\*La ingeniería de datos se preocupan principalmente por implementar los ***pipelines*** que permiten automatizar la obtención de datos y su posterior limpieza para que otros profesionales de los datos(científicos de datos o expertos en machine learning) puedan realizar su labor. Son la primera parte de la cadena.

\*¿Qué es la Ciencia e Ingieniería de Datos?

La Ciencia de Datos es la disciplina que se encarga de extraer conocimiento de los datos disponible. Casi siempre cuando te realizas una pregunta sobre datos estas fuentes se encuentra escondidas, ocultas o de difícil acceso. A nuestro alrededor hay datos en tu computadora, mesa, reloj, etc.

Los datos están por todas partes.

La Ciencia de datos es multidisciplinaria. A diferencia de muchos otros ámbitos profesionales dentro del mundo de la tecnología cuando hablamos de un científico de datos es una persona que sabe de matemáticas, ingeniería de software y sabe de negocios.

Se apoya en la Computer science, Matemáticas(Regresiones e Inferencias),

* También se auxilia de:
* Bases de Datos
* Análisis de texto y procesamiento de lenguaje natural
* Análisis de redes
* Visualización de datos
* Machine learning e Inteligencia Artificial
* Análisis de señales digitales
* Análisis de datos en la nube(Big Data)

\*Roles

* Existen por lo menos tres diferentes roles para tener un pipeline completo de ciencia de datos. Este curso trata sobre el primer rol:
* Data engineer: Se encarga de obtener los datos, Limpiarlos y estructurarlos para posterior análisis, crear pipelines de análisis automatizado, utilización de herramientas en la nube, análisis descriptivo de los datos.
* Data scientist: Una vez tiene los datos se encarga de generar el análisis matemático de ellos, encontrar las relaciones entre las variables, las correlaciones, las causas y por último genera los modelos predictivos y prescriptivos.
* Machine Learning engineer: Se encarga de llevar las predicciones a escala, de subirlos a la nube y allí generar muchas predicciones. Se encarga de mantener la calidad del modelo.

\*Configuración del ambiente

* Anaconda es una instalación de Python que ya trae preinstalado todos los paquetes necesarios para tu labor en la Ciencia de Datos, tiene más de 1400 paquetes. Nos permite configurar ambientes virtuales para poder utilizar diferentes versiones de nuestros paquetes.
* conda --version -> para conocer la versión y saber que lo tenemos instalado  
  conda –help -> nos da todos los comandos que podemos usar.  
  conda list  -> nos lista todos los paquetes que Anaconda instaló.
* Una buena práctica es generar un ambiente virtual por cada proyecto, los ambientes virtuales nos permiten generar varios proyectos con diferentes versiones de la librería sin generarnos errores de compatibilidad. Tradicionalmente en Python se utiliza virtualenv
* conda create --name [nombre-del-proyecto] [librerías-a-usar]
* conda create --name platzi\_data beautifulsoup4 requests numpy pandas matplotlib yaml
  + Beatifulsoup4: para parsear y manipular html.
  + Requests: para generar solicitudes a la web
  + Numpy: para hacer análisis numérico de nuestros datos
  + Pandas: para analizar los datos, modificarlos, transformarlos y generar análisis descriptivo sobre los mismos
  + Matplotlib: permite generar visualizaciones de nuestros datos
  + Yaml: permite generar algunas configuraciones
* Para activar -> conda activate platzi\_data  
  Para salir -> conda deactivate
* conda env list -> nos muestra los ambientes virtuales que tenemos  
  conda remove --name [nombre-del-proyecto] –all -> eliminar nuestro entorno virtual con todos nuestros paquetes, ej: conda remove –name platzi\_data –all

