



Week 6 – Table Aggregation, Conditionals, Iteration and Sampling, Probability

Slides by Suraj Rampure

Fall 2017

Administrative

Our midterm is **next Friday during lecture!** This weekend, several review materials will be posted. They will all be found at <http://data8.org/fa17/resources.html>.

Next week's lab will be a midterm review session. There won't be checkoffs, but you should still attend. Send me a quick email at suraj.rampure@berkeley.edu telling me what you want me to cover!

Project 1 is due Thursday. You shouldn't work on it during lab until you're finished the worksheet and lab notebook, though.

group and pivot

Group

When we group a table, we create a new table with a summary of the values in the original table.

`<table name>.group(<column name(s)>, optional <function>)`

To group, we use the above syntax. As per usual, our table names are variables and column names are strings (or integer indexes). We can include either a single string, or an array of strings, depending on the number of columns we want to group by. The function argument is optional – if you don't include it, the default function is **count**.

Group

When we group a table, we create a new table with a summary of the values in the original table.

users: Table

manufacturer	model	carrier	monthly_payment
Apple	iPhone X	T-Mobile	79.99
Samsung	Galaxy Note 8	Verizon	84.99
Samsung	Galaxy S7	AT&T	64.99
Apple	iPhone 7+	AT&T	51.99
Apple	iPhone SE	Sprint	43.99
HTC	One	Sprint	76.99
Google	Pixel 2	AT&T	101.99
Samsung	Galaxy S8	Verizon	84.99
Google	Pixel XL	T-Mobile	59.99
Apple	iPhone X	T-Mobile	66.99

manufacturer	count
Apple	4
Google	2
HTC	1
Samsung	3

Result of calling
`users.group("manufacturer")`

manufacturer	count
Apple	4
Google	2
HTC	1
Samsung	3

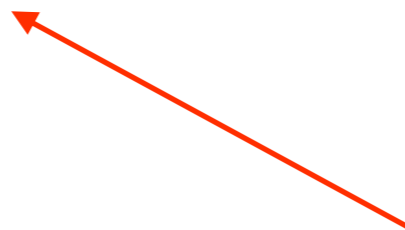
Result of calling
`users.group("manufacturer")`

manufacturer	model	count
Apple	iPhone 7+	1
Apple	iPhone SE	1
Apple	iPhone X	2
Google	Pixel 2	1
Google	Pixel XL	1
HTC	One	1
Samsung	Galaxy Note 8	1
Samsung	Galaxy S7	1
Samsung	Galaxy S8	1

Result of calling
`users.group(["manufacturer", "model"])`

manufacturer	model mean	carrier mean	monthly_payment mean
Apple			60.74
Google			80.99
HTC			76.99
Samsung			78.3233

Result of calling
`users.group("manufacturer", np.mean)`



Notice that there are more rows in this table than in the other two – this is because instead of having one row for each unique **“manufacturer”** value, there is one row for each unique combination of **“manufacturer”** and **“model”** value.

Pivot

When we **pivot** a table, we restructure the tables and columns. We can either use 2 or 4 arguments, as outlined below.

```
<table name>.pivot(<column name>, <column name>)
```

or

```
<table name>.pivot(<column name>, <column name>, <column name>, <function>)
```

- **First argument:** Name of column that will make up the columns of the new table
- **Second argument:** Name of the column that will make up the rows of the new table
- **Third argument (optional):** Name of column whose values will make up the new table
- **Fourth argument (optional):** Function used for aggregation (**mean**, **sum**, etc.)

manufacturer	AT&T	Sprint	T-Mobile	Verizon
Apple	1	1	2	0
Google	1	0	1	0
HTC	0	1	0	0
Samsung	1	0	0	2

Result of calling `users.pivot("carrier", "manufacturer")`

carrier	Apple	Google	HTC	Samsung
AT&T	51.99	101.99	0	64.99
Sprint	43.99	0	76.99	0
T-Mobile	73.49	59.99	0	0
Verizon	0	0	0	84.99

Result of calling `users.pivot("manufacturer", "carrier", "monthly_payment", np.mean)`

Conditionals

Booleans

TOP DEFINITION



boolean

1. noun. When someone tryna flex but you know he ain't real so you call b■■s■■.

2. adj. When you know you taking the L and you need to express your tightness.

1. "Boy I'll break your ankles"

"Stfu thats some boolean"

2. "I just took the biggest L on that math test"

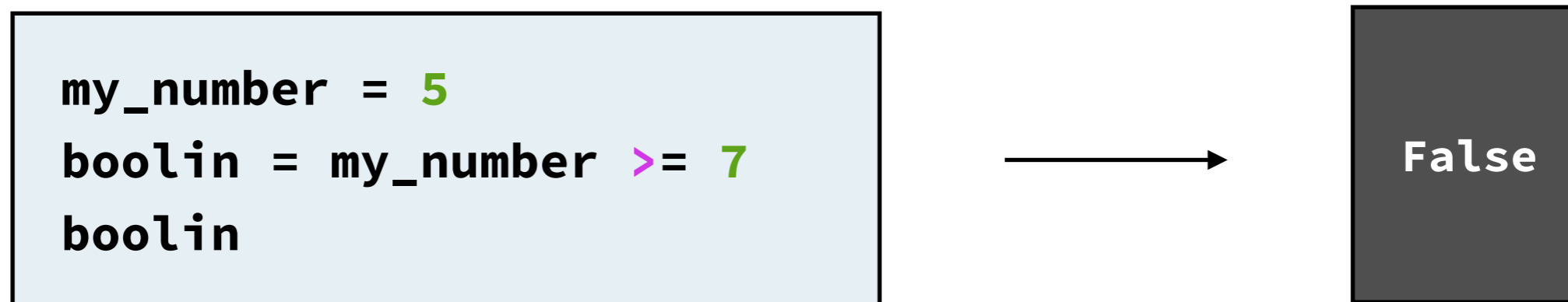
"Yeah boi that s■■ was boolean"

Booleans

A **boolean** is a variable with two possible values – **True** and **False**.



