Lecture 13

Conditionals and Loops

MATH1401

Fall 2021

Review

Class Checklist

- **Homework 4 - Due Date** : Tuesday: 10/5 – 9 PM
  - Graded Questions:1.1,1.3, 1.4-.1.6, 2.1, 2.3

- **Homework 5 - Due Date** : Friday: 10/8 – 9 PM
  - Graded Questions:1.1-1.4

Grouping

The `group` method aggregates all rows with the same value for a column into a single row in the resulting table.

- *First argument*: Which columns to group by
- *Second argument*: (Optional) How to combine values

```
sky.group(['city', 'material'], max)
```

Pivot

- Cross-classifies according to two categorical variables
- Two required arguments:
    - First: variable that forms column labels of grid
    - Second: variable that forms row labels of grid

- **Table_name.pivot(‘label1’,’label2’)** – Creates pivot table and classifies based on label1 and label2.

Pivot

- Cross-classifies according to two categorical variables
- Two optional arguments (include **both** or **neither**)
    - `values`=’column_label_to_aggregate’
    - `collect`=function_to_aggregate_with

- **Table_name.pivot(‘label1’,’label2’,’numerical’,function)** – Applys function to numerical value for each group defined by label1 and label 2

(Demo 12)

Summary – Sections 9.0-9.3

- Understand randomness
- **Experiments** – Randomly assign groups to remove confounding factors
- **Simulation** – Rerun Experiment and check whether whether conclusions are due to randomness or treatment

Checklist – Sections 9.0-9.3

- **Boolean Comparisons –** *Code yes or no questions*
- **Conditional Statements –** *Answer yes or no questions*
- **For Loops –** *Repeat yes or no questions*
- **Randomly Selecting –** *Select random data*
- **Appending Values –** *Add values to an array*

Comparison and Booleans

Comparison Operators

The result of a comparison expression is a `bool` value

```
x = 2          y = 3
```

Assignment statements

Comparison Operators

The result of a comparison expression is a `bool` value

```
x = 2                  y = 3
```

Assignment statements

```
x > 1                  x > y                  y >= 3

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

Comparison expressions

Comparison Operators

The result of a comparison expression is a `bool` value

```
x = 2                y = 3
```

Assignment statements

```
x > 1                x > y                y >= 3

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

Comparison expressions

Aggregating Comparisons

Summing an array or list of bool values will count the True values only.

```
sum([True, False, True])  == 2
```

Aggregating Comparisons

Summing an array or list of bool values will count the True values only.

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

Aggregating Comparisons

Summing an array or list of bool values will count the True values only.

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

1    + 0     + 1            == 2
True + False + True         == 2
```

(Demo)

Control Statements

Control Statements

These statements *control* the sequence of computations that are performed in a program

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

If and elif

```
if <conditional>:
    <if body>
elif < conditional >:
    <elif body 0>
elif < conditional >:
    <elif body 1>
...
else:
    <else body>
```

If statement has:

- conditional
- body

elif statement has:

- conditional
- body

(Demo)

Random Selection

Random Selection

`np.random.choice`

- Selects uniformly at random
- with replacement
- from an array,
- a specified number of times

```
np.random.choice(some_array, sample_size)
```

(Demo)

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`
    - `array_2` elements must have the same type as `array_1` elements

(Demo)

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

(Demo)