

Programación para el análisis de datos

Federico Pousa
fpousa@udesa.edu.ar

Objetivo

Este curso presenta los conceptos y técnicas de programación generales esenciales para una audiencia de análisis de datos sin experiencia previa en programación.

El objetivo es equipar a los estudiantes con las habilidades de programación necesarias para tener éxito en los otros cursos del programa.

Vamos a separar en 2 la materia

1era parte: Python

2da parte: R (Con Daniel Fraiman)

Programa

1. Introducción a la computación. Introducción a la programación en Python. Entorno de ejecución (python, ipython, jupyter notebook). Instaladores de paquetes (pip). Variables, estructuras de control (condicionales, ciclos). Tipos de datos. Estructuras de datos en Python (diccionarios, listas por comprensión). Clases. Funciones. Manejo de errores, excepciones. Manejo de archivos. debugging
2. Bibliotecas numéricas (numpy)
3. Bibliotecas para estructuración de datos (pandas)
4. Bibliotecas de herramientas estadísticas (scipy)
5. Bibliotecas para graficar (matplotlib, seaborn)

Cronograma

2023-03-02	Jueves	Python intro I
2023-03-04	Sábado	Python intro II
2023-03-07	Martes	Vectorial: numpy+scipy
2023-03-09	Jueves	Dataframes: Pandas
2023-03-11	Sábado	Graficación: matplotlib + seaborn + plotly
2023-03-14	Martes	Terminar guías en clase y dudas Presentar TP, datos y validar hipótesis
2023-03-16	Jueves	TP en clase

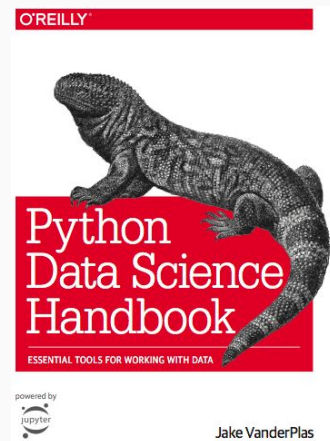
Evaluación

- Trabajo práctico en grupos de 3 alumnos.
- Resolución de consignas.
- Creación de hipótesis, validación y respuesta.
- Informe.
- Evaluación individual a definir.

Bibliografía

- Infinidad de fuentes en internet
- “Libro”: VanderPlas, Jake. Python data science handbook: Essential tools for working with data. " O'Reilly Media, Inc.", 2016.

<https://jakevdp.github.io/PythonDataScienceHandbook/>



Comunicación

- Email: fpousa@udesa.edu.ar
- Campus: <https://campusvirtual.udesa.edu.ar/>
- Dictado de clases: Zoom en el campus
- Clase taller:
 - Servidor de Discord en el campus

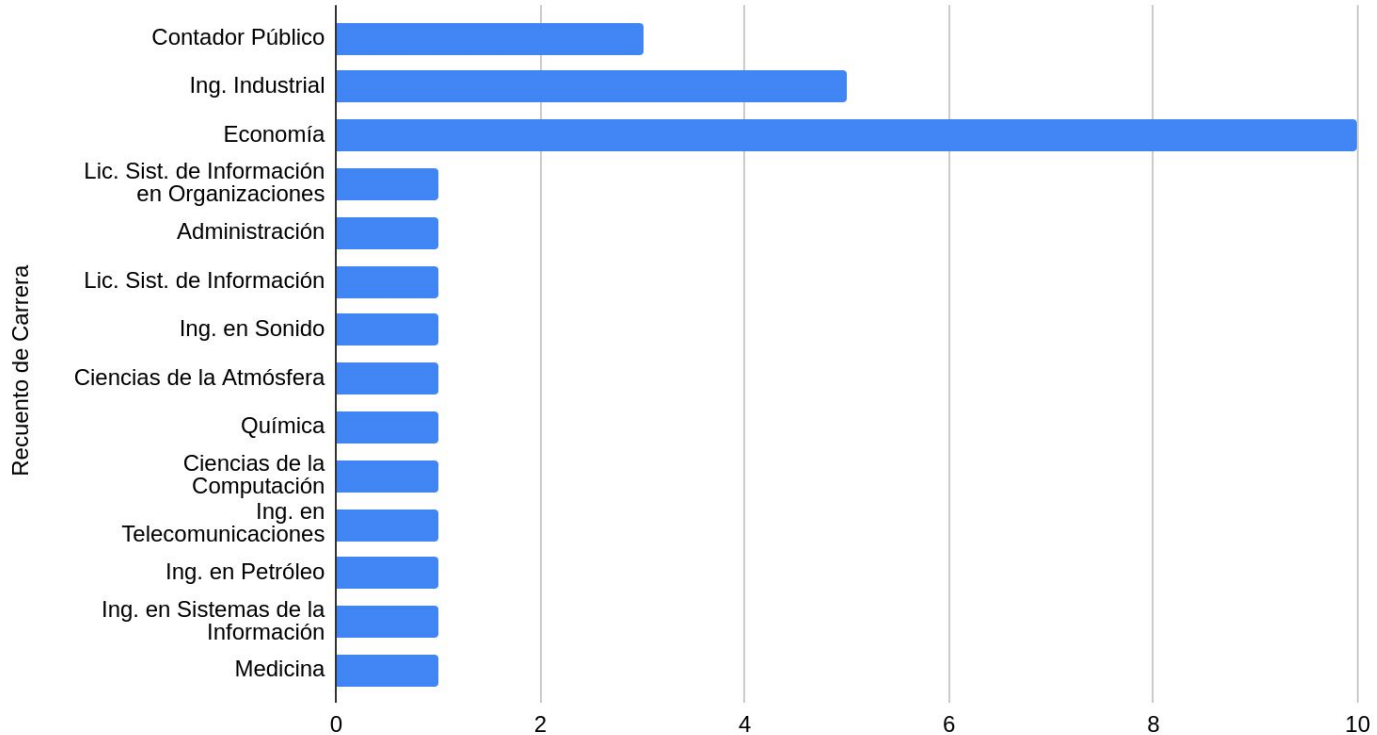
I me mine

- Doctor en Ciencias de la Computación (UBA), Área Investigación Operativa.
- Docente en Ciencias de la Computación (UBA), Maestría en Data Science (UDESA), Maestría en Management+Analytics (UTDT).
- Socio y Data Science & Operations Research en Eryx.
- <https://www.linkedin.com/in/federico-pousa/>

Ustedes

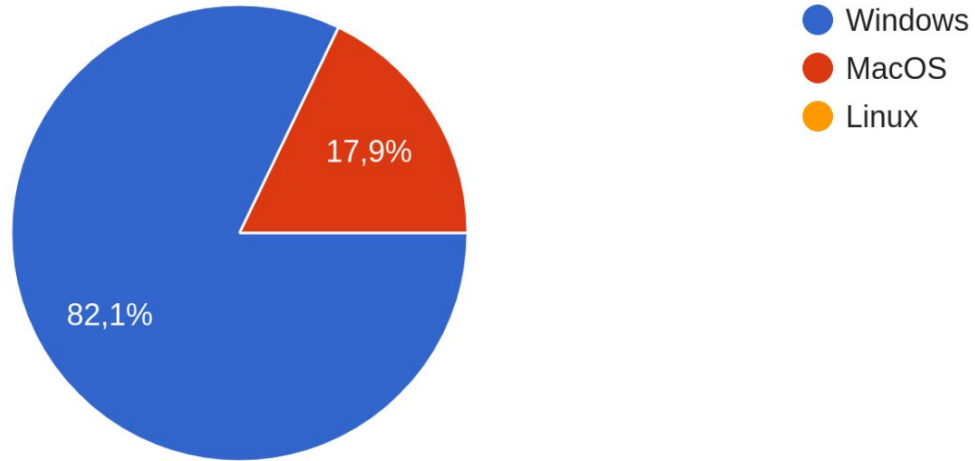
Ustedes

Recuento de Carrera



¿Qué sistema operativo usás?