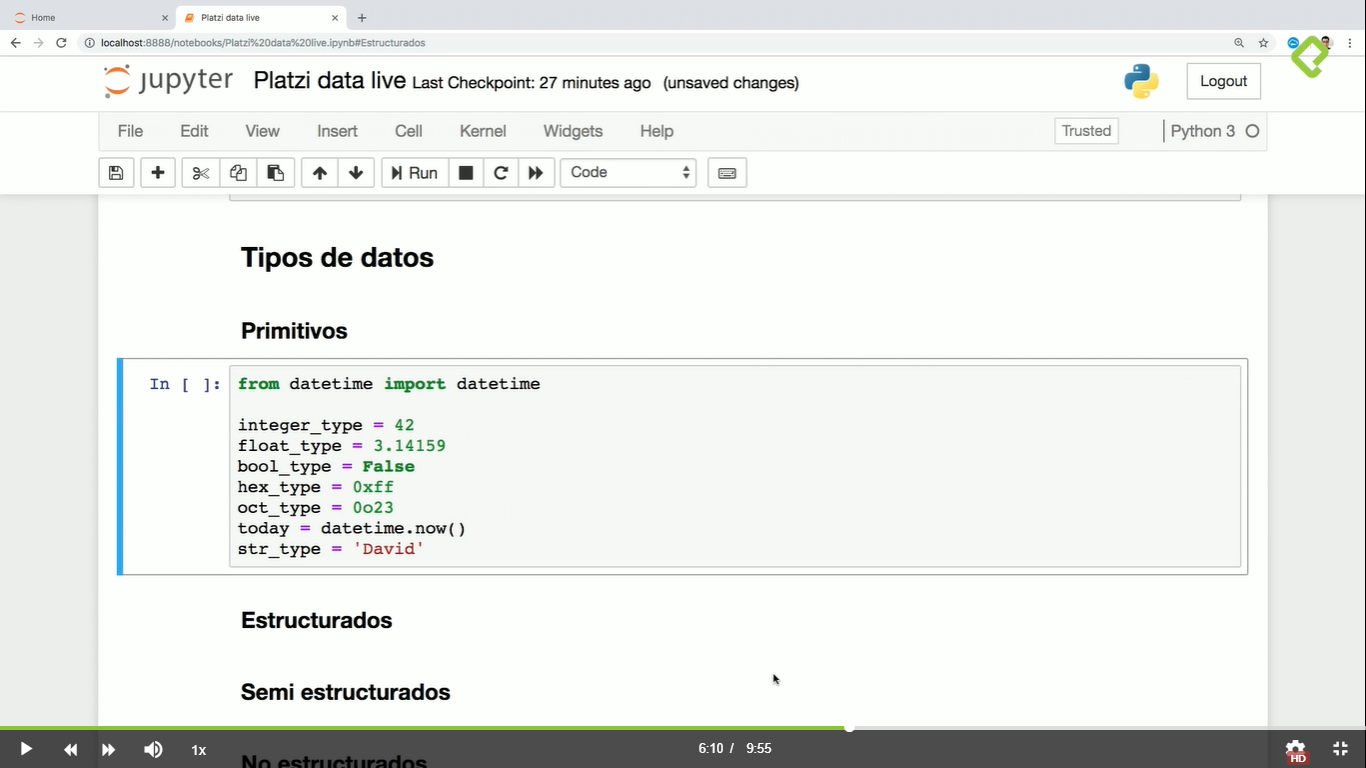
\*Jupyter Notebooks

* Algo interesante que tenemos con Anaconda es que nos trae Jupyter Notebooks.
* Jupyter Notebooks es un entorno de programación en el cual podemos mezclar ejecución de código en vivo, visualizaciones y añadir markdown.
* jupyter notebook para inicializar nuestro servidor de jupyter
* Jupyter Notebook tiene diferentes tipos de celdas en las cuales podemos escribir código o markdown. Si queremos ejecutar nuestro código hacemos ctrl + enter y si queremos ejecutar y añadir una nueva celda shift + enter.
* Jupyter Notebook tiene dos modalidades, la modalidad de edición y navegación.
* Notas:
  + Para crear una nueva línea presionamos b
  + Se puede dar Esc para salir de modo escribir y enter para entrar, y para ejecutar como se mencionó anteriormente
  + Presionando la letra m se crea un markdown
  + Presionando p salen todos los comandos de jupiter
  + Para eliminar una celda se presiona dd
  + Se puede copiar con c y pegar con v de una celda
  + Para limpiar todo lo ejecutado se entra en el menú cell/all output/clear
  + Truco response? -> para mostrar en una especie de consola información sobre lo que vamos a recibir
  + Truco response?? -> código del objeto response
  + Truco print(dir(response)) -> nos muestra todos los métodos que podemos utilizar con el objeto response, para usar los método podemos usar print(response.nombredelmetodo) si da 200 quiere decir que todo salió correctamente
  + Truco print(response.text) -> entrega el código html de la página

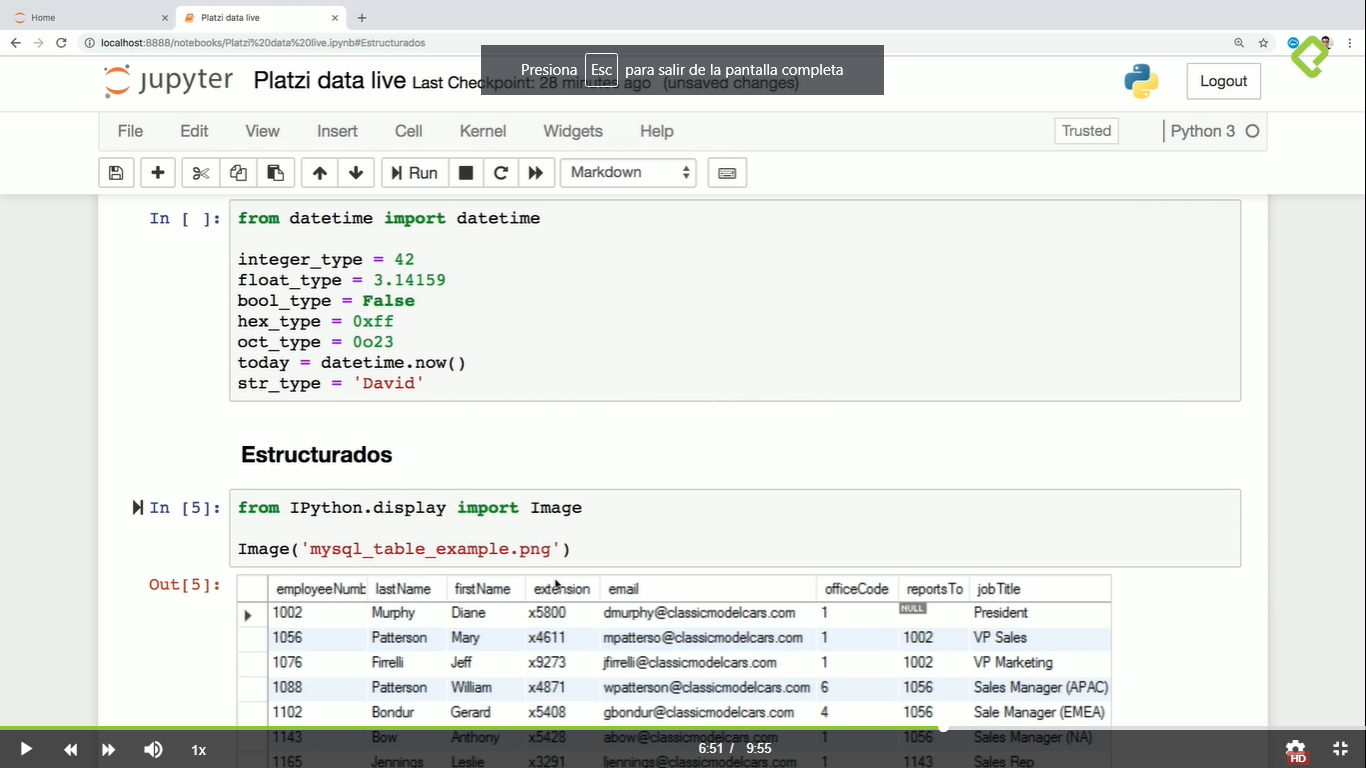
\*Profundizando en tipos de datos.

