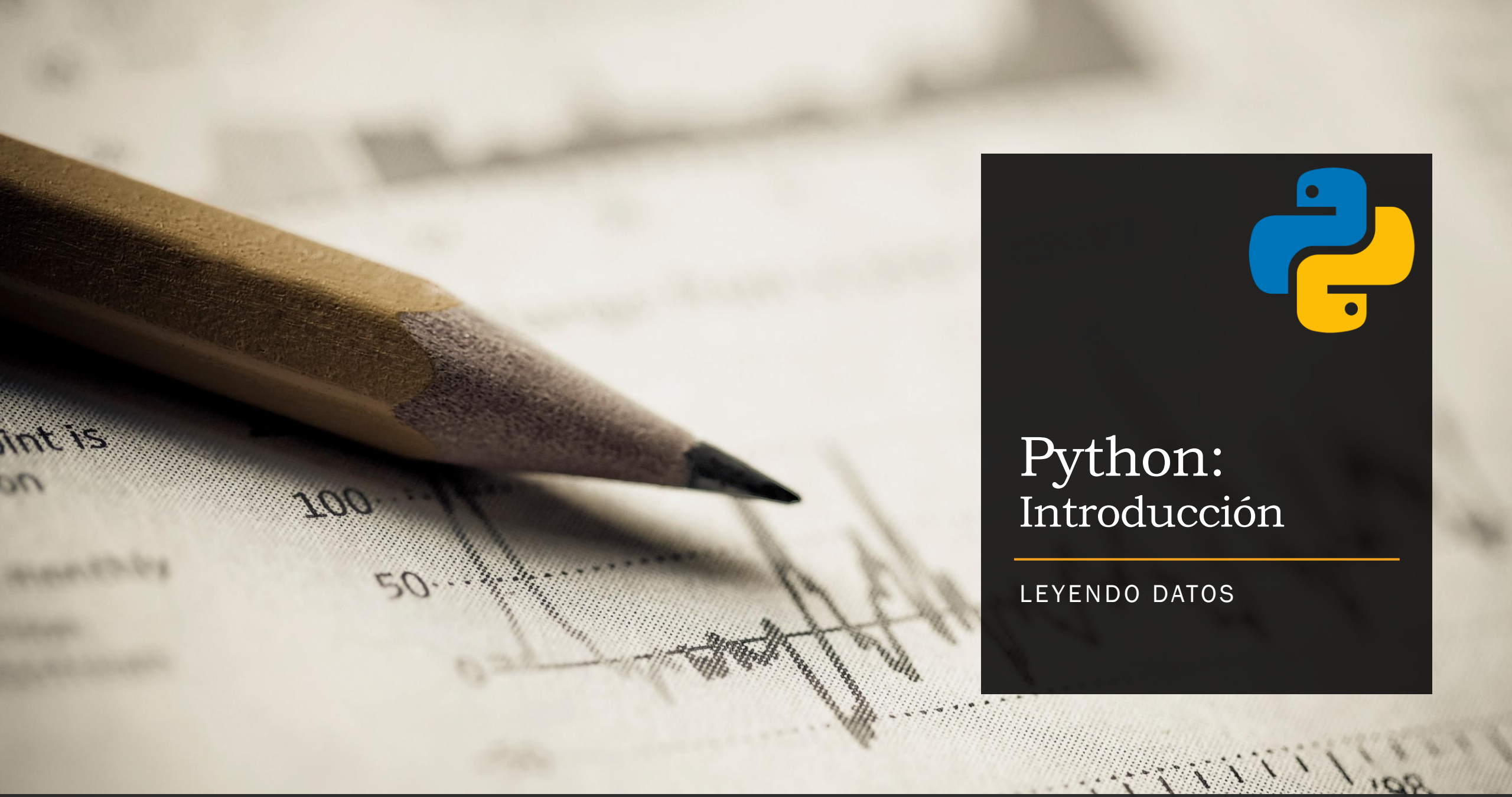


Python:
Introducción

LEYENDO DATOS

Esta clase

- Conceptos de datos y análisis
- Formas de almacenar datos

Datos

Los datos son cualquier clase de información que identifica conceptos, números o relaciones entre situaciones naturales del mundo que nos rodea.

