CS1070: Taming Big Data

Numpy, Pandas and DataFrames

Logistics

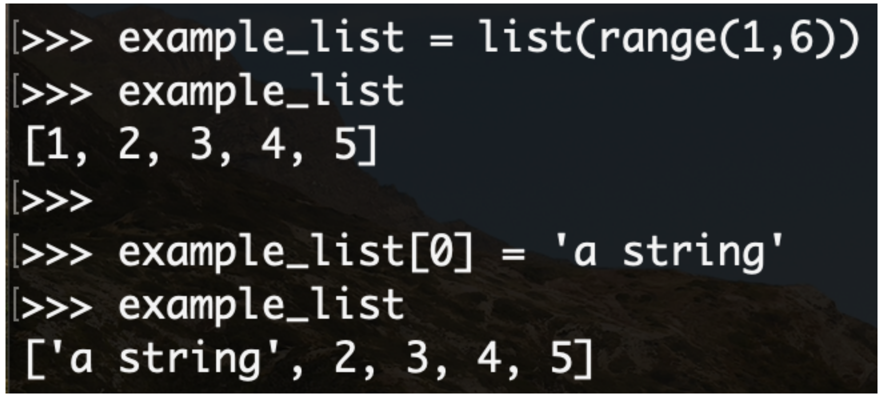
- Homework 1 due on Tuesday

Review

- TODO: Boolean logic review, control structure review

Lists vs. Dictionaries

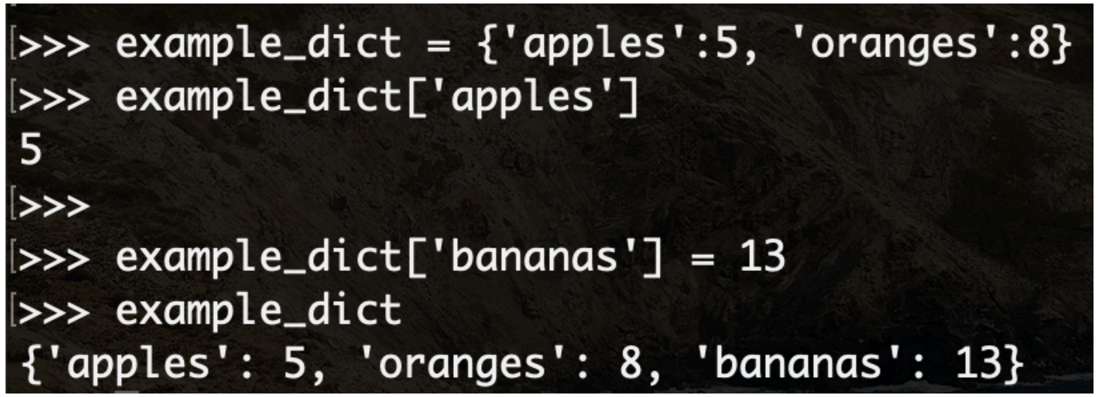
- A list is a **mutable**, **ordered** collection that can contain any type of data



```
[>>> example_list = list(range(1,6))
[>>> example_list
[1, 2, 3, 4, 5]
[>>>
[>>> example_list[0] = 'a string'
[>>> example_list
['a string', 2, 3, 4, 5]
```

Lists vs. Dictionaries

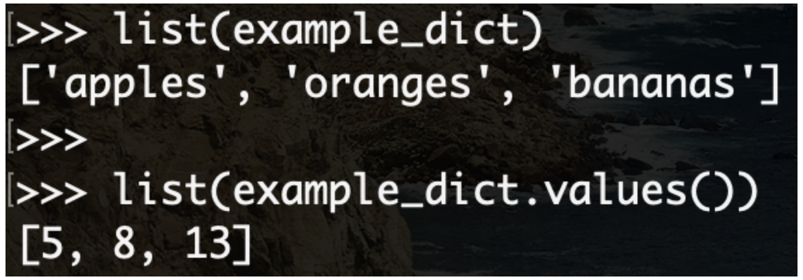
- A dictionary is an **unordered** collection of **key:value** pairs
- Instead of looking up a value by an index, you look them up by a **key**



```
>>> example_dict = {'apples':5, 'oranges':8}
>>> example_dict['apples']
5
>>>
>>> example_dict['bananas'] = 13
>>> example_dict
{'apples': 5, 'oranges': 8, 'bananas': 13}
```