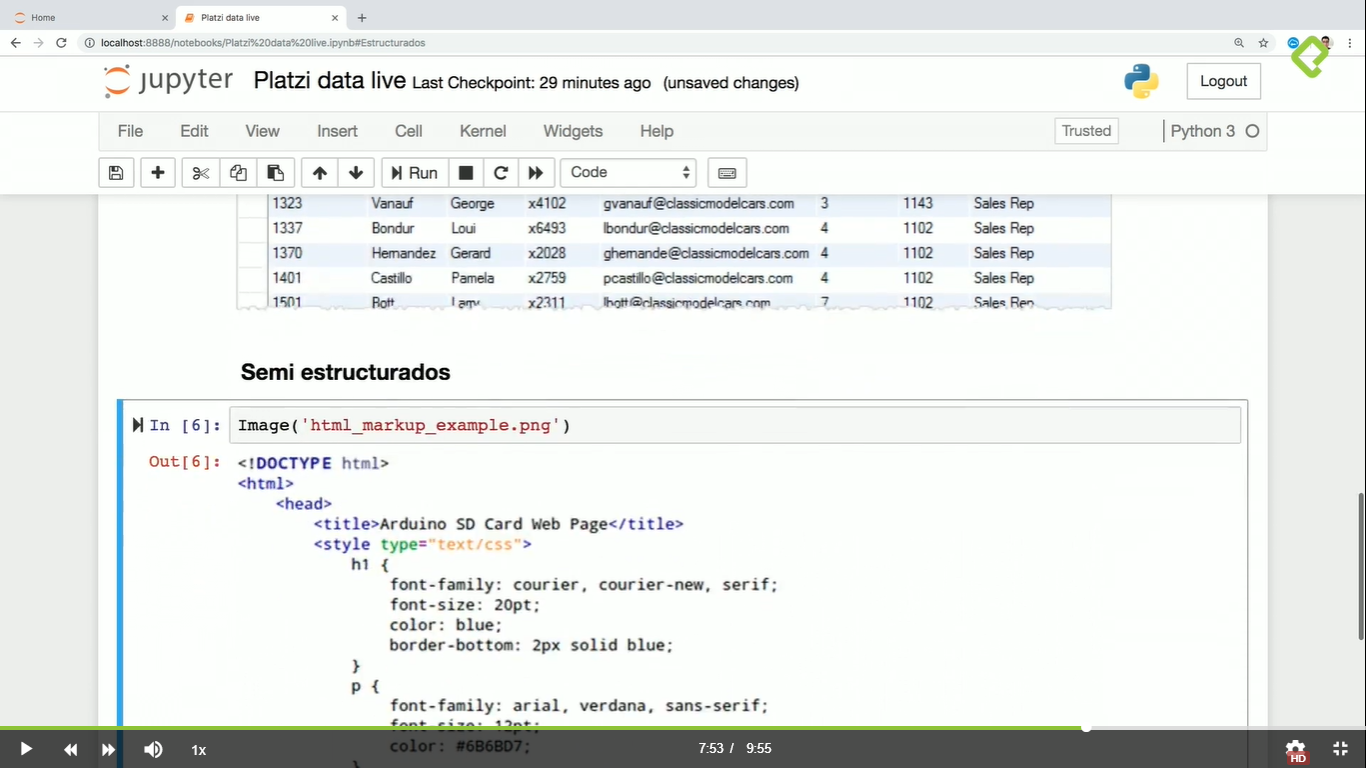
* Los datos vienen en mucha formas y estas formas las podemos clasificar de diferentes maneras, permitiéndonos poder aplicar técnicas distintas a cada uno de los tipos de datos.
  + Los primeros datos son los primitivos.