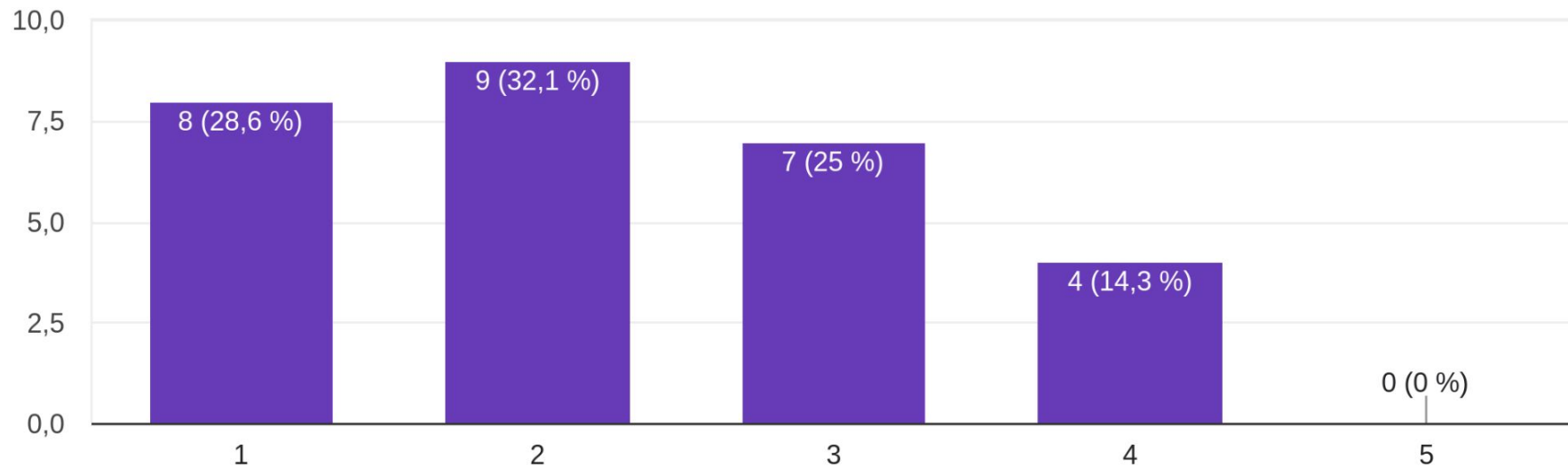
28 respuestas



Ustedes

¿Cuánta experiencia te parece que tenés usando la terminal/consola de tu sistema operativo?

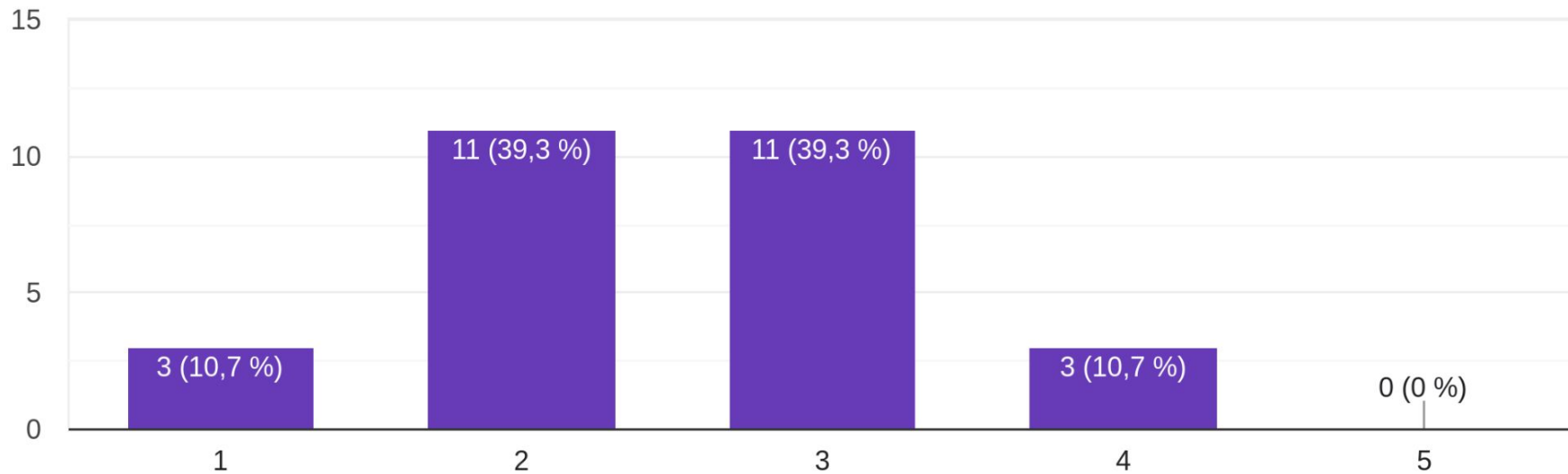
28 respuestas



Ustedes

¿Cuánta experiencia sentís que tenés en Python?

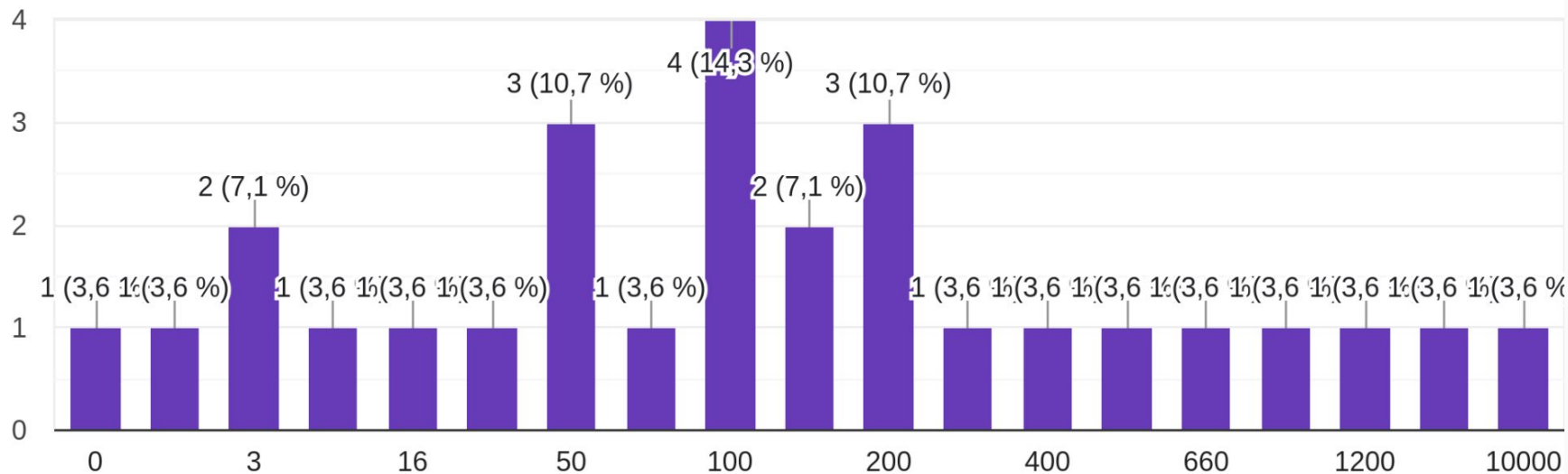
28 respuestas



Ustedes

¿Cuántas líneas tiene el código más largo que escribiste?

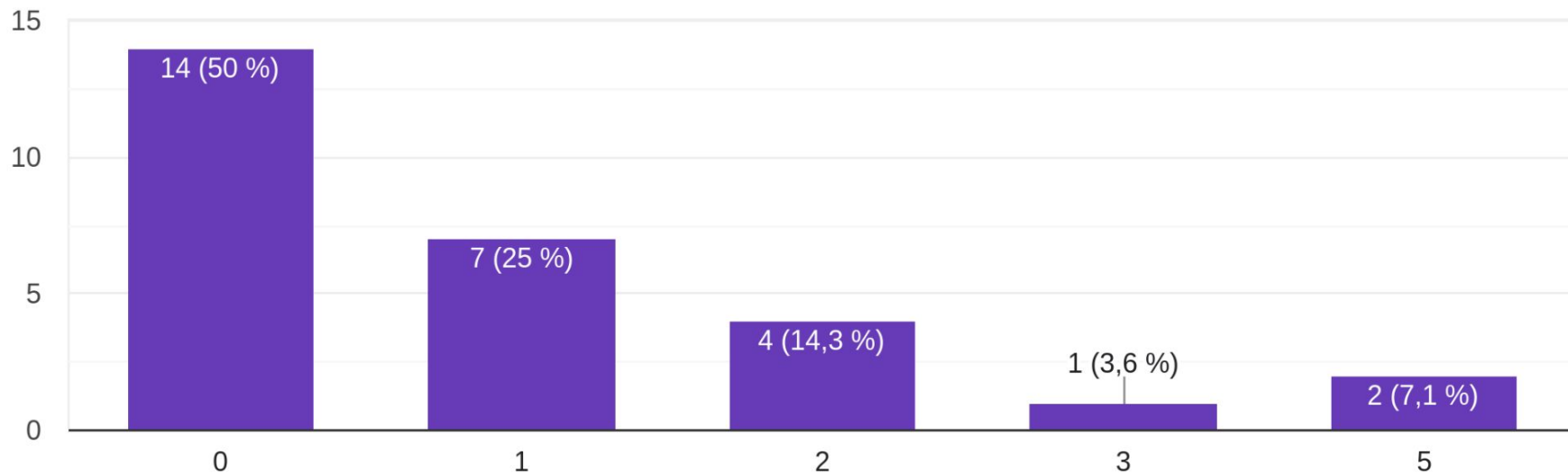
28 respuestas



Ustedes

¿Cuántos días por semana escribís código Python?

