COMS BC1016
Introduction to Computational Thinking and Data Science

Python Review

Oct 13, 2025

Upcoming Schedule

Today

Bring a pencil/pen

Date	Topic	Lab	Assignment
10/13	Programming/Python Review		
10/15	Midterm Review	*No Lab*	HW4 Due
10/20	**Midterm Exam**		
10/22	Special Topics - Bias in AI	*No Lab*	

Homework Grades

- HW grades should now be visible
- There’s a small set of students who do not currently have grades for their first couple HW
    - We are aware and working on getting grades out ASAP

Regrades

- Full policy is in the syllabus
- TL;DR:
  - If you notice a mistake in the grading, please email your lab TA and me to have us look over it
  - Regrades must be requested within a week of when your grade was released

Lecture Outline

- Midterm Info
- Python Review
  - Data types
  - Functions and control statements
  - Visualizations

Midterm Info

Midterm Logistics

- Next week Monday (10/20) during class
- Paper exam
  - **Closed book, no electronic devices**
  - Allowed to bring a **single 5”x7” notecard**
  - **Bring a pen or pencil**
- TAs will lead a review session on Wednesday with sample questions

What you are *not* expected to know

- Exact syntax of `datascience` module functions
  - You should know what the functions do, but you will not be tested on memorizing the exact order of inputs
- Lectures 1-2

What you are expected to know

- Programming concepts:
  - Table functions/methods we've used thus far
    - Operations we can perform on tables
    - Visualizations (plots, bar charts, histograms, …)
  - NumPy functions we've used thus far
  - Built-in Python
    - Data types, basic operations on and with data types
- Concepts and Definitions:
  - Topics covered in slides

Suggested materials to study

- Slides
- Textbook Chapters
- Demo code

Data Types Review

Numbers

- Integers: Whole numbers
  - e.g., `3, -10, 25`
- Floats: Anything with decimals
  - e.g., `3.1, -10.2, 2.0`
- Basic calculations
  - e.g., `+, -, *, /`

Strings

- Text in python! Starts and ends with either a single quote or a double quote:
  - `“a”`
  - `‘This is a sentence’`
  - `“This is another sentence. Wow!”`
- You can convert values to a string using `str(..)`
  - `str(5)` becomes `“5”`
- You can convert strings of numbers to numbers
  - `int(’12’), float(‘1.2’)`

Booleans

- Booleans are data types for truth values: **`True`** or **`False`**
  - **`True`** is equivalent to `1`
  - **`False`** is equivalent to `0`
- `bool(x)` turns `x` into a boolean
  - e.g., `bool(1)` evaluates to **`True`** and `bool(0)` evaluates to **`False`**

Arrays

- Arrays are a sequence of values
  - e.g., `["Mystery", "Abby", "Jinu", "Baby", "Romance"]` or `[1,2,3,5]`
- Arrays are zero-indexed
  - The first element is the 0th and the second is 1st
- Can perform component-wise arithmetic
  - Note this only works for numpy arrays but not built-in Python lists!

```
from datascience import *
onetwothree = make_array(1,2,3)
onetwothree * 2
```

```
array([2, 4, 6])
```

```
from datascience import *
onetwothree = make_array(1,2,3)
twothreefour = make_array(2,3,4)
onetwothree + twothreefour
```

```
array([3, 5, 7])
```

```
onetwothree = [1,2,3]
twothreefour = [2,3,4]
onetwothree + twothreefour
```

```
[1, 2, 3, 2, 3, 4]
```

Tables

A table is a way of representing data sets

- Each row is an individual
- Each column is an attribute of the individual

Name	Age	Weight	Coloring	Sex	Owner
Ruby	14	8	tuxedo	F	Alice
Gertrude	15	12	tuxedo	F	Alice
Hamby	8	16	tabby	M	Bob
Fig	3	7	tabby	F	Bob
Corina	6	10	tortie	F	Carol
Frito	2	8.5	tabby	M	Carol

Control Flow and Functions

Anatomy of a Function

Name, Parameters, Body, Return Statement

Example:

```
def convert_to_figs(weight):
    new_weight = (weight/7).round(1)
    return new_weight
```