Nombres de personas, estadísticas de rendimiento de un computador, información económica de un país, etc.

Datos

¿De dónde vienen los datos? ¡Todo lugar!

Ciencia: Bases de datos de astronomía, genética, datos ambientales, datos de transporte, etc.

Ciencias sociales: Libros escaneados, documentos históricos, datos de redes sociales, información de herramientas como el GPS.

Comercio: Reportes de ventas, transacciones del mercado de valores, tráfico de aerolíneas, etc.

Entretenimiento: Imágenes de internet, taquillas de Hollywood, datos de tráfico de Netflix, etc.

Medicina: Records de pacientes, resultados de estudios clínicos, etc.

Ejemplos

Colisionador de Hadrones en Suiza

Investigación de la naturaleza de las partículas

Durante sus experimentos captura más

de **40 terabytes de datos por segundo**!

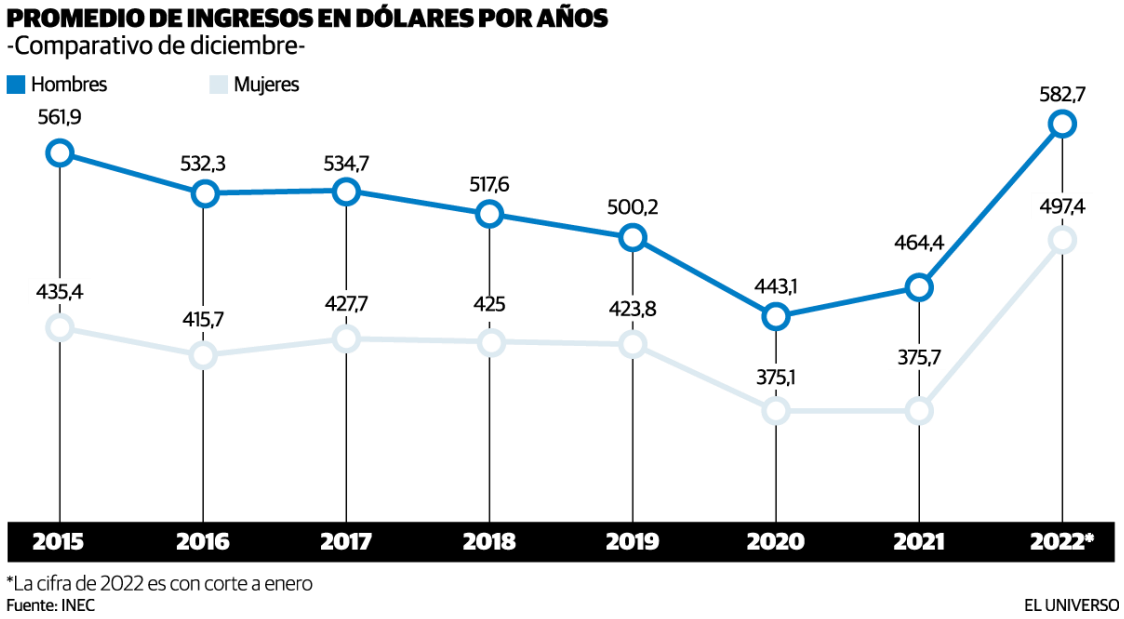
Ejemplos

El INEC captura datos socioeconómicos

Durante un censo por ejemplo la entidad

¡Captura información de millones de ecuatorianos!

Adicionalmente genera reportes periódicos de otros datos micro y macro económicos!



Ejemplos

Deportes

Durante un evento deportivo se capturan muchos números que permiten definir el rendimiento de los atletas.

