* Course Overview
* Course Introduction
  + Data Science: set of fundamental principles that guide the extraction of knowledge of data
  + Kaggle: data science competition
* Target Audience
* Course Prerequisite
* Data Science Project Cycle O…
  + Extract data
  + Organize data
  + Analyze and create models
  + Present
* Why Python for Data Science
  + Easy and intuitive
  + Tools and libraries
  + Active community
  + Scalability and fast
  + Production python based application stack
* Course Outline
  + Set up environment
  + Python distributions
  + Jupyter notebook
  + Data science project template
  + Versioning
  + Extracting data
    - Databases
    - Apis
    - Web scraping
    - Titanic dataset
    - Database connectors
    - Requests
    - Beautiful soup
  + Basic exploratory data analysis
  + Numpy
  + Panda
  + Advanced exploratory data analysis
  + Data munging(identifying issues)
  + Feature engineering
  + Visualization
  + Matplotlib
  + Machine learning
  + Build and evaluate models
  + Kaggle submission
  + Scikit-learn
  + Model tuning
  + Model persistence
  + Machine learning API
  + Pickle library
  + Flask library
* Summary
* Introduction
* Overview
  + Python distributions
  + Jupyter notebook
* Python Distributions for Data ..
  + Option 1
    - Base python
    - Then install packages one by one
  + Option 2
    - Specialized python distributions
    - Comes with preinstalled and optimized python packages
  + Python distributions for Data Science
    - Anaconda
    - Enthought canopy
* Python 3.x vs Python 2.x
  + Python 3.x
    - Clean & faster
    - Future
  + Python 2.x
    - Stable third-party packages
    - Better community support
    - Backward compatibility
* Demo: Installing Anaconda D…
  + Download installer at anaconda.com
  + Open terminal
    - Type ‘python --version’
    - Type ‘pip list’ to see install packages
    - Type ‘conda list’ to packages in anaconda distribution
* Jupyter Notebook
  + Formerly know as IPython notebook
  + Combine code block, human-friendly text, images, videos in a single document
  + Run in web browsers
  + Support different kernels
  + Viewed with nbviewer(also in github)
  + Export to various formats such as pdf
* Demo: Setting up Jupyter Not..
  + Use terminal
    - Make folder
    - Navigate to folder
    - Type ‘jupyter notebook ‘ to launch jupyter note book server on your local machine
* Demo: Jupyter Notebook - Ba…
  + In terminal type ‘jupyter notebook’ to launch
  + In the browser it will open jupyter notebook
  + Click ‘new’
    - Then select the kernel
  + Jupyter notebook is made up of cells
    - These cells can contain different types of items
    - Will be treated as code
  + On each cell you can click run
    - Or ‘CTRL + Enter’ for windows
    - Or ‘Command + Enter’ for Mac
  + The number next to cell is the execution number of the cell
  + “Shift + Enter” will execute shell and create new cell below it
  + Markdowns
    - Use to create formatted text in web browsers
    - Click on ‘Cell’ -> ‘Cell Type’ -> ‘Markdown’
    - Or use the short cut ‘Esc + M’
    - The brackets next to the cell will be blank
    - Ex) # My First Notebook
  + “Esc + L” to show line numbers
  + ! to run shell commands in jupyter
    - !python --version
  + Will automatically save after a while
    - “CTRL + S” to manually save
* Demo: Jupyter Notebook -Ma..
  + Magic functions start with % for single line, %% for multiple lines
  + %matplotlib inline
    - Commonly used data visualization
  + %time x = range(10000)
  + %lsmagic
    - List magic functions avaliable
* Data Science Project Template
  + Structing your project in a consistent fashion
  + Consistency
  + Standardization
  + Collaboration
  + Reusability
  + Code Quality
  + Cookiecutter data science - Driven data
    - <https://github.com/drivendata/cookiecutter-data-science>
* Demo: Setting up Cookiecutte…
  + Install the template
    - Type ‘pip install cookiecutter’
    - Or if you have conda ‘conda install cookiecutter’
  + Use cookiecutter to start a new project
    - Type ‘cookiecutter <https://github.com/drivendata/cookiecutter-data-science>’
    - Give it a name
* Versioning for Data Science Pr..
  + Common repository
  + Track changes
  + Suitable for individuals too
  + Versioning Systems
    - Git
    - Github
    - Bitbucket
* Demo: Add Project To Git
  + ‘git init’: initialize empty git repository
  + ‘git add .’ take all files in folder
  + ‘git commit’: create check point
  + ‘git log’: to see information
* Summary
* Introduction
* Overview
  + Databases
  + Api
  + Web scraping
  + Libraries
    - Request
    - BeautifulSoup
  + Query languages
* Extracting Data from Databases
  + SQLite
  + MySQL
  + SQL Server
  + Steps
    - Import package
    - Connect top the database
    - Create the cursor
    - Execute query
    - Fetch results
    - Close connection
* Demo: Extracting Data from D…
  + Ex)
    - import sqlite3
    - #create the database if not avaliable
    - connection = sqlite3.connect(“classroomDB.db”)
    - connetion.close()
    - connection = sqlite3.connect(“classroomDB.db”)
    - cursor = connection.cursor()
    - create\_table = “””
    - CREATE TABLE classroom {
    - student\_id INTEGER PRIMARY KEY,
    - name VARCHAR(20),
    - ……
    - ); “”””
    - cursor.execute(create\_table)
    - connection.commit()
    - connection.close()
  + ex) insert data
    - classroom\_data = [ (1, “Raj”, “M”, 70, 84, 92),
    - (2, “Poonam”, “F”, 87, 69, 93),
    - …….