int, str, bool, float, hex, oct, datetime, objetos especiales

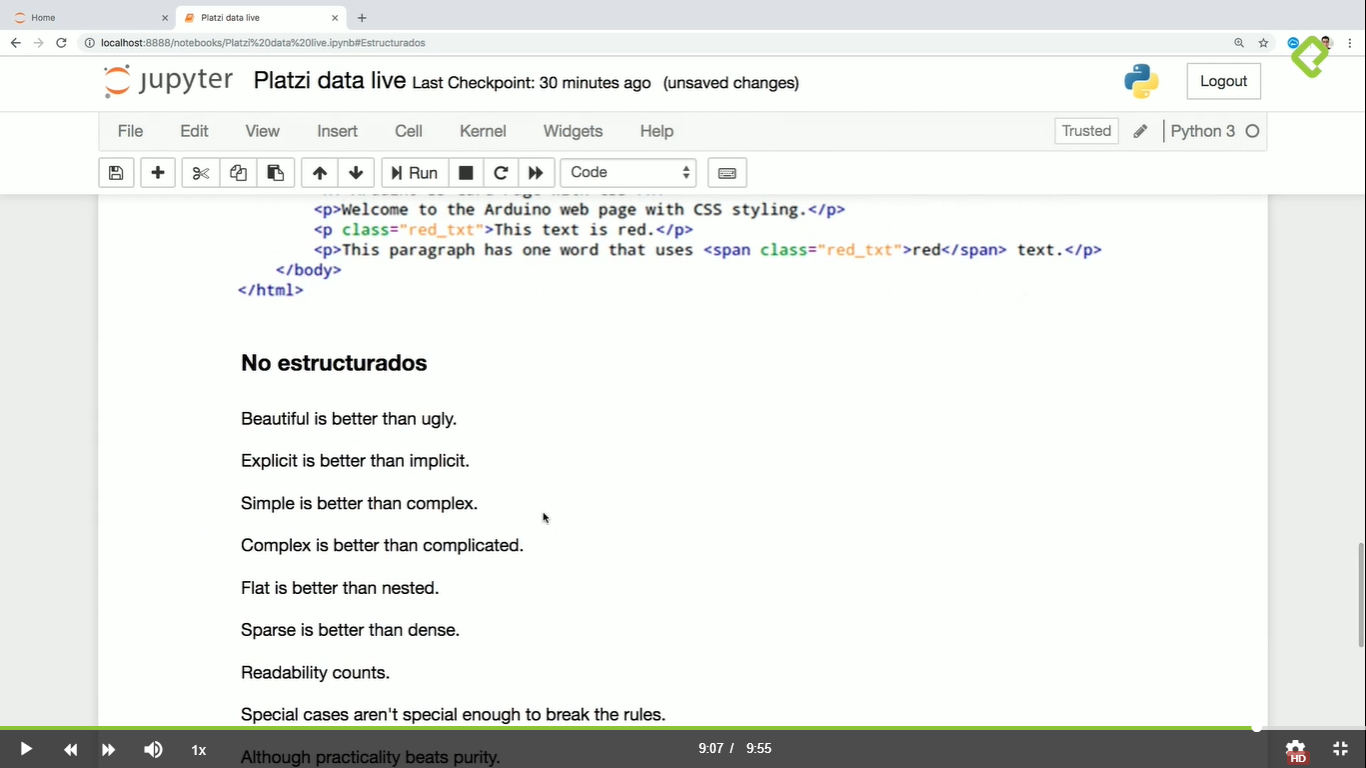


* + Tenemos clasificaciones ulteriores como los datos estructurados, semi estructurados y no estructurados.





* + Los estructurados son los más fáciles de acceder a su información, los semis estructurados donde podemos usar las APIs y los No estructurados son la mayoría de los datos que te vas a encontrar en tu desarrollo profesional.

-> texto

\*Fuentes de datos

* Web: Es una mina enorme con datos financieros, de startups, del clima, precipitación fluvial, astronómicos, de negocios, etc.
* APIs: Endpoints que viven en la web y nos devuelven JSON. Por ejemplo, la API de twitter, google, facebook.
* User Analytics: Son el comportamiento del usuario dentro de nuestra aplicaciones, algo similiar a los que nos ofrece Google Analytics.
* IoT: Se ha vuelto una mina espectacular en los últimos años. Como automóviles.
* Nota datasets: googledataset, data.world, kaggle

\*ETL

* ETL = Extract Transform Load
* Extract: Es el proceso de lectura de datos de diversas fuentes
  + Base de datos
  + CRM
  + Archivos CSV
  + Datasets públicos
* Transform: En este momento cuando nosotros tenemos que transformar los datos, tenemos que identificar datos faltantes o datos erróneos o una edad negativa. En esta etapa donde tenemos que identificar todos los problemas y solucionarlos.
  + Limpieza
  + Estructurado
  + Enriquecimiento.
* Load: Una vez transformados debemos insertarlos en el data warehouse

Depende del tipo de solución que se haya escogido

\*Introducción a las tecnologías web

* Las tecnologías web en principio podemos pensarlas como el internet, pero el internet es mucho más grande, es la red de redes, la forma en la que millones de computadores se conectan entre ellas para transferirse información.

El internet también se compone de otros pedazos como telefonía(voip), mail(pop3, imap), compartir archivos(ftp). El internet es una red que une varias redes públicas, privadas, académicas, de negocios, de gobiernos, etc.

La web específicamente es un espacio de información en el cual varios documentos(y otros recursos web) se pueden acceder a través de URLs y vínculos(links). La comunicación se da a través del protocolo HTTP.