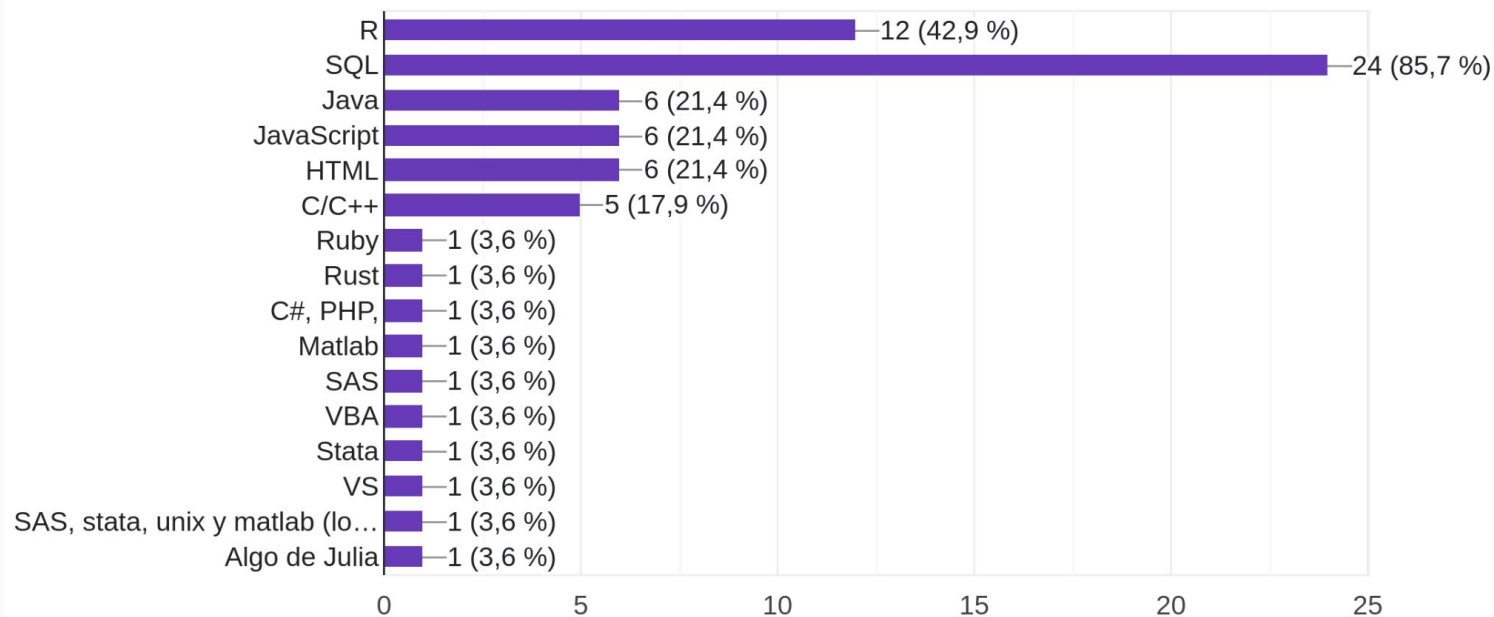
28 respuestas



Ustedes

¿Qué otros lenguajes utilizaste?

28 respuestas



Ustedes

¿Qué aplicaciones de Data Science conocés?					
STATA, EViews, EXCEL, ACCESS					
Salud, ciencias sociales, comunicaciones, procesos productivos, finanzas					
Varias; en mi trabajo anterior realizábamos workshops con clientes para el descubrimiento de casos de aplicación de Data Science en diversas industrias.					
ninguna					
Por nombrar algunas, análisis de sentimientos, reconocimiento de patrones, detección de anomalías, etc					
ninguna					
AB Testing, scoring crediticio, segmentación, análisis predictivo					
Análisis de Datos, visualización de Datos, construcción de Modelos utilizando regresiones (tanto lineales como logísticas), implementación de desarrollos a producción.					
Reconocimiento de patrones					
no sé a que se refieren con aplicaciones, pero creo que ninguna					
PowerBI					
0					
La mayoría con la industria del MKT, detección de canales fraudulentos, retención de usuarios, predicción de CPA.					
Modelos predictivos, reconocimiento de patrones, clustering					
Entendiendo aplicaciones como los usos posibles, se me vienen primero a la mente: Análisis predictivos con regresiones básicas (por ejemplo, evoluciones de venta, respuesta o					
Ninguna					
modelado predictivo, reconocimiento de patrones, la detección de anomalías, clasificación y categorización. Análisis de sentimiento, sistemas de recomendación.					
Detección de churn, predicción de life time value, Detección de fraude, scoring crediticio, segmentación de clientes, predicción de la demanda, procesamiento de lenguaje natural,					
Distintos IDE de los lenguajes mencionados (RStudio, Jupyter, Spyder, SAS Enterprise) y editores como VSC.					
(Si es que la pregunta se refería a esto)					
Si la pregunta en cuestión se trata de una app, no conozco alguna en particular.					
Casi interminable la lista de aplicaciones de machine learning.					
Sistemas de recomendación, Análisis predictivo, Detección de Anomalías, Clasificación, otros					
Power BI, Tableau, Qlik					
Modelado para detectar ciertos atributos					
Ninguna					
Detección de anomalías, detección de fraude, modelado predictivo (ventas/stock/default créditos), aplicaciones de NLP/Audio/Computer vision varias					
-					
Análisis predictivo, Detección de anomalías, Clasificación					

ChatGPT y DataScience



hello



Hello! How can I help you today?



do you know a good dataset about retail operations?



There are several datasets available related to retail operations, some popular ones include:



1. Retail Sales Dataset: This dataset contains the historical sales data for 45 stores of a retail company. It includes information about the date, store number, department, and weekly sales.
2. Retail Trade Panel Data: This dataset includes information on retail trade sales and inventory levels for over 25,000 stores.
3. Online Retail Data Set: This dataset contains all the transactions made for an online retail company based in the UK over a period of one year.
4. Instacart Market Basket Analysis: This dataset contains over 3 million grocery orders from more than 200,000 Instacart users.
5. Walmart Recruiting - Store Sales Forecasting: This dataset contains historical sales data for 45 Walmart stores located in different regions.

Please keep in mind that some of this data may require an application or purchase to access, and it's important to check the terms of use and data availability before using it.



please point me to one of them, I want one that is free to use and preferably not that large



One dataset that is free to use and relatively small in size is the Online Retail Data Set. This dataset contains all the transactions made for an online retail company based in the UK over a period of one year. The dataset includes information about the customer, the product, and the transaction, such as the invoice number, the stock code, the description of the product, the quantity purchased, the invoice date, and the unit price.



You can find the dataset here:

<https://archive.ics.uci.edu/ml/datasets/Online+Retail>

Please keep in mind that when using this dataset, you should reference the source and follow the terms of use.



perfect, so I downloaded that dataset, do you know it? do you know the columns of the dataset?



Yes, I am familiar with the Online Retail Data Set. The dataset contains 8 variables or columns:



1. "InvoiceNo": A unique identifier for the transaction.
2. "StockCode": A unique identifier for the product.
3. "Description": A short text describing the product.
4. "Quantity": The number of items purchased.
5. "InvoiceDate": The date and time of the transaction.
6. "UnitPrice": The price of one item.
7. "CustomerID": The unique identifier for the customer.
8. "Country": The country where the customer is located.

Please note that the CustomerID column has some missing values, so it may not be complete for all rows. Also, some InvoiceNo starts with 'C' indicating it was a cancellation of the order.

ChatGPT y DataScience



can you give me a piece of code to read this dataset in python?



Sure, here is a piece of code that uses the Pandas library to read the Online Retail Data Set in Python:



python

Copy code

```
import pandas as pd

# Read the csv file into a dataframe
df = pd.read_csv('OnlineRetail.csv', sep=',', header=0, encoding='latin1')

# Show the first 5 rows of the dataframe
print(df.head())
```

This code first imports the Pandas library, then reads the 'OnlineRetail.csv' file into a DataFrame. The `read_csv()` function takes in several parameters such as the file path, the separator used in the csv file (`sep=','`), the number of the row that contains the column names (`header=0`) and the encoding used in the file (`'latin1'`). The last line of code prints out the first 5 rows of the DataFrame using the `head()` function.

You can adjust the code to suit your needs and requirements.

Please make sure that the csv file is in the same directory of your python script or Jupyter notebook or provide the complete path of the file.

