Reminders

- HW00 due tonight
  - Individual assignment
  - Only 41 submissions (as of 10am today)
  - You can use 2 late days

- Lab01 due Monday

- Lab00:
  - If you haven't gotten it in yet, do it before the other assignments

- HW01
  - Will be released tonight
  - Due Thursday 02/03
Office Hours & Next few classes

- **Today:**
  - Adam 1pm-2pm after class

- **Next week:**
  - Tuesday still remote
  - Thursday TBD

- **Tuesday Feb 8th:**
  - TA review or watch last year’s recording or no class
  - Your choice
<table>
<thead>
<tr>
<th>Question 2</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>- / 2.0 pts</td>
</tr>
<tr>
<td>Question 3</td>
<td>- / 1.0 pts</td>
</tr>
<tr>
<td>Question 4</td>
<td>- / 2.0 pts</td>
</tr>
<tr>
<td>Question 5</td>
<td>- / 1.0 pts</td>
</tr>
<tr>
<td>Question 6</td>
<td>- / 1.0 pts</td>
</tr>
<tr>
<td>Question 7</td>
<td>- / 1.0 pts</td>
</tr>
<tr>
<td>Question 8</td>
<td>- / 0.0 pts</td>
</tr>
</tbody>
</table>

**Failed Tests:**
- q2_1 - Public (0.0/1.0)
- q2_2 - Public (0.0/1.0)
- q3_1 - Public (0.0/1.0)
- q3_2 - Public (0.0/1.0)
- q3_3 - Public (0.0/1.0)
- q4_1 - Public (0.0/1.0)
- q4_2 - Public (0.0/1.0)
- q5_1 - Public (0.0/1.0)
- q5_2 - Public (0.0/1.0)
- q5_3 - Public (0.0/3.0)
- q5_4 - Public (0.0/1.0)
- q8_1 - Public (0.0/1.0)
- q7_1 - Public (0.0/1.0)

**Passed Tests:**
- q9_1 - Public (0.0/0.0)
Course Outline

- **Exploration**  
  - Introduction to Python  
  - Working with data

- **Inference**  
  - Probability  
  - Statistics

- **Prediction**  
  - Machine Learning  
  - Regression & Classification

Week 1 - 5

Week 6 - 10

Week 11-14
Course Outline

- Exploration Week 1 - 5
  - Discover patterns
  - Articulate insights

- Inference Week 6 - 10
  - Make reliable conclusions about the world
  - Statistics is useful

- Prediction Week 11-14
  - Informed guesses about unseen data
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, max, min
- Table
Tables
Table Structure

- A **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>
Table methods

- 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
Tables – select and drop

- `t.select(…)` – constructs a new table with just the specified columns
- `t.drop(…)` – constructs a new table in which the specified columns are omitted

- These operations create a new table
Tables – select and drop

- `.select(<Column Name>)`
  - Returns a new table with the specified columns

- `.select(<Int i>)`
  - Returns a new table with the column at index I

- `.drop(<Column Name>)`
  - Returns a new table without the specified columns

- `.drop(<Int i>)`
  - Returns a new table without the column at index i
Some Table Operations

- `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

More are listed at [http://coms1016.barnard.edu/python-reference.html](http://coms1016.barnard.edu/python-reference.html)
An array contains a sequence of values

- All elements of an array should have the same type
- Arithmetic is applied to each element individually
- Adding arrays add elements (if same length!)
- A column of a table is in an array
  - All values in a single column are the same type
A range is an array of consecutive numbers

- `np.arange(end)`: An array of increasing integers from 0 up to `end`
- `np.arange(start, end)`: An array of increasing integers from `start` up to `end`
- `np.arange(start, end, step)`: A range with `step` between consecutive values