    - connection = sqlite3.connect(“classroomDB.db”)
    - cursor = connection.cursor()
    - for student in classroom\_data:
    - insert\_statement = “”” INSERT INTO classroom
    - (student\_id, name, gender, physics\_marks, chemistry\_marks, mathematics\_marks) VALUES ({0}, “{1}”, “{2}”, {3}, {4}, {5}); “””.format(student[0], student[1], student[2], student[3], student[4], student[5])
    - cursor.execute(insert\_statement)
    - connection.commit()
    - connection.close()
  + extract data
    - connection = sqlite3.connect(“classroomDB.db”)
    - cursor = connection.cursor()
    - query = “ SELECT \* FROM classroom”
    - cursor.execute(query)
    - result = cursor.fetchall()
    - for row in result:
    - print(row)
    - connection.close()
  + MySQL example
    - !conda install -y -q pymsql
    - import pymsql
  + connect to database
    - cnx = {‘host’: ‘[server url]’,
    - ‘username’: ‘[username]’,
    - ‘password’: ‘[password]’,
    - ‘db’: ‘[name of database]’}
    - connection = pymysql.connect(cnx[‘host’], cnx[‘username’], cnx[‘password’], cnx[‘db’] )
    - connection.close()
* Extracting Data Through APIs
  + Application Programming Interfaces(API)
  + REST API
  + Common https verbs: GET, POST
  + Ex)
    - import requests
    - result = requests.get(url)
* Demo: Extracting Data Throug…
  + Ex)
    - import requests
    - url = <https://api.adata/gov...>
    - requests = requests.get(url)
    - result.status\_code
    - result.headers
    - result.text
    - result.json
* Extracting Data Using Web Sc…
  + Document Object Model(DOM)
* Demo: Web Scraping Using Re..
  + Ex)
    - import requests
    - from bs4 import BeautifulSoup
    - from Ipython.core.display import display, HTML
    - display(HTML(html\_string))
    - ps = BeautifulSoup(html\_string)
    - print(ps)
    - body = ps.find(name=”body”)
    - print(body.findAll(name=”p”))
    - for p in body.findAll(name=”p”):
    - print(p.text)
    - print(body.fin(name=”p”, atts={“id”:”author”}))
* Demo: Getting Titanic Dataset…
  + Login to kaggle
  + Go to titantic disaster page
  + Click on data
  + Download ‘test.csv’, ‘train.csv’
  + Can also download it through scripts
* Demo: Getting Titanic Dataset…
  + Send credentials
  + Authorization
  + Request file
  + Send file
  + Store credentials in file .env
    - Don’t add file to version control
  + Ex
    - !pip install python-dotenv
    - from dotenv import load\_dotenv, find\_dotenv
    - #find .env file
    - dotenv\_path = find\_dotenv()
    - #load entries in .env file to the environment variables
    - load\_dotenv(dotenv\_path)
    - import os
    - #get from environment variable
    - KAGGLE\_USERNAME = os.environ.get(“KAGGLE\_USERNAME”)
    - print(KAGGLE\_USERNAME)
    - import requests
    - from requests import session
    - import os
    - from dotenv import load\_dotenv, find\_dotenv
    - payload = {
    - ‘action’: ‘login’,
    - ‘username’: os.environ.get(“KAGGLE\_USERNAME”),
    - ‘password’: os.environ.get(“KAGGLE\_PASSWORD”)
    - }
    - url = “https://www.kaggle.com/c/titanic/download/train.csv’
    - with session() as c:
    - c.post(‘https://www.kaggle.com/account/login’, data=payload)
    - response = c.get(url)
    - print(response.text)
    - from requests import sessions
    - payload = { …}
    - def extract\_data(url, file\_path):
    - with session() as c:
    - c.post(‘https://www.kaggle.com/account/login’, data=payload)
    - with open(file\_path, ‘w’) as handle:
    - response = c.get(url, stream=True)
    - for block in response.iter\_content(1024):
    - handle.write(block)
    - train\_url = ‘https://www.kaggle.com/c/titanic/download/train.csv’
    - test\_url = ‘https://www.kaggle.com/c/titanic/download/test.csv’
    - raw\_data\_path = os.path.join(os.path.pardir, ‘data’, ‘raw’)
    - train\_data\_path = os.path.join(raw\_data\_path, ‘train.csv’)
    - test\_data\_path = os.path.join(raw\_data\_path, ‘test.csv’)
    - extract\_data(train\_url, train\_data\_path)
    - extract\_data(test\_url, test\_data\_path)
* Demo: Creating Reproducible ..