Elementos básicos de la web:

* + HTML: nos da la estructura de la información. Es un lenguaje para anotar pedazos de información para que el navegador o otros tipos de programa puedan interpretar que tipo de información se encuentra ahí.
  + CSS: nos permite darle colores, arreglar el texto y añadir diferentes elementos de presentación.
  + Javascript: nos permite añadir interactividad y cómputo a nuestra web.
  + JSON: Simplemente es una forma de transmitir datos entre servidores y clientes. Es la forma estándar en las que en la web y las aplicaciones se comunican con los servidores backend.

\*¿Cómo trabajar con un documento HTML?

* En el caso de Python la librería estándar para manipular los documentos HTML se llama BeautifulSoup. (<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>)
  + BeautifulSoup nos ayuda a organizar gramaticalmente(parsear) el documento HTML para que tengamos una estructura con la cual podamos manejar y extraer información. BeautifulSoup convierte el string de HTML en un árbol de nodos para poder manipularlo.
  + Para manipularlo podemos usar los selectores CSS con soup.select()
* Para manipular BeautifulSoup importamos la librería
  + Import bs4

\*Implementando nuestro web scrapper: Configuración

* Crear un directorio y dentro del el:
  + Config.yaml
  + Common.py -> nos va a permitir cargar la configuración cuando iniciemos el software
  + Main.py

\*Implementando nuestro web scrapper: Obteniendo enlaces del front page

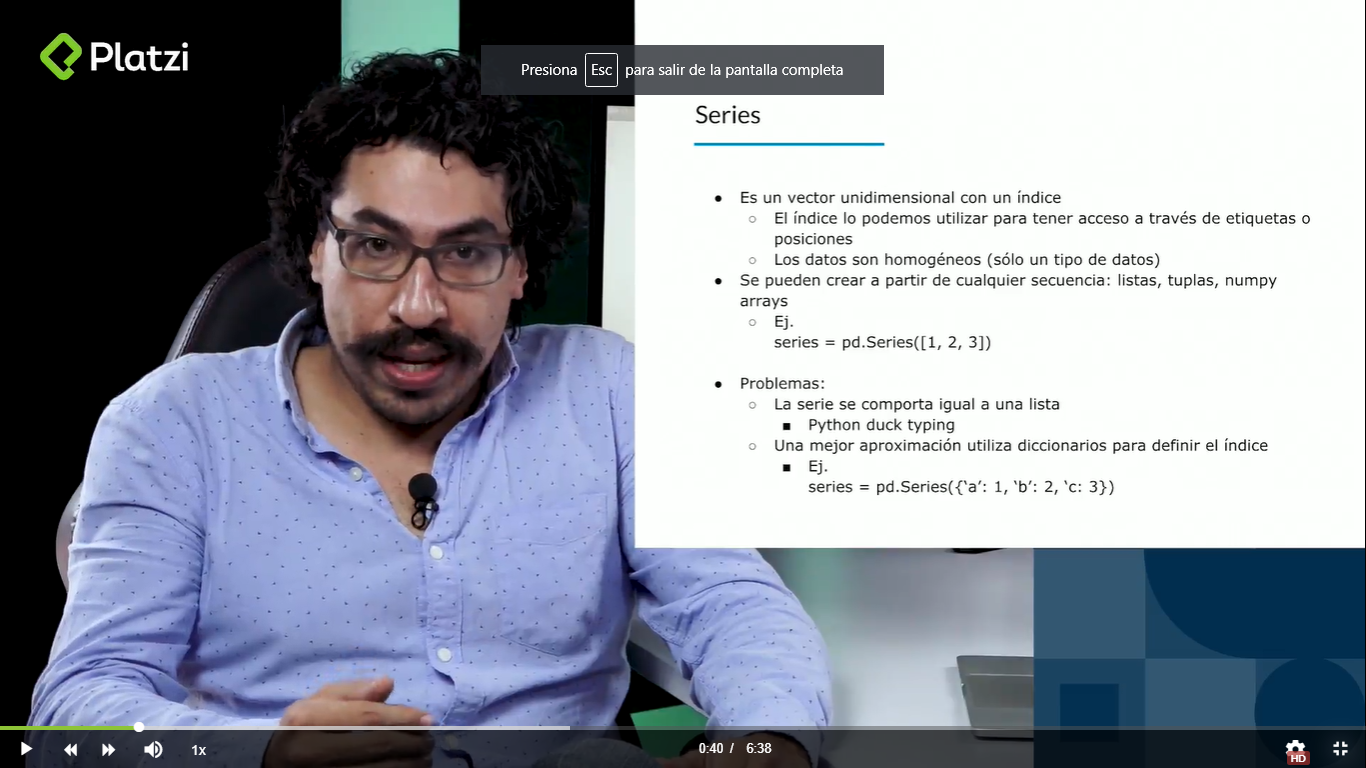
def \_visit(self, url):

response = requests.get(url)

response.raise\_for\_status() -> método que lanza un error si la solicitud no fue concluida