ChatGPT y DataScience



mmmm, but I didn't find a csv, I found a xlsx file



If you have the Online Retail dataset in an .xlsx format, you can still use Pandas to read it into a DataFrame. You will need to use the `read_excel()` function instead of `read_csv()`. Here is an example of how to read an .xlsx file:



python

Copy code

```
import pandas as pd

# Read the excel file into a dataframe
df = pd.read_excel('OnlineRetail.xlsx', sheet_name='Sheet1')

# Show the first 5 rows of the dataframe
print(df.head())
```

This code reads an excel file named 'OnlineRetail.xlsx' and loads the data into a DataFrame, also specifying the sheet name that contains the data, in this case 'Sheet1'. The last line of code will print out the first 5 rows of the DataFrame using the `head()` function.

You can adjust the code to suit your needs and requirements.

Make sure that the xlsx file is in the same directory of your python script or Jupyter notebook or provide the complete path of the file.

Also keep in mind that you will need to have openpyxl library installed to read excel files. If it's not installed you can install it by running `!pip install openpyxl` in your Jupyter Notebook or running `pip install openpyxl` in your command line.



great, so I have the dataframe and you already know the columns, so maybe you can suggest something interesting to plot right?



Sure, here are a few ideas of interesting plots you can create with the Online Retail Data Set:



1. **Time Series Plot:** One interesting plot you could create is a time series plot of the number of transactions over time. You can group the transactions by date and then plot the number of transactions per day. This will give you an idea of how the number of transactions has changed over time and if there are any patterns or trends in the data.
2. **Bar Plot:** Another interesting plot you could create is a bar plot of the number of transactions by country. This will give you an idea of which countries are the most important for the online retail company and if there are any countries that the company should focus on.
3. **Scatter Plot:** You could also create a scatter plot to explore the relationship between the unit price and the quantity of items sold. This can give you an idea of which items are the most popular and if there is a relationship between the price and the number of items sold.
4. **Pie Chart:** you can create a pie chart representing the percentage of each country in the total sales.
5. **Heatmap:** you can create a heatmap showing the sales by hour of the day.

These are just a few examples of the types of plots you can create with the Online Retail Data Set. There are many other possibilities depending on the questions you are trying to answer and the insights you are trying to gain from the data.



To create these plots you can use libraries such as Matplotlib and Seaborn.

ChatGPT y DataScience



I think the second one is very interesting, share your code for that



Sure, here is an example of how you can create a bar plot of the number of transactions by country using the Online Retail Data Set and the Pandas and Matplotlib libraries:  

```
python  Copy code

import pandas as pd
import matplotlib.pyplot as plt

# Read the excel file into a dataframe
df = pd.read_excel('OnlineRetail.xlsx', sheet_name='Sheet1')

# Group the data by country and count the number of transactions
transactions_by_country = df.groupby('Country')['InvoiceNo'].nunique()

# Plot the data
transactions_by_country.plot(kind='bar', xlabel='Country', ylabel='Number of Transactions')
plt.title("Number of Transactions by Country")
plt.show()
```

This code first imports the Pandas and Matplotlib libraries, then reads the 'OnlineRetail.xlsx' file into a DataFrame and specifying the sheet name that contains the data. Then it groups the data by the 'Country' column and counts the number of unique 'InvoiceNo' values using the `groupby()` function, this will give you the number of transactions for each country.

ChatGPT y DataScience

Then it plots the data using the `plot()` function of Pandas, specifying the type of plot as a bar plot (`kind='bar'`), the x-axis label as 'Country', the y-axis label as 'Number of Transactions' and the title as "Number of Transactions by Country".

Finally, it shows the plot using the `show()` function of the Matplotlib library.

Keep in mind that this is just an example, you can adjust the code to suit your needs and requirements.

If you have missing values in the country field and you want to exclude them from the plot, you can use the following code to dropna values before the groupby operation:

python

Copy code

```
df = df.dropna(subset=['Country'])
```

Also, if you want to show the values of each bar you can use the following code snippet:

python

Copy code

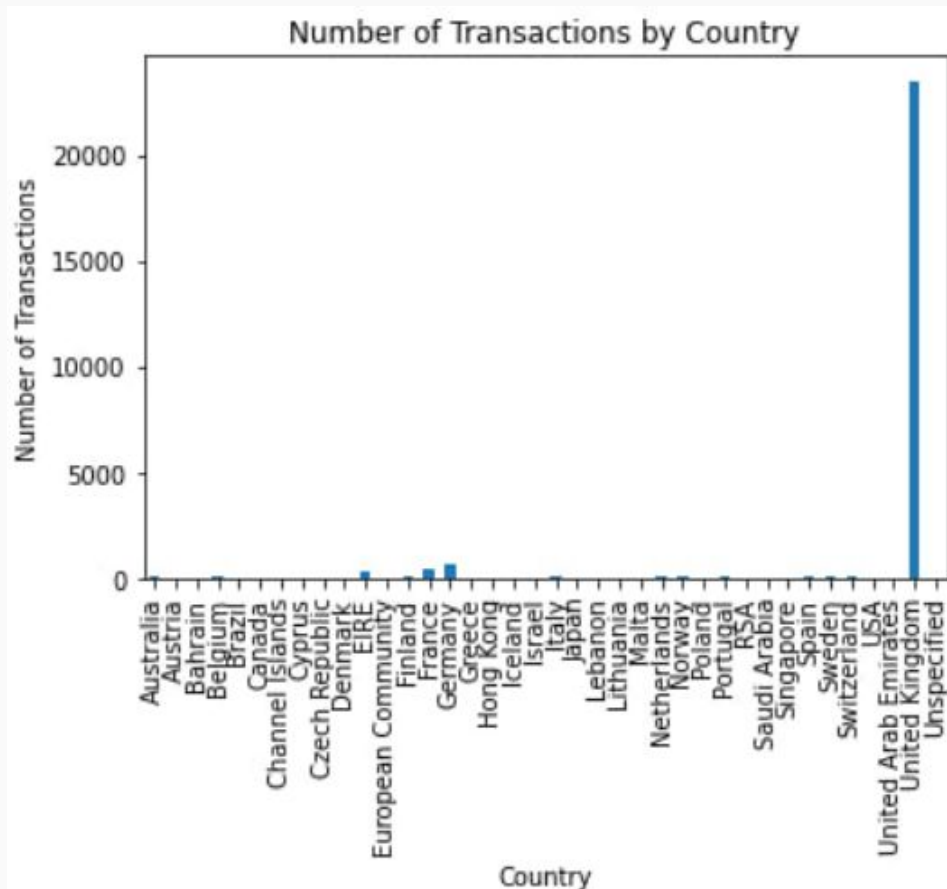
```
transactions_by_country.plot(kind='bar', xlabel='Country', ylabel='Number of Transactions')
for i in range(len(transactions_by_country)):
    plt.text(x = i*0.3, y = transactions_by_country[i]*1, s = transactions_by_country[i], s
```

If the number of countries is too high and you want to show only the top n countries that has the highest values, you can use the following code snippet:

python

Copy code

```
transactions_by_country = transactions_by_country.sort_values(ascending=False)[:n]
transactions_by_country.plot(kind='bar', xlabel='Country', ylabel='Number of Transactions')
plt.title("Number of Transactions by Country")
plt.show()
```



ChatGPT y DataScience



I'm good with the first code, I don't think I need the following ones for now, buuuut, I have one problem, the names of the countries in the graph are completely vertical so they can't be read very easily, can you modify the code so they appear rotated by some angle, let's say 30 degrees?



Yes, you can rotate the x-axis labels by a certain angle by setting the rotation parameter of the `xticks()` function in Matplotlib. Here is an example of how to rotate the x-axis labels by 30 degrees in the previous code:



python

Copy code

```
transactions_by_country = df.groupby('Country')['InvoiceNo'].nunique()
transactions_by_country.plot(kind='bar', xlabel='Country', ylabel='Number of Transactions')
plt.title('Number of Transactions by Country')
plt.xticks(rotation=30)
plt.show()
```