Functions with Multiple Arguments/Parameters

Functions can take in multiple inputs

- Each argument is given a unique name and separated by commas
- Specifying default values for particular inputs to makes them optional

```
def convert_to_figs(weight, decimal_places=1):
    '''Divides the input by 7 (Figs weight) and then rounds to
    the given number of decimal places'''
    new_weight = (weight/7).round(decimal_places)
    return new_weight
```

Recall: `apply`

Use `apply` to call a function on each element in a column

```
def convert_to_figs(weight):
    new_weight = (weight/7).round(1)
    return new_weight
```

```
cat_tbl.apply(convert_to_figs, 'Weight')
```

Returns an array with convert_to_figs called on each element in the 'Weight' column

apply with Multiple Inputs

For functions with multiple inputs, **apply** can take multiple columns

```
def convert_to_figs(weight, decimal_places=1):
  new_weight = (weight/7).round(decimal_places)
  return new_weight
```

```
cat_tbl.apply(convert_to_figs, 'Weight', 'Precision')
```

Control Statements

- Two major types are `if` and `for`
  - `if` statements specify code that should be run conditioned on something being true
    - They can also specify if alternative code should be run otherwise
  - `for` loops allow executing code over each element in some sequence of items

`if` statements

```
if statement_1:
    first_code_block
elif statement_2:
    second_code_block
elif statement_3:
    third_code_block
else:
    fourth_code_block
```

Runs if statement_1 == True

Runs if statement_1 != True
AND statement_2 == True

statement_1 != True
AND statement_2 != True
AND statement_3 == True

nothing above == True

`for` Statements

- Executing a `for` runs code with each element in an iterable

variable name

array of values

```
for item in some_array:
    print(item)
```

code to evaluate in each iteration of the loop

Table Functions

Grouping by a Single Column

The `group` method aggregates all rows with the same value in column `c`

- `tbl.group(c)`
- `tbl.group(c, func)`

`group` can optionally apply `func` to grouped values, for example:

- `len`: count of grouped values (default)
- `list`: list of all grouped values
- `sum`: total of all grouped values

```
cat_tbl.group('Owner')
```

Owner	count
Alice	2
Bob	2
Carol	2

```
cat_tbl.group('Owner', np.average)
```

Owner	Name average	Age average	Weight average	Coloring average	Sex average
Alice		14.5	10		
Bob		5.5	11.5		
Carol		4	9.25		

Grouping by Multiple Columns

The `group` method can also aggregate all rows that *share the combination of values* from multiple columns

```
cat_tbl.group(['Owner','Sex'])
```

Owner	Sex	count
Alice	F	2
Bob	F	1
Bob	M	1
Carol	F	1
Carol	M	1

```
cat_tbl.group(['Sex','Coloring'], sum)
```

Sex	Coloring	Name sum	Age sum	Weight sum	Owner sum
F	tabby		3	7	
F	tortie		6	10	
F	tuxedo		29	20	
M	tabby		10	24.5	

Joining Two Tables

Sometimes data about the same individuals are in different tables

- `join` combines the two datasets together

- Entries that do not appear in both tables are not included in the new table

To combine entries from `table1` and `table2` based on columns `c1` and `c2`

- `table1.join(c1, table2, c2)`

Pivot Tables

```
tbl.pivot(col_var, row_var, values, collect)
```

- `values`: Table column to aggregate

- `collect`: Function to aggregate with

Either include **both** `values` and `collect` or **neither**

```
cat_tbl.pivot('Owner', 'Sex', 'Age', np.average)
```

Sex	Alice	Bob	Carol
F	14.5	3	6
M	0	8	2

Group vs Pivot

Group

- One combo of grouping variables **per row**
- **Any number** of grouping variables
- Aggregate values of **all other columns** in the table
- Missing combos are **absent**

Pivot

- One combo of grouping variables **per entry**
- **Two** grouping variables: columns and rows
- Aggregate values of **values column**
- Missing combos **= 0 (or empty string)**

```
cat_tbl.group(['Sex','Coloring'], np.average)
```

Sex	Coloring	Name average	Age average	Weight average	Owner average
F	tabby		3	7	
F	tortie		6	10	
F	tuxedo		14.5	10	
M	tabby		5	12.25	

```
cat_tbl.pivot('Sex', 'Coloring', 'Weight', np.average)
```