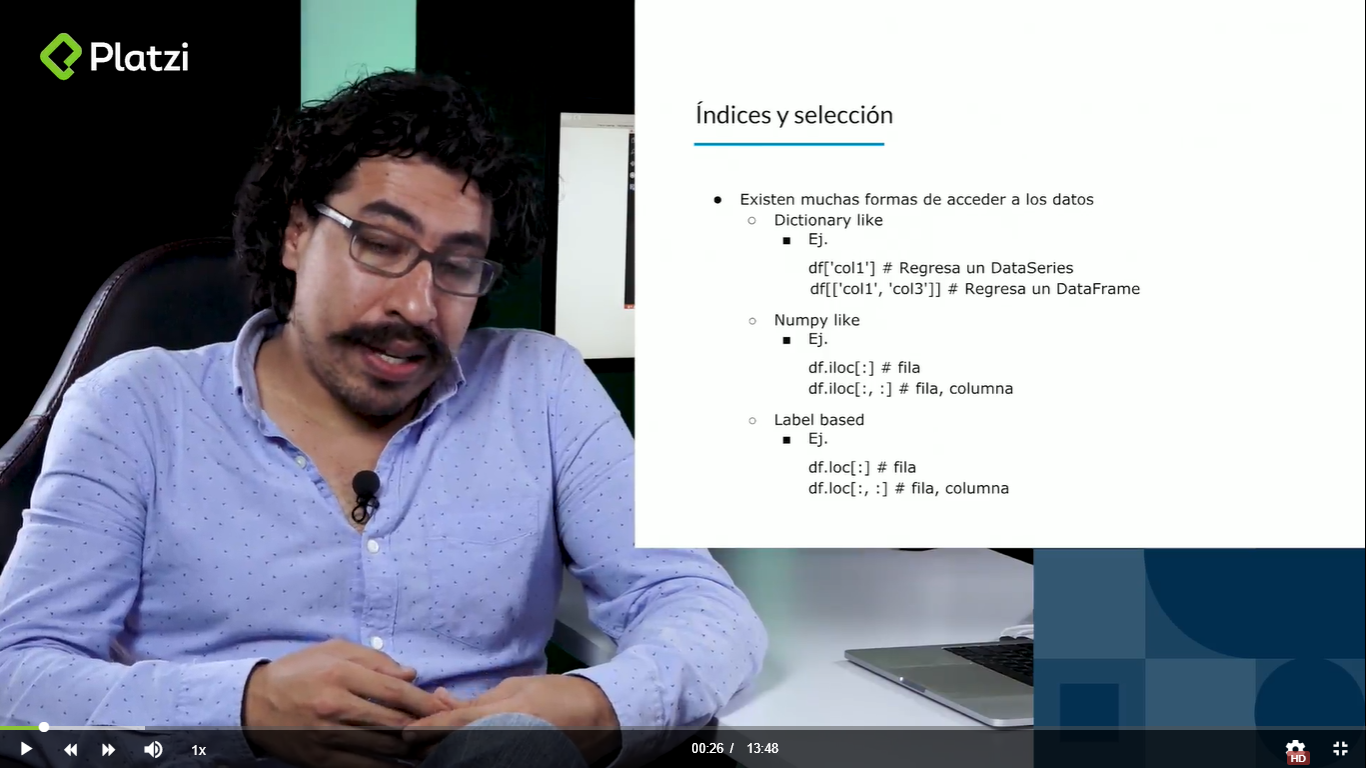
\*Introducción a Pandas

* Pandas nos otorga diversa facilidades para el ““domados de datos””. Nos otorga dos estructura de datos:
  + Series: Es un array unidimensional que representa una columna.