We can assign variables based on the value of an expression that evaluates to a boolean:



Fun fact: **True** is equivalent to **1** and **False** is equivalent to **0**. Try running `5*True-False`: what happens?

Conditionals

Conditional statements allow us to control what code is run by specifying certain conditions that must be **True**.

```
if <condition 1>:  
    <output 1>  
elif <condition 2>:  
    <output 2>  
.  
.  
.  
else:  
    <output n>
```

Python evaluates the output of the first condition in this sequence that is **True**.

Conditionals

Conditional statements allow us to control what code is run by specifying certain conditions that must be **True**.

```
grade = 78
if grade >= 95:
    print("A+")
elif 90 <= grade <= 94:
    print("A")
elif 80 <= grade <= 89:
    print("B")
elif 70 <= grade <= 79:
    print("C")
else:
    print("D or F")
```



"C"

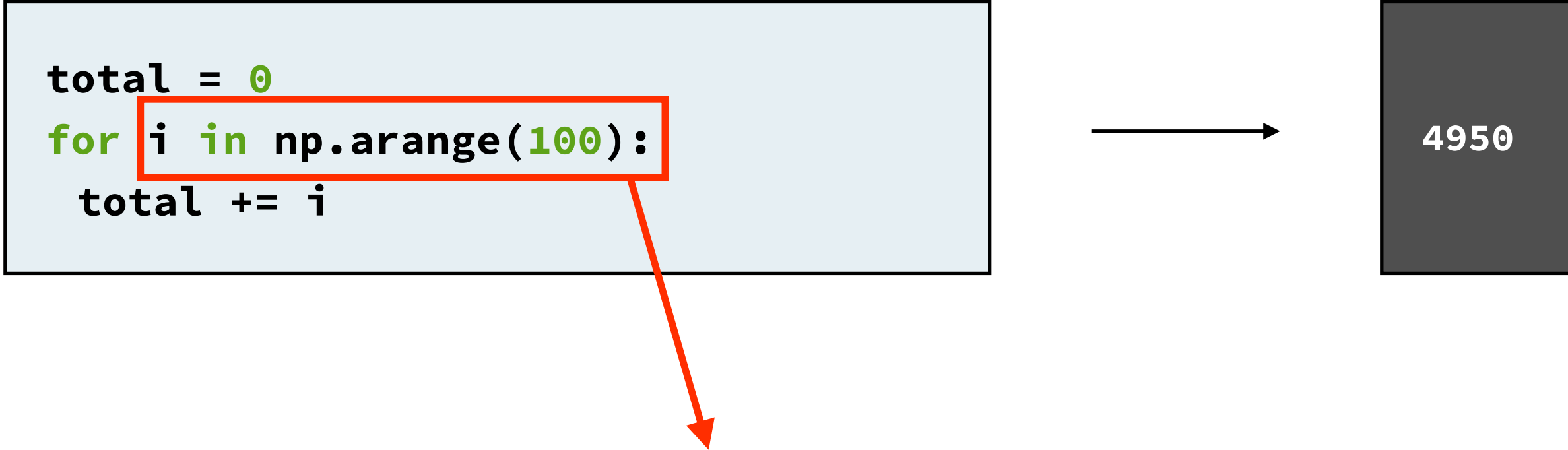
Python evaluates the output of the first condition in this sequence that is **True**.

Iterations and Sampling

Iteration

For loops allow us to repeat code several times. The most common use case for this in Data 8 is for sampling.

```
total = 0
for i in np.arange(100):
    total += i
```



4950

We repeat the code underneath this line **100** times: once for each value of **i** in **[0, 1, 2, 3, 4, ..., 98, 99]**. We don't necessarily need to use the value of **i** in our code – see the following slide.

Iteration

This is a classic sampling simulation. Here, we've modeled a fair coin with an array of two elements – "H" and "T." By calling `np.random.choice(coin)`, we **uniformly at random** select one element from this array. We do this 1000 times and count the number of times we got heads, and divide by 1000, to get the proportion of heads.

```
coin = ["H", "T"]
outcomes = []
for i in np.arange(1000):
    flip = np.random.choice(coin)
    outcomes = np.append(outcomes, flip)

p = np.count_nonzero(outcomes == "H")/1000
```



0.458

It's okay if this doesn't make a ton of sense right now – you'll get a lot of practice with this shortly.

Probability

Probability

For the purposes of this class, there are only three things you need to remember with regards to probability. Each question you see will use some combination of these rules in a different way.

1

Sum of 1: Given some event, the sum of the probabilities of all possible outcomes is 1.

+

Addition Rule: If two events are disjoint, meaning that the probability of them both occurring is 0, then we add their probabilities to find the probability that either one happens.

*

Multiplication Rule: If two events are independent, meaning that one doesn't affect the other, then we multiply their probabilities to find the probability that they both happen.

Probability

Sample Problem: Suppose we flip a coin three times in a row. What is the probability that we do not get three heads in a row (i.e. something other than HHH)?

$$P(\text{HHH}) + P(\text{not HHH}) = 1$$

$$P(\text{not HHH}) = 1 - P(\text{HHH})$$

$$P(\text{HHH}) = (1/2) * (1/2) * (1/2) = (1/2)^3 = 1/8$$

$$\text{Therefore, } P(\text{not HHH}) = 1 - 1/8 = 7/8$$