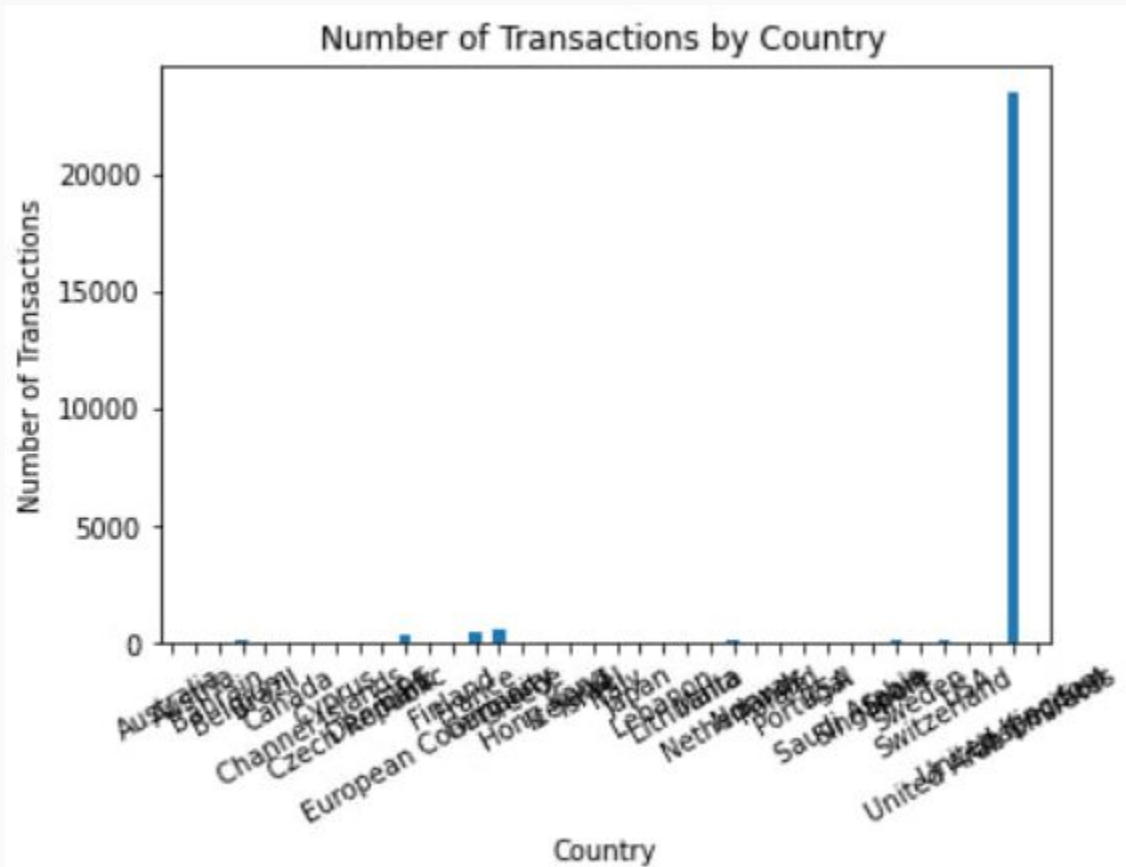
This code simply rotates the x-axis labels by 30 degrees. You can adjust the angle to whatever value you want.

If you want to rotate the x-axis labels in a horizontal position you can use the following code snippet:

python

Copy code

```
transactions_by_country = df.groupby('Country')['InvoiceNo'].nunique()
transactions_by_country.plot(kind='bar', xlabel='Country', ylabel='Number of Transactions')
plt.title('Number of Transactions by Country')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```





I used the first code and it worked correctly but matplotlib is not very nice out of the box, how do you suggest that we can make this graph more appealing



There are many ways to make a Matplotlib graph more visually appealing, depending on the specific requirements of your project and the audience you are presenting the graph to. Here are a few general suggestions to improve the appearance of your graph:



1. Use a different color palette: Matplotlib's default color palette can be quite bland. You can use a different color palette, such as the ones provided by Seaborn, to make your graph more colorful and visually interesting.
2. Change the style of the graph: Matplotlib provides several built-in styles that you can use to change the appearance of your graph. You can experiment with different styles to see which one looks best for your data.
3. Add gridlines: Gridlines can help to make the graph more readable, especially if you have a lot of data points.
4. Add a legend: A legend can help to clarify the meaning of different parts of the graph.
5. Add a title and axis labels: A title and axis labels can help to provide context for the graph and make it more self-explanatory.

Here is an example of how to apply some of these suggestions to the previous code:

```
python Copy code

import seaborn as sns

transactions_by_country = df.groupby('Country')['InvoiceNo'].nunique()
sns.set_style("darkgrid")
sns.barplot(x=transactions_by_country.index, y=transactions_by_country.values, palette="Blue")
plt.title("Number of Transactions by Country")
plt.xlabel("Country")
plt.ylabel("Number of Transactions")
plt.xticks(rotation=30)
plt.show()
```

This code uses the seaborn library to change the style of the graph, it uses a blue color palette and adds gridlines, xlabel and ylabel to the graph. You can play around with the different options provided by seaborn and matplotlib to make your graph more visually appealing.

Please let me know if you need more help.

ChatGPT y DataScience



there are too many countries, and also there is one country that has a very large value so it kinda breaks all the graphs, do you think you can solve that?



Yes, there are several ways to handle cases where there are too many data points or outliers in a graph.



One way to handle this issue is to limit the number of countries displayed on the graph by only showing the top n countries with the highest number of transactions. Here is an example of how to do this:

python

Copy code

```
import seaborn as sns

transactions_by_country = df.groupby('Country')['InvoiceNo'].nunique().sort_values(ascending=False)
top_n = 10 # number of countries to display
transactions_by_country = transactions_by_country[:top_n]
sns.set_style("darkgrid")
sns.barplot(x=transactions_by_country.index, y=transactions_by_country.values, palette="Blue")
plt.title("Number of Transactions by Country (Top 10)")
plt.xlabel("Country")
plt.ylabel("Number of Transactions")
plt.xticks(rotation=30)
plt.show()
```

Another way to handle this is to use a logarithmic scale on the y-axis. Logarithmic scales can be useful when the data has a wide range of values and the difference between small values is much greater than the difference between large values. Here's an example of how to do this:

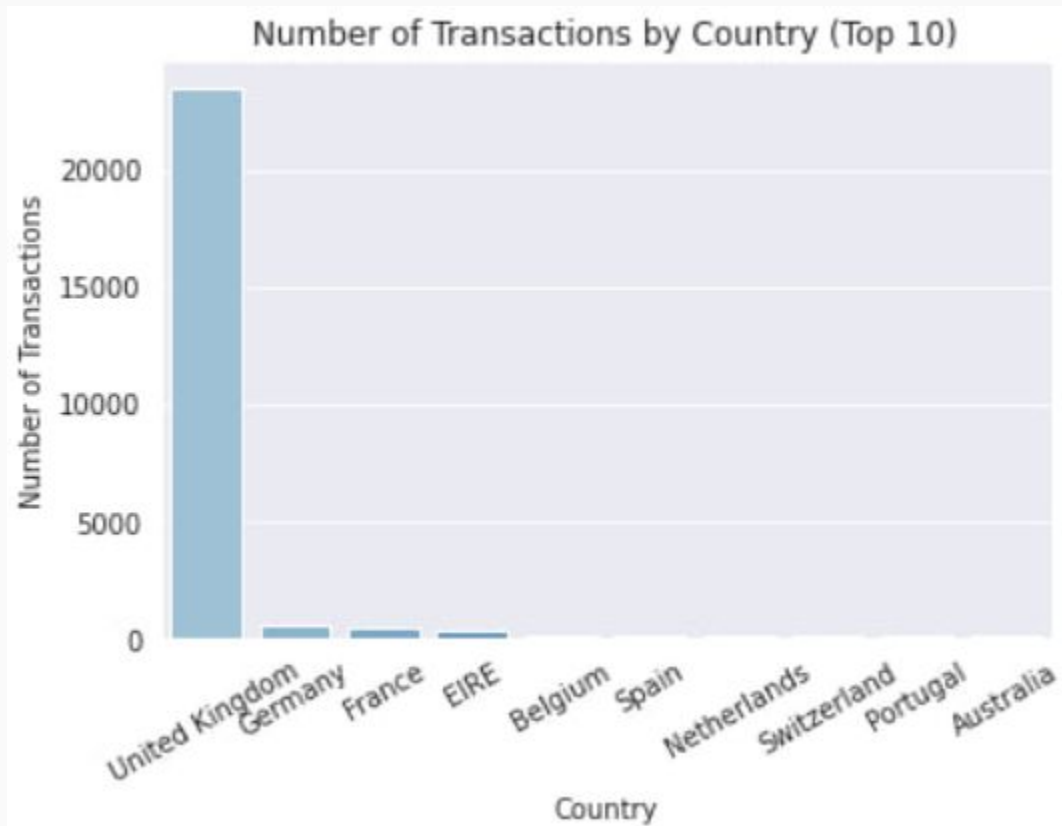
```
python Copy code