#	Delegaciones	Total				Masculina				Femenina				Mixta			
					T				T				T				T
1	Colombia	171	104	79	354	80	49	37	166	81	47	37	165	10	8	5	23
2	Venezuela	61	69	78	208	36	41	35	112	22	28	39	89	3	0	4	7
3	Ecuador	40	51	55	146	17	19	24	60	19	26	27	72	4	6	4	14
4	Chile	37	49	71	157	17	22	41	80	19	26	28	73	1	1	2	4
5	Perú	33	40	72	145	10	17	33	60	21	20	35	76	2	3	4	9
6	República Dominicana	22	24	44	90	12	9	22	43	8	14	18	40	2	1	4	7
7	Guatemala	7	18	23	48	0	9	7	16	6	8	11	25	1	1	5	7
8	Paraguay	7	15	15	37	4	8	7	19	3	5	8	16	0	2	0	2
9	Panamá	5	8	20	33	2	3	9	14	3	5	11	19	0	0	0	0
10	El Salvador	4	5	5	14	3	1	3	7	0	2	2	4	1	2	0	3
11	Bolivia	2	6	6	14	1	4	4	9	1	2	1	4	0	0	1	1
	Total	389	389	468	1246	182	182	222	586	183	183	217	583	24	24	29	77

Datos

Entonces, como podemos ver, toda clase de información capturada puede ser considerado un dato.

Lo importante es que con información tenemos el poder de crear nueva información.

Aplicar operaciones estadísticas, generar reportes, visualizaciones, etc.

Datos

Uno puede encontrar datos en muchos contextos.

1. En servicios web, se distribuyen o utilizan con el fin de ejecutar cierta rutina u ofrecer cierta interfaz para acceder o almacenar información.

2. Para análisis, en donde se explotan los datos con el fin de responder incertidumbres o contestar preguntas de diferentes tipos.

¡Y muchos otros más!

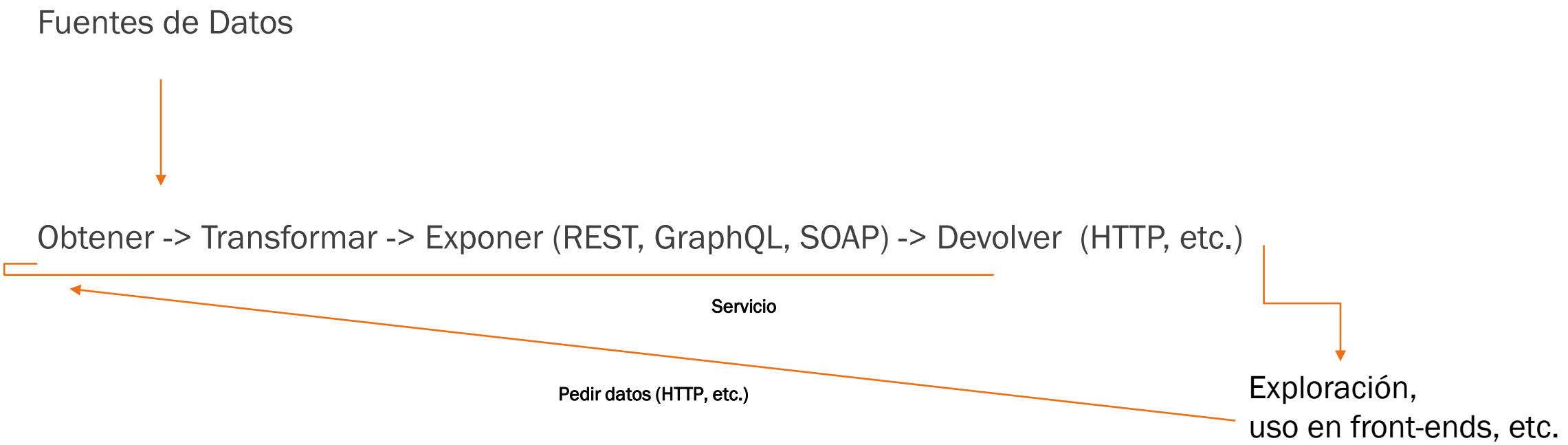
La vida del análisis de datos

Fuentes de Datos

Obtener -> Limpiar -> Integrar -> Analizar -> Visualizar

Reportes, exploración, etc.