Coloring	F	M
tabby	7	12.25
tortie	10	0
tuxedo	10	0

Randomly Selecting from Arrays

To select uniformly at random from array `some_array`

- `np.random.choice(some_array)`

To select `n` number of random elements from array `some_array`

- `np.random.choice(some_array, n)`

Randomly Sampling Tables

Returns a table with `n` rows sampled *with replacement* from Table `tbl`

```
- tbl.sample(n)
```

Returns a new table with `n` rows sampled *without replacement* from `tbl`

```
- tbl.sample(n, with_replacement=False)
```

Visualizations

Charts Summary

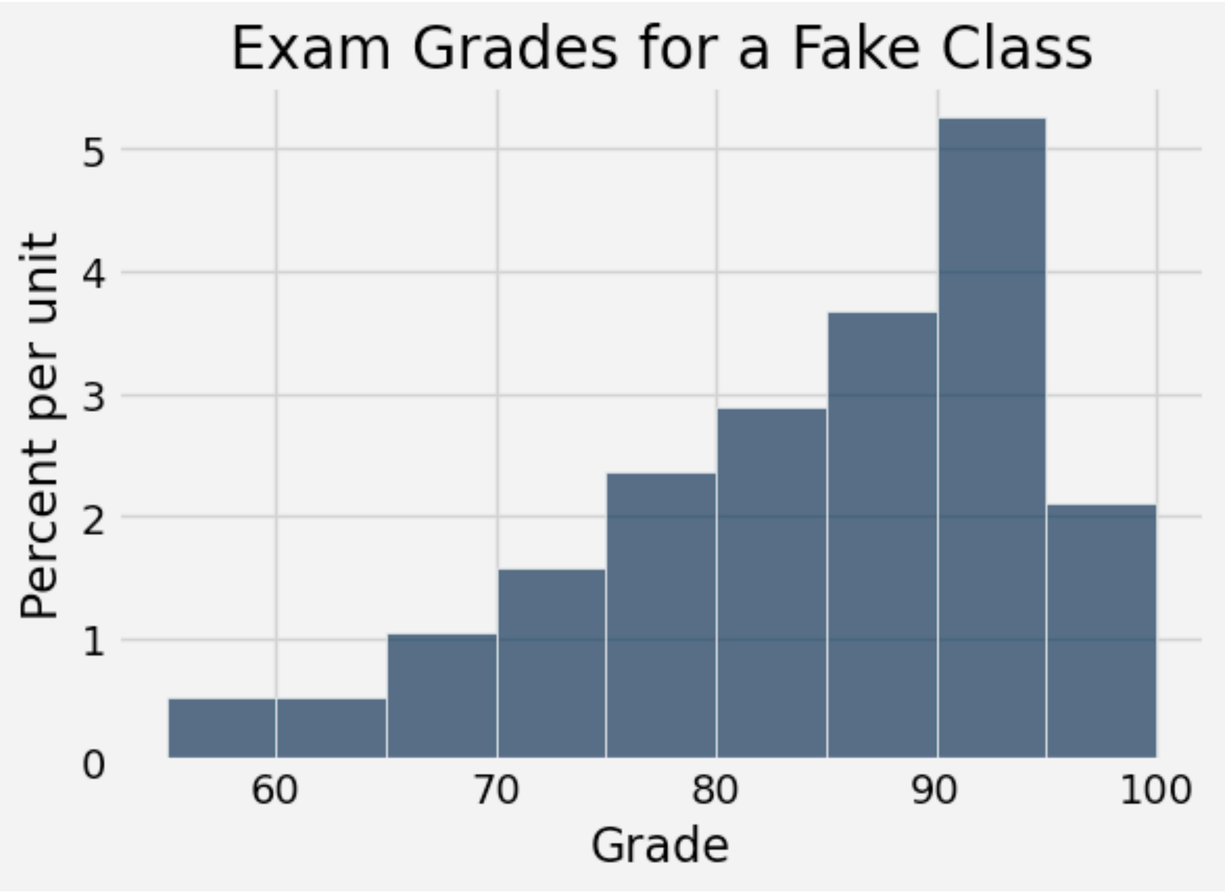
Type	Syntax	Description
Line graph	`.plot(x_axis, y_axis)`	Sequential data
Scatter Plot	`.scatter(x_axis, y_axis)`	Relation between two numerical values
Bar Chart	`.barh(column_label)`	Distribution of one categorical variable (already grouped)
Histogram	`.hist(column_label, unit, bins)`	Distribution of one numerical variable