import seaborn as sns

transactions_by_country = df.groupby('Country')['InvoiceNo'].nunique()
sns.set_style("darkgrid")
sns.barplot(x=transactions_by_country.index, y=transactions_by_country.values, palette="Blue")
plt.yscale('log')
plt.title("Number of Transactions by Country")
plt.xlabel("Country")
plt.ylabel("Number of Transactions (log scale)")
plt.xticks(rotation=30)
plt.show()
```

You can also use a combination of both options and play around with the values to see what works best for your data. Also, you can group some countries as "Other" to reduce the number of countries shown in the graph.

Please let me know if you need more help.



¿Ciencia de Datos?

¿Ciencias de datos?



Dan Ariely

January 6, 2013 at 6:17pm · 🌐



Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it...



Like



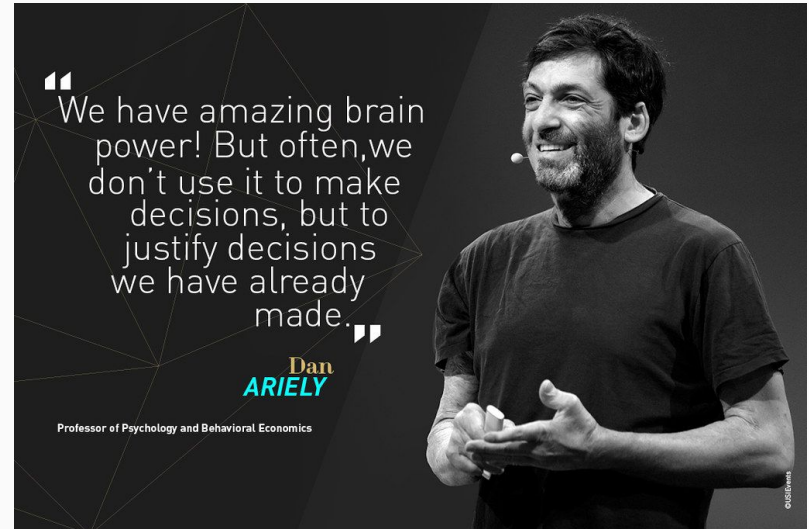
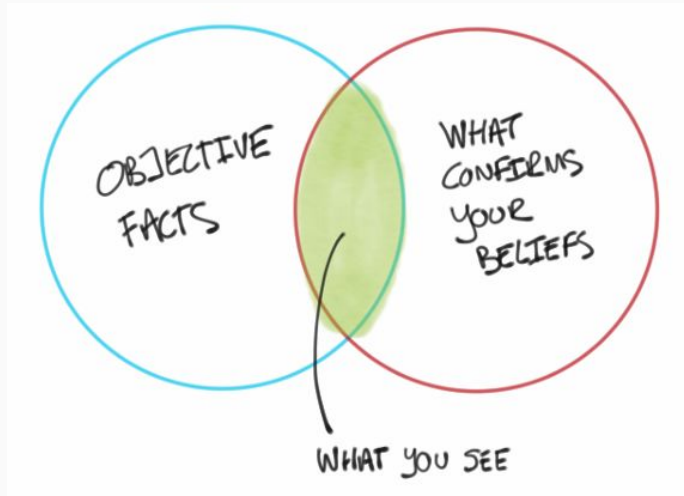
Comment



Share

¿Ciencias de datos?

- Poder hacer las preguntas correctas
- Poder abordar estas preguntas con los métodos correctos
- Saber transmitir lo que encontramos



¿Ciencias de datos?

- Obtener datos
- Organizar y limpiar datos
- Interpretar los datos
- Modelar los datos
- Mostrar los resultados



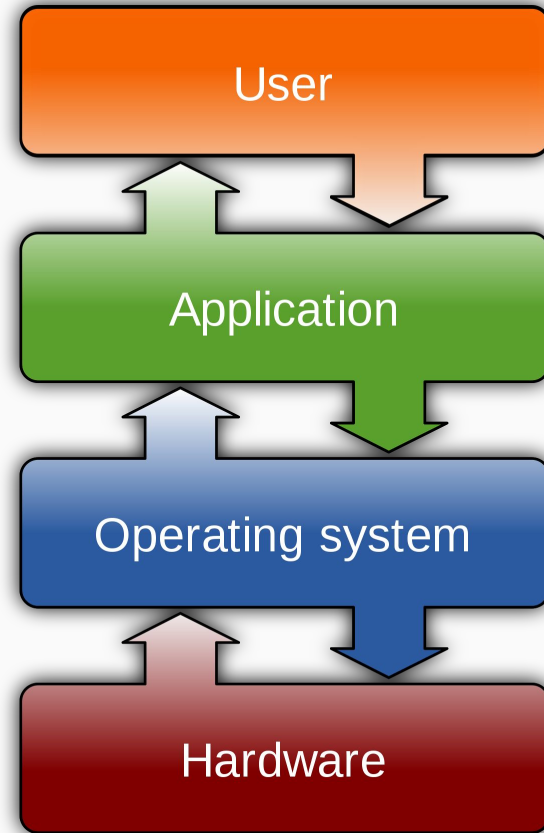
Vamos a tener que saber programar ...

Computación

Computadoras

- Hasta 1945 hubieron máquinas que computaban mecánicamente
- Transición hacia válvulas
- 1955 Primeros transistores
- 1965 Primeros circuitos integrados
- 1980 Primeras computadoras personales
- 2000 Placas de video, internet, computadoras personales accesibles





Wikipedia contributors. (2020, January 5). Memory management. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:13, February 10, 2020, from https://en.wikipedia.org/w/index.php?title=Memory_management&oldid=934247639

¿Por qué Python?

- Lenguaje de programación de alto nivel
- Interpretado
- Orientado a objetos
- Tipado dinámico
- Fácil interacción con otros lenguajes
- Muy bien soportado
- Version ≥ 3
- Multi-plataforma (Linux, macOS, Windows, etc)

Worldwide, Feb 2023 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.7 %	-0.7 %
2		Java	16.79 %	-1.3 %
3		JavaScript	9.65 %	+0.6 %
4	↑	C#	6.97 %	-0.5 %
5	↓	C/C++	6.87 %	-0.6 %
6		PHP	5.23 %	-0.8 %
7		R	4.11 %	-0.1 %
8	↑↑	TypeScript	2.83 %	+0.8 %
9		Swift	2.27 %	+0.3 %
10	↓↓	Objective-C	2.25 %	-0.1 %
11	↑↑	Go	1.95 %	+0.7 %
12	↑↑	Rust	1.91 %	+0.9 %

<https://pypl.github.io/PYPL.html>

- ¿Cómo podemos escribir Python?:
 - Consola
 - Script
 - Jupyter notebooks
 - Colab
 - IDEs
 - ODEs
- ¿Dónde lo podemos ejecutar?:
 - Local
 - Nube

¿Cómo empezamos?



- Siéntanse cómodos
- Siéntanse incómodos

“A person that is feeling very comfortable using a computer, next to a person that is feeling very uncomfortable using a computer, realistic painting” - Midjourney 2023



Python: Instalación

1. Descargamos Python: <https://www.python.org>
2. Instalamos Python
3. Esto nos va a permitir poder usar el intérprete de Python

MacOS/Linux: terminal

Windows: cmd

4. `python3`

```
fedepousa@Minerva:~$ python3
Python 3.8.10 (default, Nov 14 2022, 12:59:47)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Python: Instalación pip

¿**Qué** es PIP?

- PIP es un instalador de paquetes de python
- Multi-plataforma

Instalar *cualquier* cosa (desde una consola), por ejemplo:

- `pip install numpy`
- `pip install pandas`
- `pip install nltk`

¿**Cómo** instalamos PIP? (si no lo instalamos previamente)

- Bajamos `get-pip.py` (<https://bootstrap.pypa.io/get-pip.py>)
- Lo ejecutamos
- Ya está :D