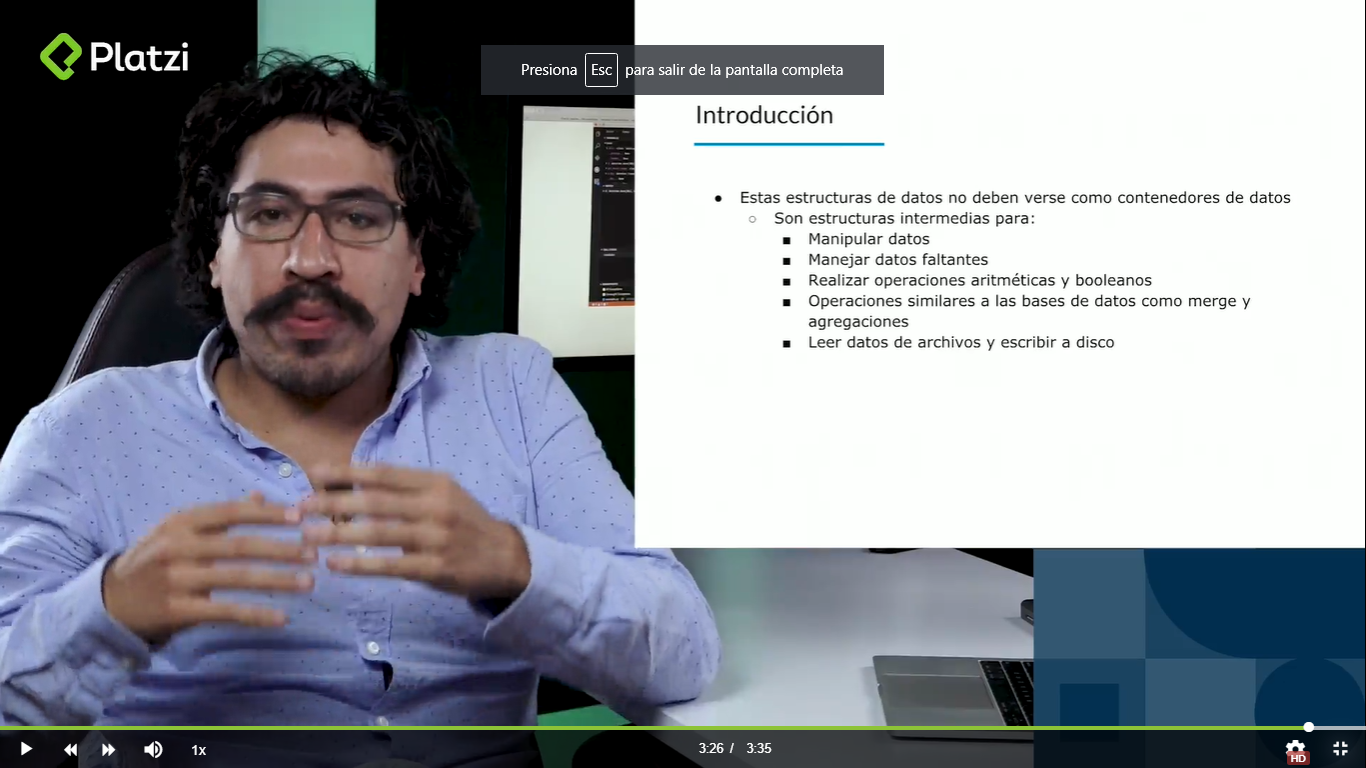


* + DataFrame: Es un conjunto de series que forman una tabla. Se pueden acceder a través de indices como un etiqueta(label) o pueden ser posicionales es decir 0 o indice 100. También pueden ser rangos o slices





Estas estructuras de datos no son contenedores de datos. En Pandas las utilizamos para transformar y enriquecer nuestros datos, manipularlos, manejar los faltantes, realizar operaciones aritméticas, combinar diferentes dataframes en uno solo para obtener una nueva tabla.



\*Data wrangling con Pandas (domado de datos)

Data wrangling es una de las actividades más importantes de todos los profesionales de datos. Simplemente es limpiar, transformar y enriquecer el dataset para objetivos posteriores.

Pandas es una de las herramientas más poderosas para realizar este ““domado”” de datos. Recordemos que Pandas trae muchas de sus abstracciones del lenguaje R, pero nos otorga lo mejor de ambos mundos, por eso es tan popular.

Nos permite:

* + generar transformaciones con gran facilidad.
  + trabajar rápidamente con datasets grandes
  + detectar y reemplazar faltantes
  + agrupar y resumir nuestros datos
  + visualizar nuestros resultados.

\*¿Cómo trabajar con datos faltantes?

Los datos faltantes representan un verdadero problema sobre todo cuando estamos realizando agregaciones. Imagina que tenemos datos faltantes y los llenamos con 0, pero eso haría que la distribución de datos se modificaría radicalmente. Podemos eliminar los registros, pero la fuerza de nuestras conclusiones se debilita.

Pandas nos otorga varias funcionalidades para identificarlas y para trabajar con ellas. Existe el concepto que se llama NaN, cuando existe un dato faltante simplemente se rellena con un NaN y en ese momento podemos preguntar cuáles son los datos faltantes con:

* + .isna() para preguntar cuales son los datos faltantes
  + .notna() para preguntar dónde hay datos completos.
  + .dropna() para eliminar el registro.

Para reemplazar:

* + .fillna() donde le damos un dato centinela
  + .ffill() donde utiliza el último valor.

\*Introducción a los sistemas de datos

Los sistemas de datos vienen en muchos sabores y colores, SQL, NoSQL, especializados en procesamiento en bloque, chorro y streaming. Este tipo de sistema nos permite realizar queries sofisticadas y compartir nuestro trabajo con otros miembros del equipo.

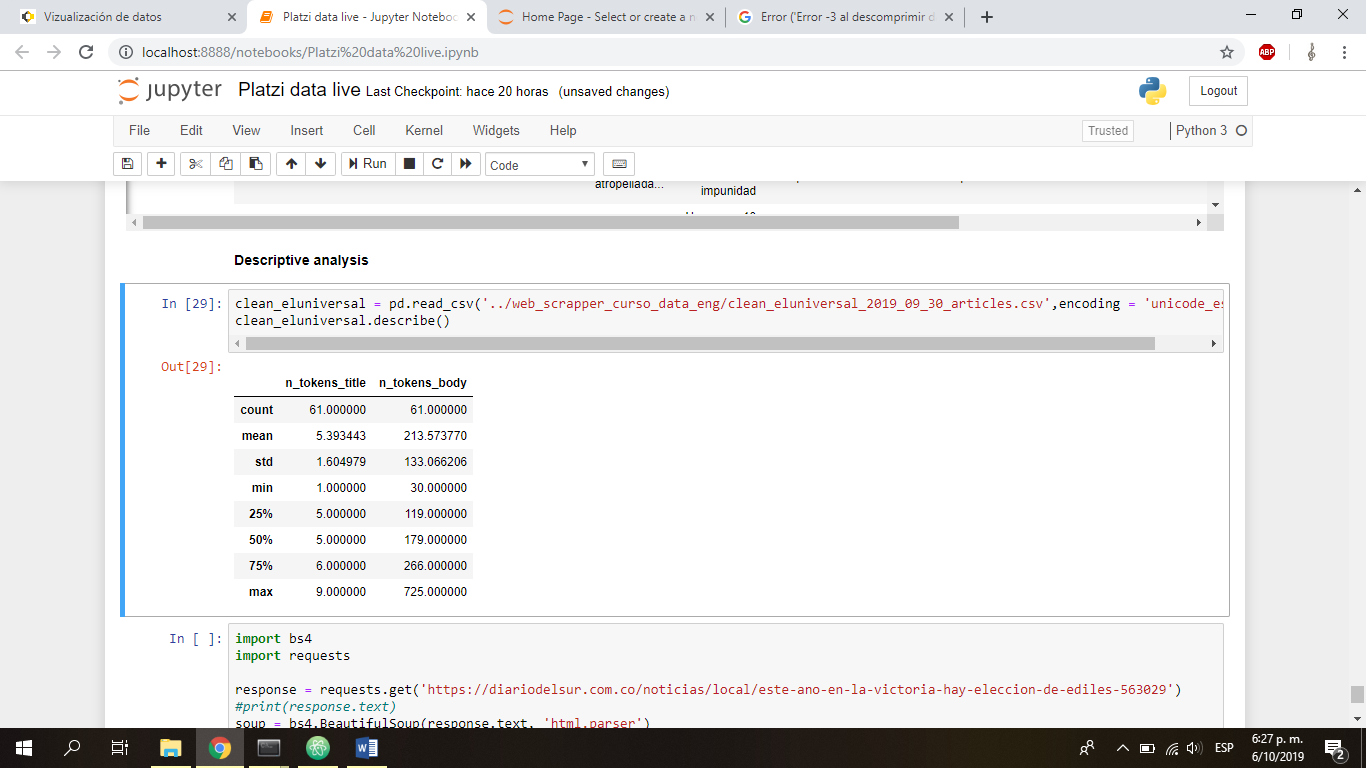
* + Procesamiento de bloque: Estamos hablando de datos históricos, qué sucedió ayer, en el trimestre pasado, cuáles fueron las ventas del año anterior o de los últimos cinco años. Nos permite realizar el procesamiento de manera eficiente.
  + Procesamiento en chorro: Significa que estamos procesando los datos conforme van llegando, las transformaciones se realizan en tiempo real, Este tipo de sistema nos sirven para cuando queremos realizar decisiones en donde la importancia del tiempo es fundamental.