Histograms

The area of each bar is a percentage of the whole

The horizontal axis is a numerical distribution - the bins don't need to be of equal size

The vertical axis is a rate (e.g., percent/year) - density



Bin size = 5

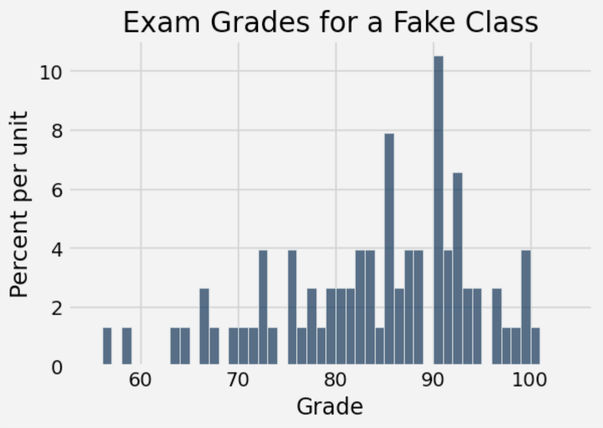
Area Principle

$$\text{area of bar} = \text{percent of entries in bin}$$

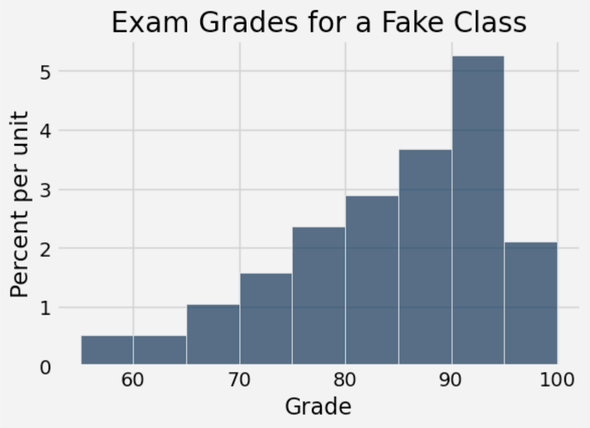
$$\text{area of bar} = (\text{height of bar}) \times (\text{width of bin})$$

$$\text{height of bar} = \frac{\text{area of bar}}{\text{width of bin}} = \frac{\text{percent of entries in bin}}{\text{width of bin}}$$

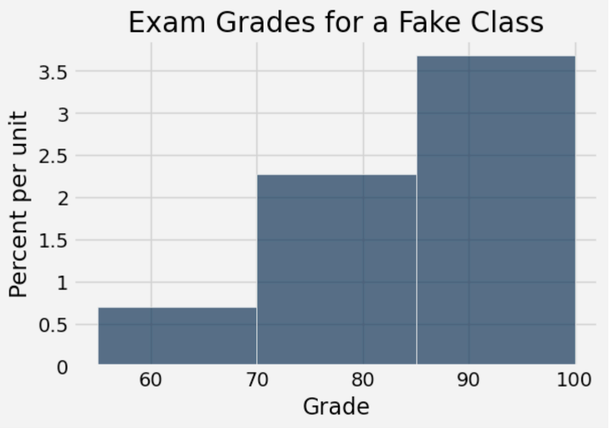
Bin size = 1



Bin size = 5



Bin size = 15



Bar Chart vs Histogram

Bar Chart

- Distribution of categorical variable
- Length of bars is proportional to the frequency / percent of individuals

Histogram

- Distribution of numerical variable
- Horizontal axis is numerical, bins can be unequal
- Area of bars is proportional of percent of individuals, height measures density

Upcoming Schedule

Date	Topic	Lab	Assignment
10/13	Programming/Python Review		
10/15	Midterm Review	*No Lab*	HW4 Due
10/20	**Midterm Exam**		
10/22	Special Topics - Bias in AI	*No Lab*	

Wed

Bring a pencil/pen