Python: Uso, entorno, jupyter

Por ahora, para simplificar vamos a usar **Jupyter**

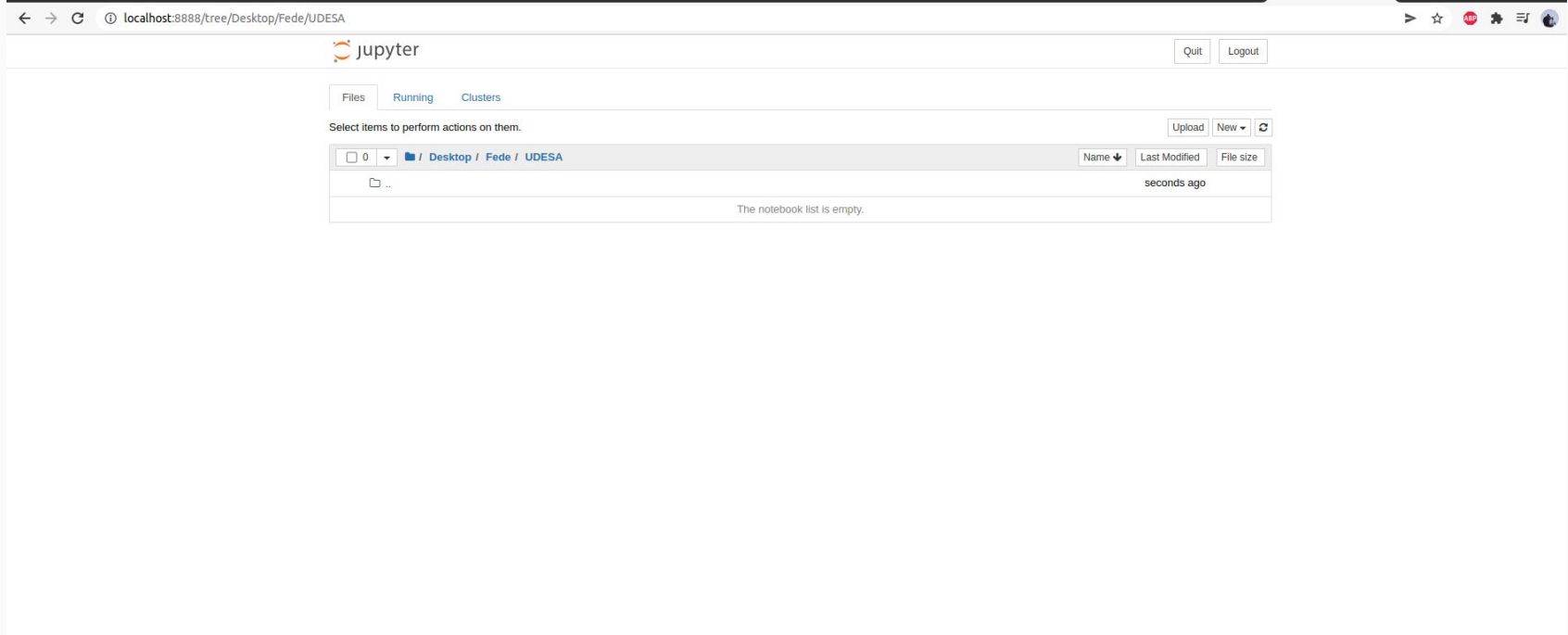
```
pip install jupyter
```

Corremos jupyter desde una consola/terminal

```
jupyter notebook
```

Link referencias: <https://jupyter.org/index.html>

Python: Uso, entorno, jupyter



Ambientes en Python

- Cuando desarrollamos, nos interesa que lo que hacemos sea replicable en otra máquina.
- Además, al tener Python tantas herramientas disponibles, es común instalar muchas librerías que pueden generar incompatibilidades.
- Hay varios “sabores” de ambientes virtuales para subsanar estos problemas.

**"YO TUVE
QUE INSTALAR
NUMPY, PANDAS
Y MATPLOTLIB"**

REQUIREMENTS.TXT

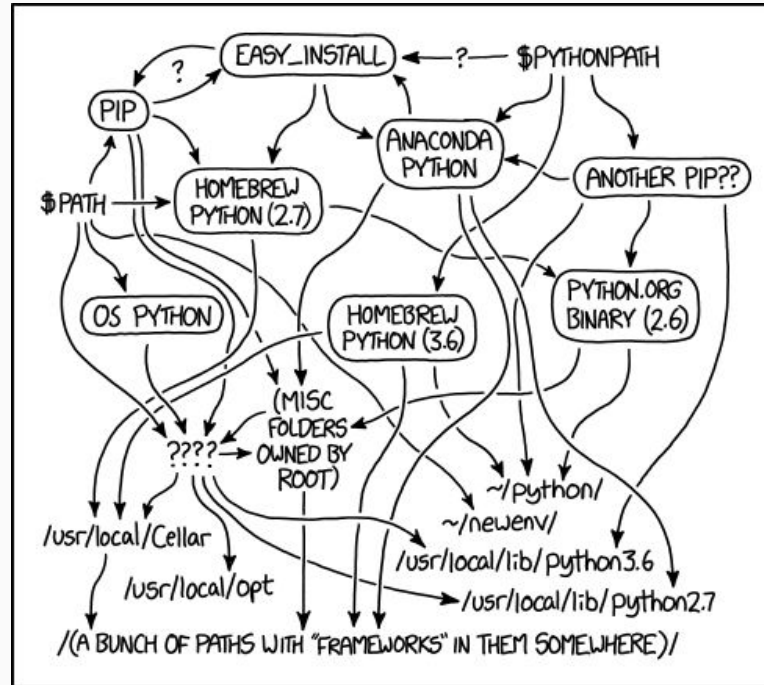
**VIRTUALENV/CONDA
+ PYENV**

**DOCKER
KUBERTENES AWS
GOOGLE CLOUD
AZURE GURU NINJA**

imgflip.com



Una clásica instalación



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

- La idea de un ambiente virtual es que las librerías que instalemos no se instalen a nivel global del sistema operativo, sino solamente dentro de un ambiente virtual controlado.
- Podemos tener muchos ambientes virtuales en una máquina, cada uno con sus librerías y sus respectivas versiones (que pueden ser todas distintas).
- Hay muchas versiones distintas de ambientes virtuales: virtualenv, virtualenvwrapper, conda.

- Un ejemplo de manejador de ambientes virtuales es el propio de python3.
- Puede encontrar la documentación oficial en <https://docs.python.org/3/library/venv.html>
- Se crea un nuevo ambiente corriendo *python3 -m venv PATH*.
- Para acceder al ambiente se corre con *source PATH/bin/activate*.
- Para salir del ambiente se corre *deactivate*.

- Otra herramienta muy utilizada es Conda. Es tanto un package manager (pip), como un manejador de versiones (virtualenv).
- Con *conda create* se crean ambientes.
- Con *conda activate* se ingresa al ambiente.
- Con *conda deactivate* se sale del ambiente.
- Con *conda install* se instalan librerías.

- En los ambientes es muy común tener un archivo `requirements.txt` que indica todas las librerías necesarias en dicho proyecto.
- Con `pip install -r requirements.txt` se pueden instalar todas las librerías necesarias juntas.
- Para armar el archivo de requerimientos se puede correr `pip freeze > requirements.txt`.
- El anterior comando primero lista todas las librerías actualmente instaladas y luego las vuelca a un archivo de texto.

- Además de las herramientas para ambientes virtuales, existe Pyenv que se encarga de que podamos utilizar diferentes versiones de python en una misma computadora.
- También existen pyenv-virtualenv y pyenv-virtualenvwrapper para trabajar conjuntamente.
- <https://github.com/pyenv/pyenv>
- <https://github.com/pyenv/pyenv-virtualenv>
- <https://github.com/pyenv/pyenv-virtualenvwrapper>

Demo de jupyter!

Demo de colab!

Python: Ejercicio 0

1. Instalar python, pip, jupyter
2. Correr jupyter
3. Crear un archivo primera_prueba.py (usar Text File)
4. Escribir un código que imprima en pantalla "Hola Mundo!"
5. Ejecutar el programa en una consola de Python3
6. Modificar el programa anterior para que luego de "Hola Mundo!" imprima en otra línea "Chau Mundo!"
7. Celebre el progreso :D

Python

Vamos a aprender la sintaxis de Python de a poco, hoy vamos a ver y ejercitar

- Indentación
- Asignación
- Comentarios
- Tipos de datos:
 - Numérico
 - String
 - Bool
 - Listas diccionarios

"PYTHON INDENTATION"

(CODE THAT WORKS)

```
n = [3, 5, 7]

def double_list(x):

    for i in range(0, len(x))
        x[i] = x[i] * 2
    return x

print double_list(n)
```

(CODE THAT FAILS)

```
n = [3, 5, 7]

def double_list(x):

    for i in range(0, len(x))
    x[i] = x[i] * 2
    return x

print double_list(n)
```



Python: Asignación

Operación para modificar el valor de una variable

Sintaxis:

`nombre_de_la_variable = expresion`

¿Qué hace?

1. Evalúa la expresión de la derecha
2. Se guarda en algún lugar de la memoria el resultado de la expresión que puede ser llamada y usada mediante el *nombre_de_la_variable*

Ejemplos:

```
1. x = 5
2. y = x
3. w = y+x+1
4. res = funcion1(4,5,w)
```

Python: Comentarios

Los comentarios son textos que no se ejecutan en el programa, sirven para hacer más claro el código. En el contexto de esta materia son **obligatorios**

Sintaxis:

esto es un comentario

¿Qué hace?

1. Nada, solo anota el código para que sea más entendible

Ejemplos:

```
1.  # Guardo un x en la variable x
2.  x = 5
3.
4.  # Guardo en w, la suma de las variable x mas un 3
5.  w = x+3
6.
7.  # Ejecuto la funcion1 con los parametros 4,5 y w, guardo este valor en la variable res
8.  res = funcion1(4,5,w)
```

Python: Tipos de datos

En Python todos los valores tienen un **tipo de dato asociado**. Las variables **no** porque funcionan como etiquetas de los valores

Python provee algunos tipos de datos y las librerías que incluiremos agregaran más, también podremos crear nuestros propios tipos de datos.

- Numeros: enteros (int), decimales (float)
- Texto: Se denotan entre comillas
- Valores de verdad: True o False
- Listas y diccionarios

Python: Tipos de datos / Números

En Python todos los valores tienen un **tipo de dato asociado**. Las variables **no** porque funcionan como etiquetas de los valores

Python provee algunos tipos de datos y las librerías que incluiremos agregaran más, también podremos crear nuestros propios tipos de datos.