La vida de los servicios de datos



Trabajando en análisis

¡Un analista de datos simplemente trabajará en las diferentes partes de aquel gráfico!

Por ejemplo, antes de las computadoras uno podía manualmente analizar información y crear estimaciones.

Los computadores simplemente nos permiten hacerlo de manera más eficiente.

Trabajando en análisis

De hecho, uno no necesita programar. ¡Existen muchas utilidades que no requieren que el analista tenga conocimientos avanzados de bases de datos o lenguajes!

Por ejemplo, Excel. Funciona como una base de datos, y a su vez nos permite realizar operaciones matemáticas.

¿Por qué usar Python?

Por su facilidad de uso, una persona puede fácilmente entender el lenguaje y aprovecharlo para realizar operaciones analíticas de manera más eficiente.

Para empezar, es gratuito y al ser un lenguaje, permite fácil colaboración entre los diferentes usuarios del mismo.

Python para análisis de datos

La presencia de muchas librerías permite empoderar al analista mediante utilidades que interactúen con distintos entornos (leer archivos, leer bases de datos, etc.)

Herramientas como Excel por ejemplo son bastante útiles en ciertos contextos, pero a medida que los sets de datos se vuelven más grandes, la utilidad pierde sus beneficios.

La vida del análisis

Fuentes de Datos

Obtener -> Limpiar -> Integrar -> Analizar -> Visualizar

Reportes, exploración, etc.

En Python esto simplemente identifica los diferentes mecanismos disponibles para acceder a datos, ya sea por medio de archivos, bases de datos, o servicios web.

La vida del análisis

Fuentes de Datos

Obtener -> **Limpiar** -> Integrar -> Analizar -> Visualizar

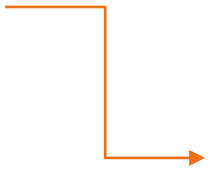
Reportes, exploración, etc.

Los datos vienen en muchas formas, a veces estructurados, a veces no. Limpiar se refiere al proceso de estandarizarlos en forma que puedan ser utilizados para el análisis correspondiente.

La vida del análisis

Fuentes de Datos

Obtener -> Limpiar -> **Integrar** -> Analizar -> Visualizar



Reportes, exploración, etc.

Integrar simplemente hace referencia a unificar las fuentes de datos requeridas en el contexto analítico.

La vida del análisis

Fuentes de Datos

Obtener -> Limpiar -> Integrar -> **Analizar** -> Visualizar

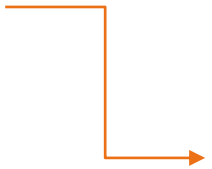
Reportes, exploración, etc.

Analizar hace referencia a la información que queremos obtener de esos datos, esto depende mucho del problema en el que estamos enfocándonos.

La vida del análisis

Fuentes de Datos

Obtener -> Limpiar -> Integrar -> Analizar -> Visualizar



Reportes, exploración, etc.

Visualizar hace referencia a las distintas formas en las que podemos explorar nuestro análisis en el contexto actual, por ejemplo, creando gráficos, o imprimiendo tablas numéricas.

La vida del análisis

Fuentes de Datos

Obtener -> Limpiar -> Integrar -> Analizar -> Visualizar

Reportes, exploración, etc.

El objetivo final del análisis es siempre contestar una o más preguntas. La forma en que esto se haga depende del problema y la razón por la que lo ejecutamos.

Análisis

¿Pero a qué nos referimos específicamente cuando decimos análisis?

