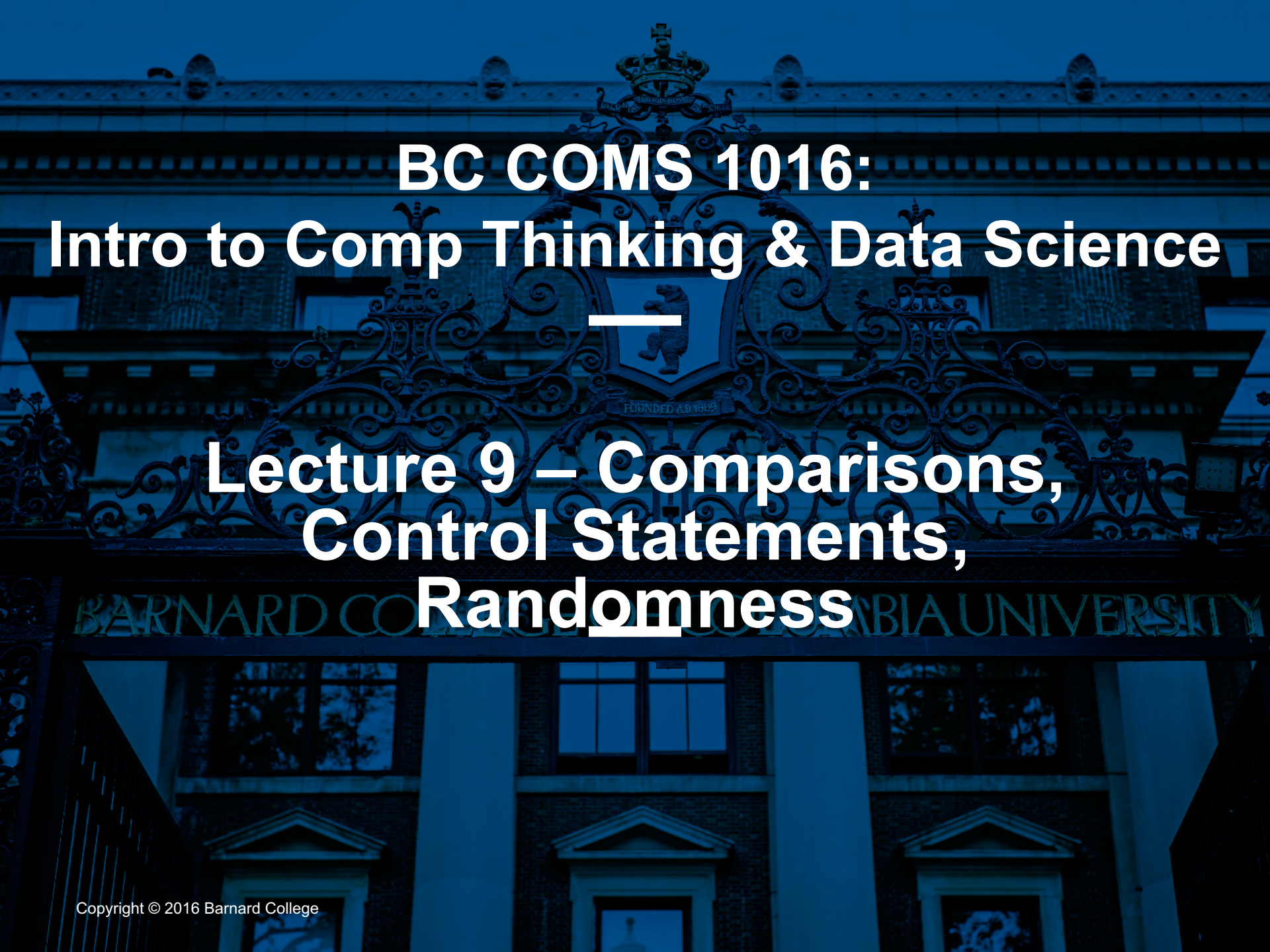


BC COMS 1016: Intro to Comp Thinking & Data Science

—

Lecture 9 – Comparisons, Control Statements, Randomness

Announcements

- HW03 - Functions, Histograms, and Groups
  - Due Monday (02/21)
- No lab this week
- Checkpoint/Project 1:
  - Paired assignment that covers the previous section of the course material
  - Released this morning (02/17) and in two weeks Thursday 03/03

Group vs Pivot

pivot groups together rows that share a combination of values.