Lists vs. Dictionaries

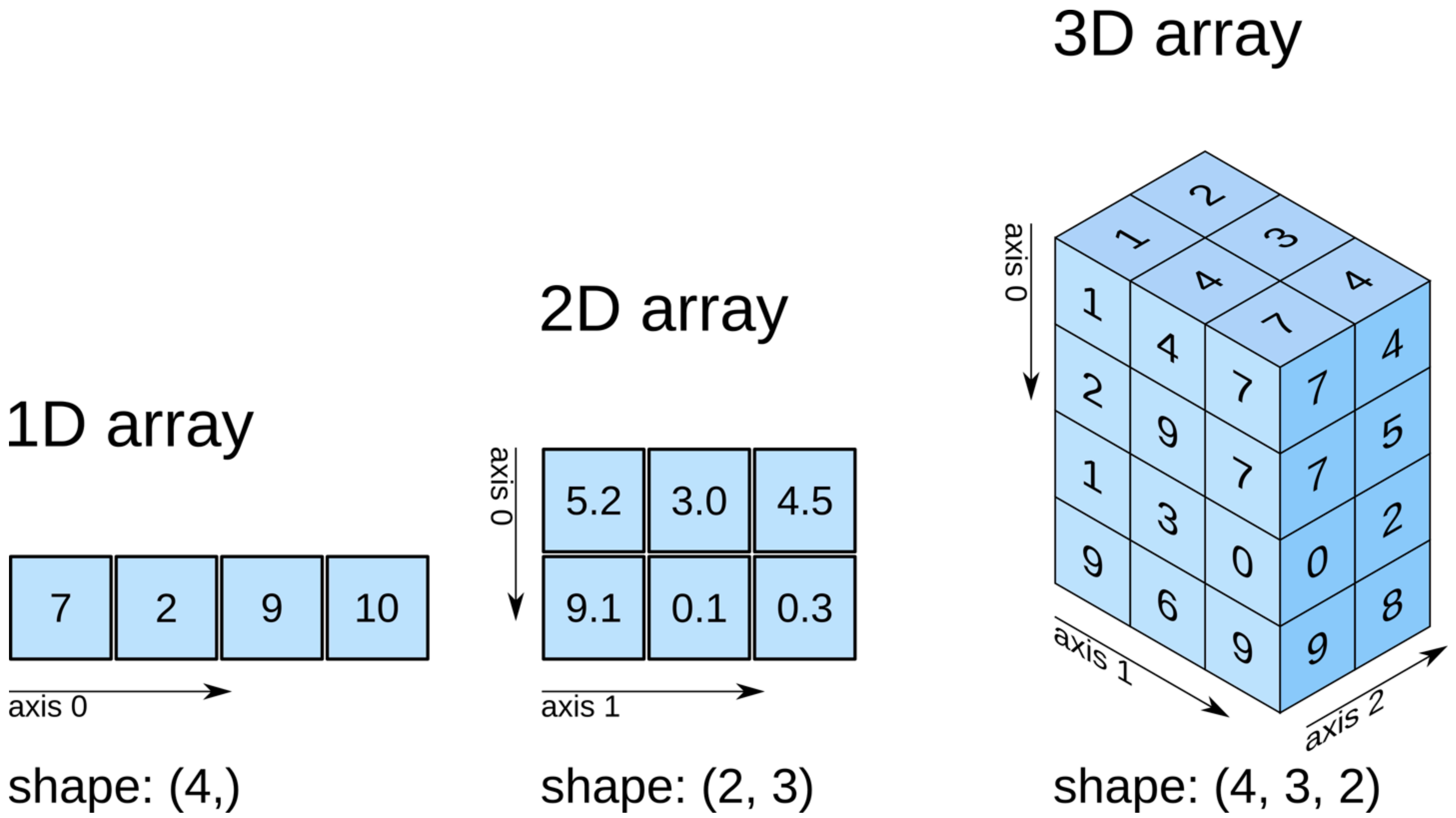
- A dictionary can be converted to a list
- list(dictionary_object): returns a list of keys
- list(dictionary_object.values())



```
>>> list(example_dict)
['apples', 'oranges', 'bananas']
>>>
>>> list(example_dict.values())
[5, 8, 13]
```