Es nada más el proceso mediante el cual obtenemos nueva información a través de métodos analíticos.

Esto puede ser algo simple como métodos estadísticos base (promedios, medianas), modelos estadísticos avanzados (regresiones, inferencias), predicciones, ¡o inclusive análisis visual (contestar preguntas viendo los datos obtenidos a mano)!

Análisis

Al ser una clase enfocada en el contexto de servicios web, no entraremos en el mundo de análisis, pero existen muchos recursos disponibles en Python para operar en sets de datos de manera básica.

A Beginner's Guide to Data Analysis in Python | by Natassha Selvaraj | Towards Data Science

Data Analytics Definition (investopedia.com)

The Age Of Analytics And The Importance Of Data Quality (forbes.com)

Procesar Datos

En realidad, lo importante para este curso es la necesidad de leer o recolectar datos.

1. ¿Dónde se encuentra esta información?
2. ¿Cómo debo ajustarla para cumplir mis objetivos?

Almacenamiento

Los datos pueden almacenarse de diferentes formas, y es importante entender que dependiendo de la clase de datos, es posible que necesitemos realizar diferentes procesos para extraer dicha información.

Almacenamiento: Bases de datos

Una base de datos es una colección organizada de información estructurada almacenada electrónicamente, y usualmente gestionada por un sistema de gestión de bases de datos (DBMS).

Comúnmente, los datos en una base de datos se modelan en tables con columnas y filas, denominándose bases de datos relacionales (relaciones entre tablas). Por ejemplo MySQL, Postgres, etc.

Sin embargo, existen bases de datos no relacionales que almacenan la información de manera diferente, por ejemplo como objetos puros (mongo, cassandra, etc.) o en una relación de grafos (neo4j).

Almacenamiento: Bases de datos

Las bases de datos relacionales generalmente permiten acceder y gestionar la información dentro de las mismas utilizando un lenguaje conocido como SQL.

SQL permite que tanto un usuario como una aplicación accedan a la información en las mismas de manera estructurada!

Almacenamiento: Bases de datos

Generalmente todos los lenguajes de programación incluyen una o más librerías que permiten a un usuario interactuar con bases de datos.

En este curso introduciremos como conectarse a bases de datos relacionales mediante Python, sin embargo existen de igual manera mecanismos muy bien documentados para interactuar con bases de datos NoSQL (nombre utilizado para todas aquellas que no siguen el modelo relacional).

sqlite3 en Python

sqlite3 es una base de datos relacional muy sencilla, no requiere un motor o servidor ya que los datos de la misma viven uno o más archivos! Python incluye una librería de manera predeterminada!

```python
import sqlite3
from sqlite3 import Error

def crear_conexion(parametros):
    conexion = None
    try:
        conexion = sqlite3.connect(parametros)
        print("Conexión exitosa")
    except Error as e:
        print(f"Error '{e}'")

    return conexion

conexion =
crear_conexion("ubicacion/del/archivo.sqlite")
conexion.execute("select * from tabla")
resultados = conexion.fetchall()
print(resultados)
conexion.close()
```

[sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.10.5 documentation]

MySQL en Python

A diferencia de sqlite3, Python no incluye módulos para interactuar con MySQL, ¡pero existen muchos disponibles en el repositorio libre!

```
python -m pip install mysql-connector-python
```

```python
import mysql.connector
from mysql.connector import Error

def crear_conexion(servidor, usuario, clave):
    conexion = None
    try:
        conexion = mysql.connector.connect(
            host=servidor,
            user=usuario,
            passwd=clave
        )
        print("Conexión a MYSQL existosa")
    except Error as e:
        print(f"Error '{e}'")
    return conexion

conexion = crear_conexion("localhost", "miusuario", "")
cursor = conexion.cursor()
cursor.execute("SELECT * FROM tabla")
resultado = cursor.fetchall()
conexion.close()
```

[MySQL :: MySQL Connector/Python Developer Guide]

SQL y NoSQL en Python

¡En realidad, existen librerías para cualquier sistema de base de datos común!