It differs from group because it organizes the resulting values in a grid

Group vs Pivot

Pivot

- One combo of grouping variables **per entry**

Group

- One combo of grouping variables **per row**

```
more_cones.pivot('Flavor', 'Color', values='Price', collect=sum)
```

Color	bubblegum	chocolate	strawberry
dark brown	0	10.5	0
light brown	0	4.75	0
pink	4.75	0	8.8

```
more_cones.group(['Flavor', 'Color'], sum)
```

Flavor	Color	Price sum
bubblegum	pink	4.75
chocolate	dark brown	10.5
chocolate	light brown	4.75
strawberry	pink	8.8

Group vs Pivot

Pivot	Group
One combo of grouping variables **per entry**	One combo of grouping variables **per row**
Two grouping variables: columns and rows	**Any number** of grouping variables
Aggregate values of **values column**	Aggregate values of **all other columns** in table
Missing combos **= 0 (or empty string)**	Missing combos **absent**

Table Review

t.select(column, …) or t.drop(column, …)
t.take([row, …]) or t.exclude([row, …])
t.sort(column, descending=False)
t.where(column, are.condition(…))
t.apply(function, column, …)
t.group(column) or t.group(column, function)
t.group([column, …]) or t.group([column, …], function)
t.pivot(cols, rows) or t.pivot(cols, rows, vals, function)
t.join(column, other_table, other_table_column)

https://coms1016.barnard.edu/python-reference.html

Comparisons

Comparison Operators

Operator	Table predicate
==	are.equal_to
!=	are.not_equal_to
>	are.above
>=	are.above_or_equal_to
<	are.below
<=	are.below_or_equal_to

The result of a comparison expression is a bool value:

True, False

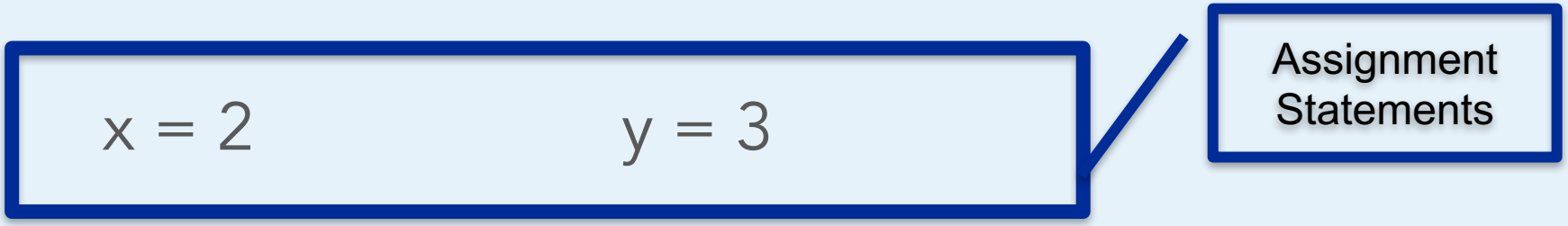
Comparison Operators

The result of a comparison expression is a bool value

x = 2 y = 3

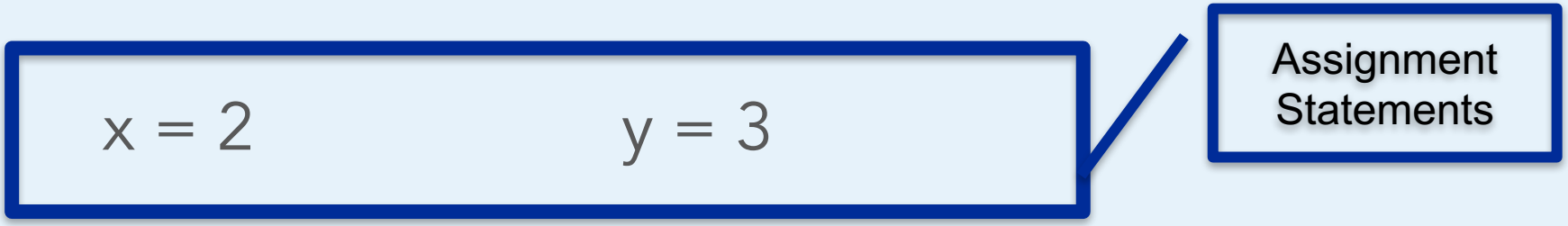
Comparison Operators

The result of a comparison expression is a bool value



Comparison Operators

The result of a comparison expression is a bool value

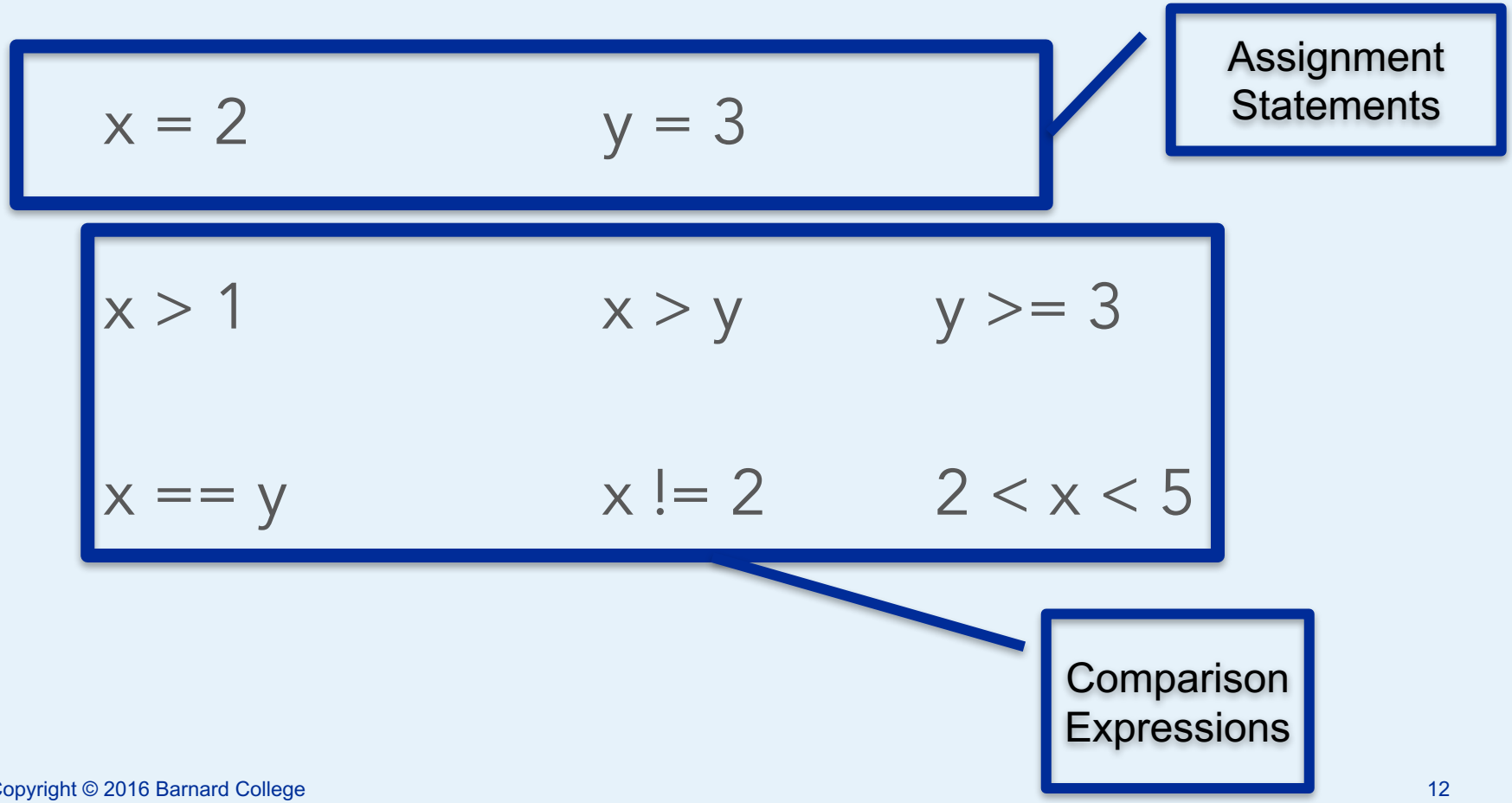


x > 1 x > y y >= 3

x == y x != 2 2 < x < 5

Comparison Operators

The result of a comparison expression is a bool value



Combining Comparisons

The result of a comparison expression is a bool value

a = True

b = False

not b

a or b

a and not b

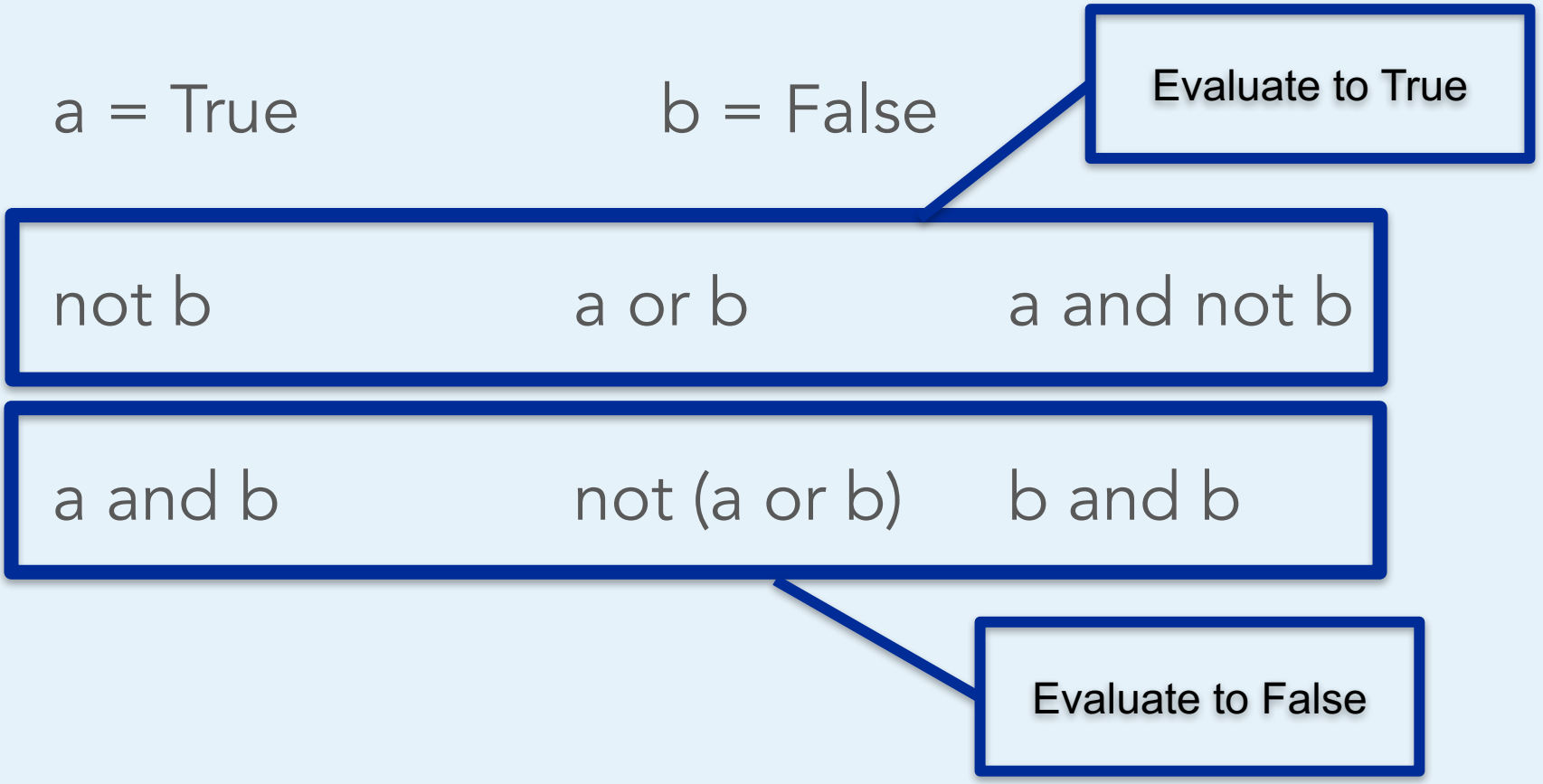
a and b

not (a or b)

b and b

Combining Comparisons

The result of a comparison expression is a bool value



Aggregating Comparisons

Summing an array or list of bool values count the number of True values

```
1 + 0 + 1                    == 2
True + False + True          == 2
sum([1    ,  0    ,  1])    == 2
sum([True, False, True)]     == 2
```

Control Statements

Control Statements

These statements *control* the sequence of computations that are performed

- The keywords if and for begin control statements
- The purpose of if is to define functions that choose different behavior based on their arguments

Random Selection

Random Selection

np.random.choice

- Selects at random
- With replacement
- From an array
- A specific number of times

np.random.choice(some_array, sample_size)

Appending Arrays

A Longer Array

- np.append(array_1, value):
  - new array with value appended to array_1
  - value has to be of the same type as elements of array_1

- np.append(array_1, array_2):
  - new array with array_2 appended to array_1
  - Elements of array_2 have to be of the same type as elements of array_1

Iteration

for **statements**

- for is a keyword that begins a control statement
- The purpose of for is to perform a computation for every element in a list or array