Numpy

- Core library for scientific computing
- Provides a really good multi-dimensional array object + tools for working with that array
- What's an array?
    - A grid of values
    - All of the same type
    - Number of dimensions is the *rank* of the array
    - The *shape* of an array is a tuple of integers giving the size of the array along each dimension


1D array
7
2
9
10
axis 0
shape: (4,)
2D array
axis 0
5.2
3.0
4.5
9.1
0.1
0.3
axis 1
shape: (2, 3)
3D array
axis 0
axis 1
axis 2
shape: (4, 3, 2)

Why not lists?

- Python lists can store the same information, but numpy is superior in terms of:
  - Size: take up less space in memory
  - Performance: implementation is faster than lists
  - Functionality: NumPy has built in, optimized functions such as linear algebra operations

Creating Numpy Arrays

```
my_list = [[1,2,3],[4,5,6]]

import numpy as np
my_array = np.array([[1,2,3],[4,5,6]])
```

Computing the Mean w/ Lists vs Arrays

```python
def mean_of_list(lst):
    row_mean = []
    for row in lst:
        row_mean.append(sum(row) / len(row))
    return sum(row_mean) / len(row_mean)

list_mean = mean_of_list(my_list)

array_mean = np.mean(my_array)
```

NumPy Documentation

- https://docs.scipy.org/doc/numpy/reference/
- https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf

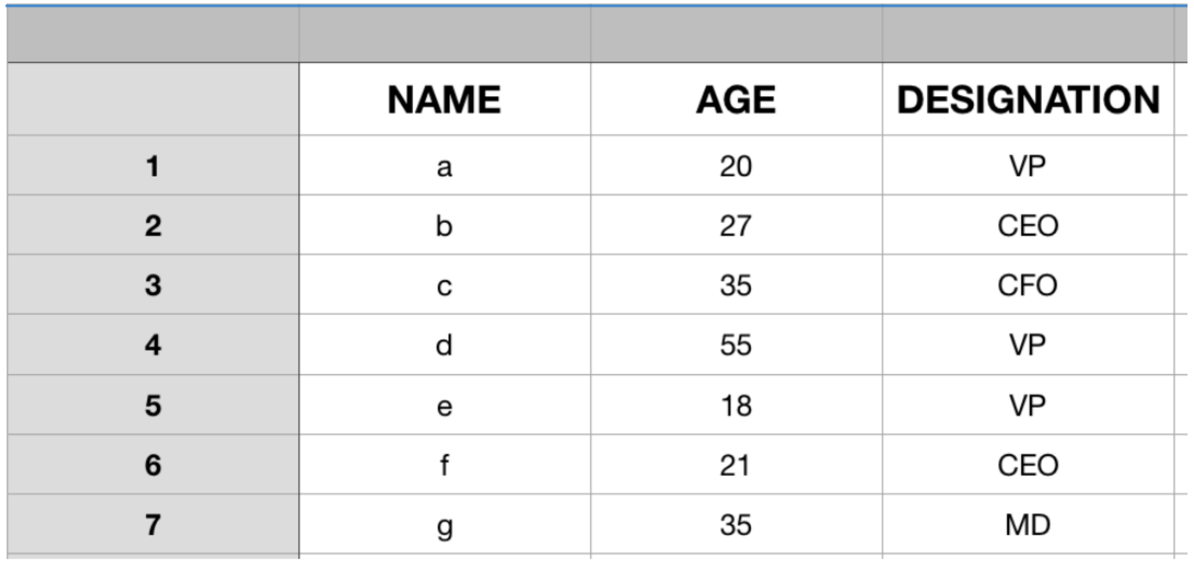
- The big things you'll need for this class:
  - How to create arrays
  - How to index into arrays
  - How to compute statistics of arrays (e.g., mean, median, standard deviation)

