BC COMS 1016: Intro to Comp Thinking & Data Science

Lecture 2
Data Types & Expressions
Announcements

- **HW 00**
  - Due Thursday (11/29)
  - Individual assignment

- **Lab 00**
  - Due Wednesday (11/28)
Labs

- Labs help solidify the concepts
- Completing labs will help you master the course material
- Grade for labs will be based on Gradescope
- How’d lab00 go?
Labs & Late day policies

- **15** late days
- **Can be used for labs**
- Only 2 late days per assignment
I have stopped!

Seriously, why am I still apologizing for my kids walking into my Zoom calls?
Subject: “BC1016: <few words summary>”

Intro boiler plate
- “Dear Prof Poliak,
  I hope you are doing ….”

Send screenshot of issue
- Popular for data science & software development
- Focus on mastering language fundamentals
- Learn through practice and doing
- Follow along in the demos
Statement Statements

- Statements perform an action
  - don’t have a value
- Assignment statement changes the meaning of the name to the left of the = symbol
- The name is bound to a value

```
hours_per wk = 24*7
```
Anatomy of a Call Expression

What function to call

Argument to the function

f(27)

"Call f on 27."
Anatomy of a Call Expression

max (15, 27)
Numbers
Two real number types in Python

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

An int never has a decimal point; a float does

A float might be printed using scientific notation
Limitations on float values

- Floats have limited size (the limit is huge)
- Floats have limited precision of 15-16 decimal places
- After arithmetic, the final few decimal places can be wrong
Strings
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)` becomes “5”
Discussion Question

Assume you have run the following statements:

\[
\begin{align*}
x &= 3 \\
y &= '4' \\
z &= '5.6'
\end{align*}
\]

What is the source of the error in each example?

A. \(x + y\)   
B. \(x + \text{int}(y + z)\)   
C. \(\text{str}(x) + \text{int}(y)\)   
D. \(y + \text{float}(z)\)
We’ve seen 5 types so far:

- int: 2
- float: 2.2
- str: ‘Red fish, blue fish’
- builtin_function_or_method: abs, max, min
Types – Every value has a type

The type function tells you the type of a value

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

An expression’s “type” is based on its value

- x = 2
- type(x) = ???
Strings that contain numbers can be converted to numbers

- `int("12")`
- `float("1.2")`
- `float("one point two")`  # Not a good idea
Conversions

Any value can be converted to a string

- `str(6)`

Numbers can be converted to other numeric types

- `float(1)`
- `int(2.3)`. # DANGER: why is this a bad idea
Tables
### Table Structure

- **Table** is a sequence of labeled columns
- **Row**: represents one individual
- **Column**: represents one attribute of the individuals

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Area (m2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>California</td>
<td>CA</td>
<td>163696</td>
</tr>
<tr>
<td>Nevada</td>
<td>NV</td>
<td>110567</td>
</tr>
</tbody>
</table>
Creating a Table

- `Table.read_table(filename)` – reads a table from a spreadsheet

- `Table()` – an empty table
Creating and extending tables:
  • `Table().with_column` and `Table.read_table`

Finding the size:
  • `num_rows`, `num_columns`

Referring to columns: labels, relabeling and indices
  • `labels` and `relabeled`; column indices start at 0
Some Table Operations

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

These operations create a new table
Table methods

- Accessing data in a column
  - `Column` takes a label or index and returns an array

- Using array methods to work with data in columns
  - `item`, `sum`, `min`, `max`, and so on

- Creating new tables containing some of the original columns
  - `select`, `drop`