The range always include `start` but excludes `end`
<table>
<thead>
<tr>
<th>Name</th>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>max(array)</code></td>
<td>3.3</td>
<td>Returns the maximum value of an array</td>
</tr>
<tr>
<td><code>min(array)</code></td>
<td>3.3</td>
<td>Returns the minimum value of an array</td>
</tr>
<tr>
<td><code>sum(array)</code></td>
<td>3.3</td>
<td>Returns the sum of the values in an array</td>
</tr>
<tr>
<td><code>abs(num), np.abs(array)</code></td>
<td>3.3</td>
<td>Take the absolute value of number or each number in an array.</td>
</tr>
<tr>
<td><code>round(num), np.round(array)</code></td>
<td>3.3</td>
<td>Round number or array of numbers to the nearest integer.</td>
</tr>
<tr>
<td><code>len(array)</code></td>
<td>3.3</td>
<td>Returns the length (number of elements) of an array</td>
</tr>
<tr>
<td><code>make_array(val1, val2, ...)</code></td>
<td>5</td>
<td>Makes a numpy array with the values passed in</td>
</tr>
<tr>
<td><code>np.average(array) np.mean(array)</code></td>
<td>5.1</td>
<td>Returns the mean value of an array</td>
</tr>
<tr>
<td><code>np.std(array)</code></td>
<td>14.2</td>
<td>Returns the standard deviation of an array</td>
</tr>
<tr>
<td><code>np.diff(array)</code></td>
<td>5.1</td>
<td>Returns a new array of size <code>len(arr)-1</code> with elements equal to the difference between adjacent elements; val_2 - val_1, val_3 - val_2, etc.</td>
</tr>
<tr>
<td><code>np.sqrt(array)</code></td>
<td>5.1</td>
<td>Returns an array with the square root of each element</td>
</tr>
<tr>
<td><code>np.arange(start, stop, step)</code></td>
<td>5.2</td>
<td>An array of numbers starting with <code>start</code>, going up in increments of <code>step</code>, and going up to but excluding <code>stop</code>. When <code>start</code> and/or <code>step</code> are left out, default values are used in their place. Default step is 1; default start is 0.</td>
</tr>
<tr>
<td><code>array.item(index)</code></td>
<td>5.3</td>
<td>Returns the i-th item in an array (remember Python indices start at 0)</td>
</tr>
<tr>
<td><code>np.random.choice(array, n) np.random.choice(array)</code></td>
<td>9</td>
<td>Picks one (by default) or some number <code>n</code> of items from an array at random. By default, with replacement.</td>
</tr>
<tr>
<td><code>np.count_nonzero(array)</code></td>
<td>9</td>
<td>Returns the number of non-zero (or <code>True</code>) elements in an array.</td>
</tr>
<tr>
<td><code>np.append(array, item)</code></td>
<td>9.2</td>
<td>Returns a copy of the input array with <code>item</code> (must be the same type as the other entries in the array) appended to the end.</td>
</tr>
<tr>
<td><code>percentile(percentile, array)</code></td>
<td>13.1</td>
<td>Returns the corresponding percentile of an array.</td>
</tr>
</tbody>
</table>
Tables & Arrays
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*
Questions in notebook
The table `nba` has columns `PLAYER`, `POSITION`, and `SALARY`

table = Table.read_table('https://www.inferentialthinking.com/data/nba_salaries.csv')

1. Create an array containing the names of all centers (C) who make more than $15M/year

   centers = table.where('POSITION', 'C')
   centers.where('\'15-\'16 SALARY', are.above(15)).column('PLAYER')

Answer:

'Dwight Howard', 'Roy Hibbert', 'Marc Gasol', 'Enes Kanter', 'DeMarcus Cousins'
Attribute Types
All values in a column of a table should be both the same type and be comparable to each other

- **Numerical** – values are from a numerical scale
  - Numerical measurements are ordered
  - Differences are meaningful

- **Categorical** – values from a fixed inventory
  - May or may not have an ordering
  - Categories are the same or different
Values as numbers are not guaranteed to be numerical

- Census example: SEX code (0, 1, 2)
- Arithmetic on these “numbers” is meaningless
- The variable SEX is still categorical, even though numbers were used for the categories
Census Data
The Decennial Census

- Every ten years, Census Bureau counts how many people there are in the U.S.

- Census Bureau estimates how many people are in US during the other 9 years

- U.S. Constitution Article 1, Section 2:
  - “Representatives and direct Taxes shall be apportioned among the several States ... according to their respective Numbers ...”
▪ https://www2.census.gov/programs-surveys/popest/datasets/

▪ https://www2.census.gov/programs-surveys/popest/datasets/2010-2015/national/totals/

▪ demo