(Demo)

Conversions

Strings that contain numbers can be converted to numbers

- `int('12')`
- `float('1.2')`
- ~~`float('one point two')`~~ `# Not a good idea!`

Any value can be converted to a string

- `str(5)`

(Demo)

Conversions

Strings that contain numbers can be converted to numbers

- `int('12')`
- `float('1.2')`
- ~~`float('one point two')`~~ `# Not a good idea!`

Any value can be converted to a string

- `str(5)`

Numbers can be converted to other numeric types

- `float(1)`
- `int(1.2) # DANGER: loses information!`

(Demo)

Comparisons

Conversions

Comparison	Operator	True example	False Example
Less than	<	2 < 3	2 < 2
Greater than	>	3>2	3>3
Less than or equal	<=	2 <= 2	3 <= 2
Greater or equal	>=	3 >= 3	2 >= 3
Equal	==	3 == 3	3 == 2
Not equal	!=	3 != 2	2 != 2

Booleans

A Boolean represents True or False value.

- `True`
- `False`

- Every comparison that is made results in a bool
- `3 > 10 -> False`
- `'hello' != 'hi' -> True`