Pandas

- Pandas is an open source library that provides high-performance data structures and data analysis tools for Python
- Our text calls is ‘the pre-eminent data-wrangling tool in Python’
- What does that actually mean?
    - Excel spreadsheet / Google Sheets for Python

Pandas DataFrame

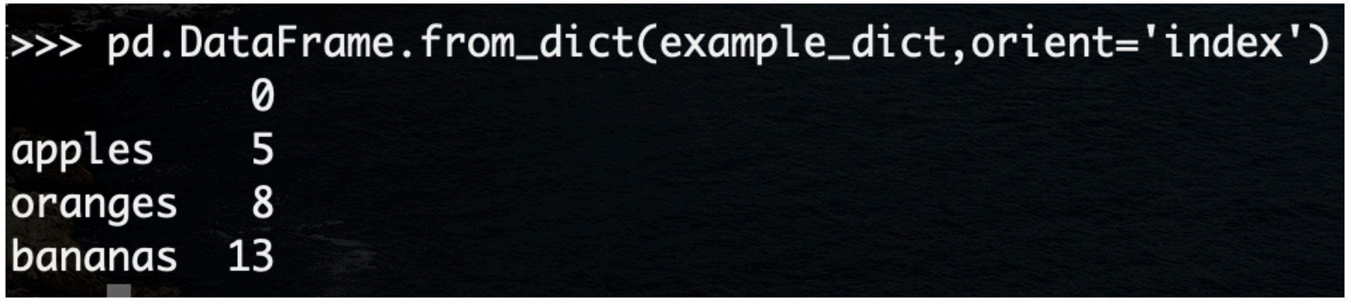
- 2D labeled data structure with rows and columns
- Each row represents one individual or sample
- Columns can be different types



	NAME	AGE	DESIGNATION
1	a	20	VP
2	b	27	CEO
3	c	35	CFO
4	d	55	VP
5	e	18	VP
6	f	21	CEO
7	g	35	MD

Pandas DataFrame

- Can be created from a dictionary, or from Excel spreahsheets, CSVs, MySQL databases, etc.



```
>>> pd.DataFrame.from_dict(example_dict,orient='index')
          0
apples    5
oranges   8
bananas  13
```

Pandas DataFrame

- Download ‘credit_card_data.xls’ from course website
- We are going to complete Exercise 2 from the text (page 14) together to see how to load an excel sheet into a pandas DataFrame

Different Types of Data Science Problems

- Data wrangling is one of the most significant things a data scientist does:
  - Figuring out how to get data and from where
  - Examining the data
  - Making sure the data is correct and complete
  - Joining it with other related data
- Once you have the data, you'll likely want to do ***predictive modeling***
  - Using a mathematical model to learn the relationships within the data in order to make accurate predictions when new data comes in

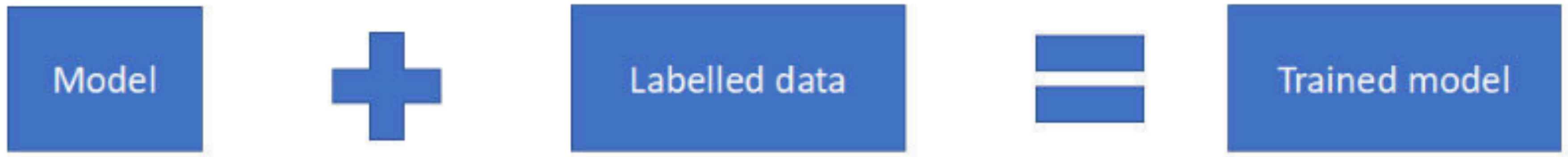
Predictive Modeling

- Data is organized in tabular format (as in a DataFrame), with independent and dependent variables
  - If you want to predict the price of a house based on features like the size of the house and number of bedrooms, the price of the house is the dependent variable, and the size and number of bedrooms are independent variables.
  - Independent variables – features
  - Dependent variables – response variable or target variable