- PostreSQL
- Cassandra
- MongoDB
- etc, etc, etc.

Su mejor amigo siempre **Google**.

sqlalchemy

Un kit de utilidades para interactuar con bases de datos ofrece la flexibilidad de trabajar con bases de datos relacionales de manera más eficiente, en particular dentro de aplicaciones complejas.

- Ofrece un modelo relacional que permite operar en una base de datos usando objetos nativos a Python.
- Un sistema eficiente de distribución de objetos y operaciones relacionales usando Python en lugar de SQL.
- Generación e introspección de bases de datos.

SQLAlchemy · PyPI

Archivos

A pesar de que las bases de datos son maneras óptimas de mantener información, los archivos son una forma también muy común de almacenar objetos de datos, ¡particularmente en el contexto de análisis!

SQLAlchemy · PyPI

Archivos CSV

CSV – comma separated values

Un formato tabular en el que podemos representar una table de información mediante una serie de comas que indiquen la separación entre columnas.

Al ser un formato de texto plano, el mismo puede leerse de manera sencilla mediante cualquier utilidad, e inclusive por programas como Excel.

Archivos CSV

Un archivo CSV es bastante simple, por ejemplo:

```
Usuario;Identificador;Nombre;Apellido
booker12;9012;Rachel;Booker
grey07;2070;Laura;Grey
johnson81;4081;Craig;Johnson
jenkins46;9346;Mary;Jenkins
smith79;5079;Jamie;Smith
```

La primera línea puede definir los encabezados de cada columna, así como no definir nada.
¡Es necesario que entendamos la estructura del mismo cuando lo estemos analizando!

Archivos CSV: Python

Al ser un archivo de texto plano, es bastante fácil leer este archivo como cualquier otro:

```
with open("ubicacion.csv") as archivo:
    for fila in archivo:
        # Dividiamos las columnas usando la , pero tambien puede ser ;
        columnas = fila.split(",")
        print(columnas)
```

Archivos CSV: Python

Sin embargo, existen muchas particularidades que debemos controlar, y por eso es mejor usar la librería incluida con Python.

```
import csv
with open("ubicacion.csv") as archivo:
    # Definir el delimitador que separa las columnas
    filas = csv.reader(archivo, delimiter=';')
    for fila in filas:
        # Automáticamente se convirtieron en columnas
        print(fila[0])

# Podemos fácilmente agregar filas!
with open('ubicacion.csv', 'w') as archivo:
    escritor = csv.writer(archivo, delimiter=';')
    escritor.writerow(['Pollo', 'Papas'])
```

csv — CSV File Reading and Writing — Python 3.10.5 documentation

Archivos

Saber leer archivos CSV es muy importante ya que una gran cantidad de sets de datos pueden ser definidos de esta manera. De hecho librerías como pandas permiten de manera sencilla leer los mismos sin tener que gestionar el recurso de manera manual!

Sin embargo, cuando hablamos de cantidades muy extensas (muchos GBs), es importante considerar otros formatos más extensibles que no sean necesariamente texto plano, entre ellos Parquet, que no cubriremos en esta clase por ser un concepto más avanzado, pero que no va mal conocer.

Parquet – Databricks

Servicios web

Tal como aprendimos, es posible comunicarse con servicios web alrededor del mundo.

Muchos sets de datos pueden ser obtenidos mediante servicios web.

Datos inesperados

A veces los datos no se encuentran de forma estructurada, o debemos recogerlos mediante herramientas que permitan extraerlos de manera eficiente.

Por ejemplo, a veces la información se encuentra en archivos PDF, o en páginas web, por lo tanto debemos procesar el contenido con librerías especializadas a manera de poder extraerla.

¡Casi siempre existe una librería robusta para procesar formatos especiales!