  + ex)
    - get\_raw\_data\_script\_file = os.join(os.path.pardir, ‘src’, ‘data’, ‘get\_raw\_data.py’)
* Public Datasets
  + Data.gov
  + Amazon aws public data set collections
  + Uci data sets
  + Repository: awesome-public-datasets
* Committing Changes to Git
* Summary
* Introduction
  + Data Munging: take care of missing values, outliers
* Overview
* Introduction to NumPy and Pa...
  + NumPy
    - Fundamental tool for scientific computing
    - High level mathematical functions
  + Pandas
    - Built on top of NumPy
    - Data structure and operations on tabular data(Pandas dataframe)\
  + Data visualizations using Matplotlib
* EDA: Basic Structure
  + Exploratory Data Analysis(EDA)
* Demo: Investigating Basic Str..
  + ex)
    - import pandas as pd
    - import numpy as np
    - import os
    - #set the path of the raw data
    - raw\_data\_path = os.path.join(os.path.pardir, ‘data’, ‘raw’)
    - train\_file\_path = os.path.join(raw\_data\_path, ‘train.csv’)
    - test\_file\_path = os.path.join(raw\_data\_path, ‘test.csv’)
    - #read the data with all default parameters
    - train\_df = pd.read\_csv(train\_file\_path, index\_col=’PassengerId’)
    - test\_df = pd.read\_csv(test\_file\_path, index\_col=’PassengerId’)
    - type(train\_df)
    - #.info() to get brief information about the dataframe
    - test\_df.info()
    - train\_df.info()
    - test\_df[‘Survived’] = -888 #adding survived with a default value
    - df = pd.concat((train\_df, test\_df))
    - df.info()
    - #use .head() to get top 5 rows
    - df.head()
    - df.head(10)
    - #use .tail() to get last 5 rows
    - df.tail()
* Demo: Selection, Indexing, an…
  + Ex)
    - #column selection using dot
    - df.Name
    - #colum selection using column name as a string
    - df[‘Name’]
    - #selecting multiple columns using a list of column name strings
    - df[[‘Name’, ‘Age’]]
    - #use loc for label based indexing
    - df.loc[5:10,]
    - df.loc[5:10, ‘Age’: ‘Pclass’]
    - df.loc[5:10, [‘Survived’, ‘Fare’, ‘Embarked’]]
    - #use iloc for position based indexing
    - df.iloc[5:10, 3:8]
    - #filter rows based on the condition
    - male\_passengers = df.loc[df.Sex == ‘male’, :]
    - #use & or | to build complex logic
    - male\_passengers\_first\_class = df.loc[((df.Sex == ‘male’) & (df.Pclass == 1)), :]
* EDA: Summary Statistics
  + #use .describe() to get statistics for all numeric columns
  + df.describe()
  + there are a bunch of numerical features
    - df.Fare.mean()
    - df.Fare.median()
    - etc
* Centrality Measure
  + One number to represent entire set of values
  + Number central to the data
* Centrality Measure: Mean
  + Average behavior
  + Problem: affected by extreme values
* Centrality Measure: Median
  + Middle value in the sorted list
* Spread Measure
  + How spread out values are from central value
  + Variability in the data
* Spread Measure: Range
  + Difference between maximum and minimum
  + Problem: Affected by extreme values
* Spread Measure: Percentiles a…
  + X percentile is y means x% of values are below y
  + Ex)
    - 50 percentile of 10 means 50% of values are below 10
  + Quartiles: 25th, 50th, 75th
    - Bucket 1: below 25th
    - Bucket 2: 25th - 50th
    - Bucket 3: 50th - 75th
    - Bucket 4: about 75th
  + Box-Whisker Plot
* Spread Measure: Variance an..
  + Measure of variability
  + How far each value in list is from mean value
  + Small variance = less spread
  + High variance = large spread
  + Variance = (sum((value-mean)^2))/count
  + Affected by extreme values
  + Unit is not clear
  + Standard Deviation
    - Square root of variance
    - Unit is same as that of the feature
    - Low standard deviation = less spread
    - High standard deviation = large spread
* Demo: Getting Summary Stati..
  + Ex)
    - #use .describe() to get statistics for all numeric columns
    - df.describe()
    - #centrality meausres
    - df.Fare.mean()
    - df.Fare.median()
    - #dispersion measures
    - df.Fare.min()
    - df.Fare.max()
    - df.Fare.quantile(.25)
    - #variance
    - df.Fare.var()
    - #standard deviation
    - df.Fare.std()
    - #box-whisker plot
    - %matplotlib inline
    - df.Fare.plot(kind=’box’)
* Counts and Proportions
* Demo: Summary Statistics for
  + Ex)
    - #use .describe(include=’all’) to get statistics for all columns including non-numeric ones
    - df.describe(include=’all’)
    - #categorical column : Counts
    - df.Sex.value\_counts()
    - #categorical column : Proportions
    - df.Sex.value\_counts(normalize=True)
    - #apply on other columns
    - df[df.Survived != -888].Survived.value\_counts()
    - df.Pclass.value\_counts()
    - #visualize counts
    - df.Pclass.value\_counts().plot(kind=’bar’)
* Summary
* Introduction