Types of Predictive Modeling

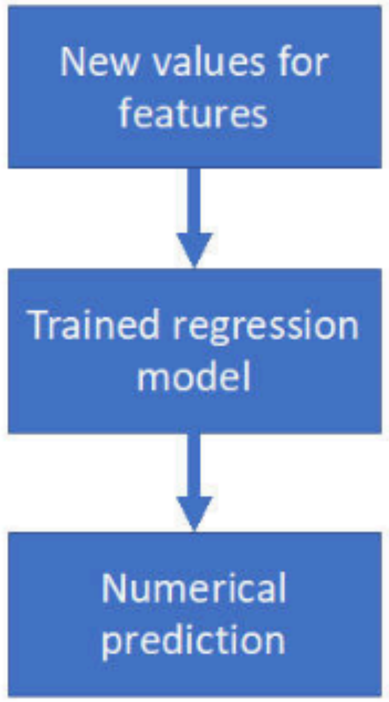
- Regression problem: predict the precise numerical value of the dependent variable on a continuous scale based on the independent variables that you observe
  - For example, given a house size and number of bedrooms, predict the exact dollar amount that house will sell for

- Classification problem: make a qualitative prediction of the dependent variable
  - For example, answer yes or no to the question “Will this house go on sale within the next five years?”



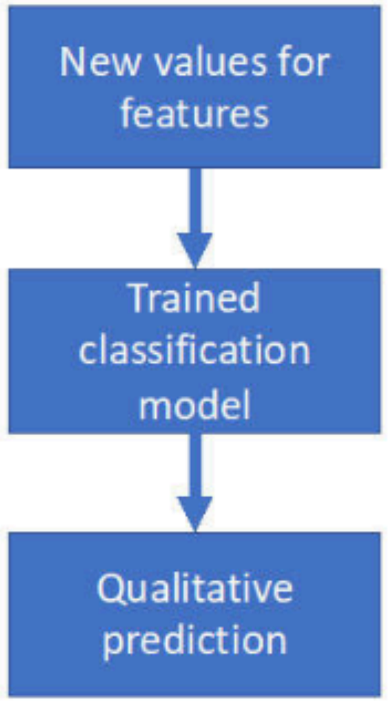
This could be a regression or a classification model.

Prediction phase: regression



For example, predict the price of a house. Hopefully get close to the actual price.

Prediction phase: classification



For example, predict the answer of a yes/no question. Hopefully get it right.

Supervised vs. Unsupervised Learning

- Right now we’re treating the “model training” as a black box
- Two types of model training (or “learning”):
  - Supervised
    - Relies on you having labels for your data
    - “Supervision” of the target variable by the known features
  - Unsupervised
    - More open questions that try to determine the underlying structure in a dataset that does not necessarily have labels

Pandas DataFrame

- Okay – going back to that credit_card_data that you downloaded and loaded into a DataFrame…
- Let’s image the following situation:
    - We are working for a credit card company
    - The have a dataset that includes some demographics and recent financial information for a sample of ~30k account holders
    - The data is organized at a credit account level – one row for each account
    - Rows are labeled by whether the account holder defaulted
- Task: develop a predictive model for whether an unlabeled account will default in the next month.

Ideas?

Step 1) Data Exploration

- Before we can get to any actual modeling, we need to make sure we understand the data that we have

- Useful questions to answer in data exploration:
  1. How many columns are there in the data?
  2. How many rows (samples)?
  3. What kind of features are there? Which are ***categorical*** and which are ***numerical***?
  4. What does the data in these features look like? (e.g., what are the statistics of it?)
  5. Is there any missing data?