El criterio principal a tener en cuenta: El tiempo que tienes. Si bien los sistemas open source son gratis, para poderlos implementar necesitas tener conocimientos de cloud, debes poder saber trabajar y mantener máquinas.

SQL vs NoSQL

* + La discusión más relevante en el mundo de las aplicaciones web y móvil, donde dependiendo de la aplicación, la decisión puede ser fundamental para el crecimiento de la app.
  + La verdad es que para los profesionales de los datos, especialmente los profesionales de los datos. Es necesario saber ambos.

\*\*\*Notas:

* Se trabaja con la librería re(expresiones regulares) para verificar las url que nos sirven y cuales no.
* Para no mostrar el error capturado con try except se usa: exc\_info=False
* Importamos datetime para guardar los archivos con la fecha y csv
* Read data -> pd.options.display.max\_rows = 10 muestra 10; el\_universal.head() -> muestra los primeros 5; el\_universal.tail() -> muestra los últimos 5
* Hashlib -> librería propia de Python generalmente para operaciones criptográficas pero se va a usar para tener el hash único de la url para poder trabajarlo
* Nltk -> librería para la ciencia de datos, nos permite tokenizar(separar la palabras del título y contarlas en el body), import nltk; from nltk.corpus import stopwords;
* Pd.describe() -> muestra datos estadísticos del dataset 

import requests

response = requests.get('http://www.platzi.com')

print(dir(response))

print(response.text)

import bs4

soup = bs4.BeautifulSoup(response.text, 'html.parser')

print(soup.title.text)

Platzi: Cursos online profesionales de tecnología

import bs4

soup = bs4.BeautifulSoup(response.text, 'html.parser')

print(soup.select('meta[name=description]'))

[<meta content="Aprende desde cero a crear el futuro de la web. Cursos de Desarrollo, Diseño, Marketing, y Negocios." name="description"/>]

import bs4

soup = bs4.BeautifulSoup(response.text, 'html.parser')

print(soup.select('meta[name=description]')[0]['content'])

Aprende desde cero a crear el futuro de la web. Cursos de Desarrollo, Diseño, Marketing, y Negocios.

import bs4

soup = bs4.BeautifulSoup(response.text, 'html.parser')

courses\_links = soup.select('.HomeCategories-items a')

courses = [course['href'] for course in courses\_links]

for course in courses:

print(course)

/categorias/desarrollo/

/categorias/negocios/

/categorias/crecimiento-profesional/

/categorias/produccion-audiovisual/

/categorias/marketing/

/categorias/diseno/

Ejemplo sacando temas del curso-platzi

import bs4

import requests

response = requests.get('https://platzi.com/cursos/scikit/')

soup = bs4.BeautifulSoup(response.text, 'html.parser')

themes\_links = soup.select('.Material-name')

themes = [course.text for course in themes\_links]

cont = 1

for theme in themes:

print(cont, theme)

cont += 1

Ejemplo de compañero que sacó título y descripción de noticias

import requests

import bs4

response = requests.get('https://elpais.com/tecnologia/')

soup = bs4.BeautifulSoup(response.text, 'html.parser')

data\_titles = soup.select('.articulo-titulo')

titles = [title.text for title in data\_titles]

data\_descriptions = soup.select('.articulo-entradilla')

descriptions = [description.text for description in data\_descriptions]

title\_description = {title: description for title, description in zip(titles, descriptions)}

for news in title\_description:

print('{} \n\t {}'.format(news, title\_description.get(news)))