- Números: enteros (int), decimales (float)

Programa	Output
<pre>1. <i># Creo dos variables a una Le asigno un entero y a otro un</i> <i>decimal</i> 2. un_entero = 4 3. un_decimal = 4.2 4. 5. <i># Printeo los valores</i> 6. print('El valor de un_entero es:', un_entero) 7. print('El valor de un_decimal es:', un_decimal) 8. <i># Printeo los tipos de las dos variables, uso la funcion type</i> 9. print('El tipo de la variable un_entero es:', type(un_entero)) 10. print('El tipo de la variable un_decimal es:', type(un_decimal))</pre>	<pre>El valor de un_entero es: 4 El valor de un_decimal es: 4.2 El tipo de la variable un_entero es: <class 'int'> El tipo de la variable un_decimal es: <class 'float'></pre>

Python: Operaciones sobre números

```
# Sumas (+)
```

```
4+9
```

13

```
# Restas (-)
```

```
9-2
```

7

```
# Multiplicacion (*)
```

```
4*2
```

8

```
# Division (/)
```

```
7/2
```

3.5

```
# Division entera (//)
```

```
7 // 2
```

3

```
# Resto o modulo (%)
```

```
7 % 2
```

1

```
# Potencia (**)
```

```
2**10
```

1024

Python: Tipos de datos / String o texto

- Texto (str) o strings: Se denotan entre comillas

Hay un montón de funciones sobre **strings**, por ejemplo, algunas:

- len: Cuenta la longitud
- .capitalize: Devuelve un string igual pero que empieza con mayúscula la primera letra
- .lower: Devuelve un string igual pero todo en minúscula
- + : toma dos string y devuelve uno nuevo concatenándolos

Hay un montón, googlearlas!

```
[7]: unString = 'esto es un string'

[8]: print('El tamaño de unString es:', len(unString))
El tamaño de unString es: 17

[9]: #Concateno mas texto
otroString = unString+', si!'

[10]: print(otroString)
esto es un string, si!

[11]: print(len(otroString))
22

[12]: print(unString.capitalize())
Esto es un string
```

- Valores de verdad: True o False

```
• # Estas 2 variables son de tipo texto (str)  
• esto_es_un_texto = "Hola!"  
• esto_tambien = 'Chau'  
•  
• # esto es una variable de tipo booleana (para modelar valor de verdad)  
• var_b = False  
• print('Imprimo valor y tipo de la variable var_b, valor:', var_b, ' | tipo:', type(var_b))  
•  
•  
• x = 3  
• print('Imprimo valor y tipo de la variable x, valor:', x, ' | tipo:', type(x))  
•  
• x = 'abcd'  
• print('Imprimo valor y tipo de la variable x, valor:', x, ' | tipo:', type(x))
```

Imprimo valor y tipo de la variable var_b, valor: False | tipo: <class 'bool'>

Imprimo valor y tipo de la variable x, valor: 3 | tipo: <class 'int'>

Imprimo valor y tipo de la variable x, valor: abcd | tipo: <class 'str'>

Python: Operaciones con resultados bool

```
# and (se cumplen ambas)
```

```
a = True
```

```
a and a
```

```
True
```

```
a and False
```

```
False
```

```
False and False
```

```
False
```

```
# or (se cumple alguna)
```

```
a = False
```

```
True or True
```

```
True
```

```
False or True
```

```
True
```

```
False or False
```

```
False
```

```
# not (niego)
```

```
not True
```

```
False
```

```
not False
```

```
True
```

Python: Operaciones con resultados bool

```
# == (igualdad)
```

```
3==3
```

```
True
```

```
3==2+1
```

```
True
```

```
"a"=="a"
```

```
True
```

```
4==3
```

```
False
```

```
# <, > (menor, mayor),  
# <=, >=, menor igual,  
# mayor igual  
3<4
```

```
True
```

```
3<3
```

```
False
```

```
3<=4
```

```
True
```

```
3<=3
```

```
True
```

Python: Tipos de datos / Listas

list en Python, son listas de elementos, tienen orden y pueden tener elementos de muchos tipos distintos dentro.

Sintaxis:

```
[ <elemento0>, <elemento1>, <elemento2>, ..., <elementoN> ]
```

Por ejemplo:

```
unaLista = [ 1, 2, 10]  
otraLista = [True, 1, False, 'hola', 'Chau']  
y_otraLista = [ [1] ,2 , 3, [4,5] ]
```

Existen diversas funciones sobre listas, algunas son:

- Concatenar (+): Toma dos listas y las une
- Longitud (len): Cuenta la cantidad de elementos
- I-esimo ([i]): Toma el i-ésimo elemento de la lista
- Agregar al final (.append): agrega un elemento a la lista

Python: Tipos de datos / Listas

```
In [115]: v1 = [2,4,1,40,True,'hola',3,4]
```

```
In [116]: v1[0] # es el primer elemento de la lista  
Out[116]: 2
```

```
In [117]: v2[-1] # el ultimo elemento del a lista  
Out[117]: 6
```

```
In [118]: v1 = [2,4,1,40,True,'hola',3,4]
```

```
In [119]: v1[0] # es el primer elemento de la lista  
Out[119]: 2
```

```
In [120]: v1[-1] # el ultimo elemento del a lista  
Out[120]: 4
```

```
In [121]: v1[-2] # el ante-ultimo elemento del a lista  
Out[121]: 3
```

```
In [122]: len(v1) # la longitud de la lista  
Out[122]: 8
```

```
In [123]: v1[1:3] # devuelvo el intervalo [1,3)  
Out[123]: [4, 1]
```


Python: Tipos de datos / Listas

list.append(x): Add an item to the end of the list; equivalent to `a[len(a):] = [x]`.

list.extend(L): Extend the list by appending all the items in the given list; equivalent to `a[len(a):] = L`.

list.insert(i, x): Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

list.remove(x): Remove the first item from the list whose value is `x`. It is an error if there is no such item.

list.pop([i]): Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list.

list.index(x):

Return the index in the list of the first item whose value is `x`. It is an error if there is no such item.

list.count(x):

Return the number of times `x` appears in the list.

list.sort(cmp=None, key=None, reverse=False): Sort the items of the list in place (the arguments can be used for sort customization, see `sorted()` for their explanation).

list.reverse(): Reverse the elements of the list, in place.

Mas: <https://docs.python.org/3/tutorial/datastructures.html>

Python: Tipos de datos / Diccionarios

Los diccionarios son estructuras de datos que almacenan relaciones. Funciona como un diccionario común. Se tienen claves y valores.

Los valores pueden ser de cualquier tipo, las claves tienen que tener ciertas propiedades.

Las claves siempre son únicas! (no puede haber dos definiciones distintas para la misma key)

Sintaxis:

- `un_dic = {key1:value1, key2:value2,...}`

Python: Tipos de datos / Diccionarios

```
In [132]: un_dic = {} # esto es un diccionario vacio
```

```
In [133]: un_dic2 = { 'Juan':1, 'Maria':2, 'Laura':3} # esto es un dic inicializado con datos
```

```
In [134]: un_dic2['Juan'] # busco en el diccionario por la clave Juan y obtengo su valor
```

```
Out[134]: 1
```

```
In [135]: un_dic2['Facundo'] # si la clave no esta en el diccionario obtengo un error
```

```
-----  
KeyError                                Traceback (most recent call last)
```

```
<ipython-input-135-b649f7338f79> in <module>
```

```
----> 1 un_dic2['Facundo'] # si la clave no esta en el diccionario obtengo un error
```

```
KeyError: 'Facundo'
```

```
In [136]: un_dic2
```

```
Out[136]: {'Juan': 1, 'Maria': 2, 'Laura': 3}
```

```
In [137]: un_dic2['Facundo'] = 'Hola!'
```

```
In [138]: un_dic2
```

```
Out[138]: {'Juan': 1, 'Maria': 2, 'Laura': 3, 'Facundo': 'Hola!'}
```

Python: Tipos de datos / Diccionarios

```
In [139]: un_dic2['Facundo'] # Ahora si deberia andar!
```

```
Out[139]: 'Hola!'
```

```
In [140]: un_dic2['Facundo'] = 'Chau' # Aca estoy pisando el valor la clave ya definida
```

```
In [141]: un_dic2
```

```
Out[141]: {'Juan': 1, 'Maria': 2, 'Laura': 3, 'Facundo': 'Chau'}
```

```
In [142]: del un_dic2['Facundo'] # Esto borra la entrada Facundo
```

```
In [143]: un_dic2
```

```
Out[143]: {'Juan': 1, 'Maria': 2, 'Laura': 3}
```

```
In [144]: un_dic2.clear() # Esto borra todo el diccionario
```

```
In [145]: un_dic2
```

```
Out[145]: {}
```

Pista: del tambien funciona para listas!

```
In [150]: l = [5,4,9]
```

```
In [151]: del l[1] # Borro el elemento en la posición 1
```

```
In [152]: l
```

```
Out[152]: [5,9]
```

Ya pueden terminar la práctica I

[Link Colab Guía 1](#)