

# INTERMEDIATE PYTHON FOR DATA SCIENCE

Nicolas Paris  
CTO, Healint

---

# HELLO!

---

## NICOLAS PARIS



- ▶ 2005-2013 Business Intelligence / Finance
- ▶ 2014 Head of Data at Healint [python]
- ▶ 2017 CTO at Healint [python, AWS]

Healint:

- >> 300.000 Monthly Active users on the platform
- >> 500 GB of data per week

---

# **INTERMEDIATE PYTHON FOR DATA SCIENCE**

---

## **GETTING TO KNOW YOU**

**<http://bit.ly/2YiuG6J>**

---

# AGENDA

---

## I. PANDAS

- ▶ 1. Overview / Pandas workflow
- ▶ 2. Filtering
- ▶ 3. Transformations: apply
- ▶ 4. Aggregations: Group By
- ▶ 5. Comments on data visualisation
- ▶ 6. Understanding vectorized operations
- ▶ 7. When to use pandas vs other tools
- ▶ 8. Hands On

## II. PERFORMANCE TIPS

- ▶ Why/When
- ▶ Non technical tips

## III. ML PIPELINE

- ▶ Why/When
- ▶ Benefits
- ▶ Examples
- ▶ Implementation

---

INTERMEDIATE PYTHON FOR DATA

---

# GETTING STARTED

---

## **INTERMEDIATE PYTHON FOR DATA**

---

### **CONNECT TO WIFI**

1. WIFI: GA / Password: yellowpencil

### **CLONE THE FOLLOWING GITHUB REPO**

<https://github.com/niparis/ga-intermediate-python-ds>

---

## **INTERMEDIATE PYTHON FOR DATA**

---

# **START A PYTHON NOTEBOOK - PYTHON 3!**

- Option A: You already have it !
- Option B: Use google colab ([colab.research.google.com](https://colab.research.google.com))
- Option C: Download Anaconda (<https://www.anaconda.com/distribution/>)

---

INTERMEDIATE PYTHON FOR DATA

---

# PANDAS

---

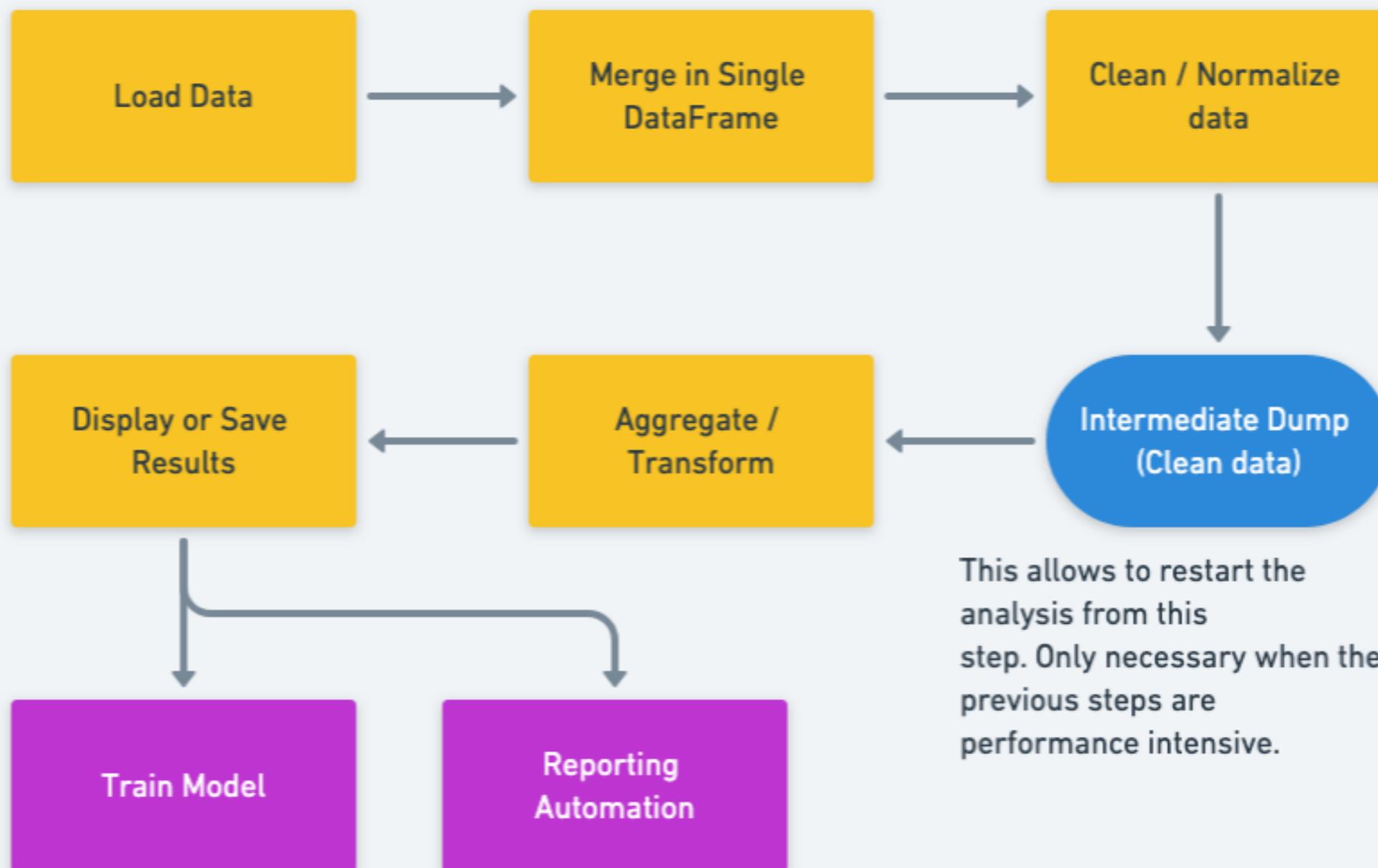
## PANDAS: A DATA TOOLBOX

---

Pandas is a python library that has become the de-facto standard for:

1. Merging datasets (especially when coming from different sources)
2. Cleaning data, which involves
  - A. Filtering to remove irrelevant data
  - B. Impute missing data
3. Creating aggregations
4. Creating new features
5. Generating data ready to be used by a model

# Pandas Workflow



---

## PANDAS: A DATA TOOLBOX

---

Pandas is a huge library - it's impossible to learn everything. Besides, the doc is great, so why learn everything ?

**However there are some fundamental building blocks of pandas that you have to know to be productive:**

**DataFrame [object]:** *The main pandas object. Represents a table. Think of it as an excel table.*

**.Loc / .Query:** *Each of these allows to filter dataframes. They have different syntax, the `.`Query` being the more modern approach.*

**.Apply:** *To transform some existing column (or create new columns from an existing one)*

**.GroupBy / .agg:** *The core of the pandas workload. Allows to do any aggregation.*

---

**PANDAS: A DATA TOOLBOX**

---

# **FILTERING**

---

## PANDAS ESSENTIALS: FILTERING

---

**WHY:** To keep only part of the data, either to :

**HOW:** Use the ` `.query` function. Here's the basic syntax:

*<dataframe>.query("column operation value")*

**NOTE:** "column operation value" is a STRING.

**EXAMPLES:**

**A. df.query("column > 4")**

Filters any row where the value of column is greater than 4

**B. df.query("column > second-column")**

Filters any row where the value of column is greater is greater than the value in second-column

---

## PANDAS ESSENTIALS: FILTERING WITH .LOC [1]

---

**NEXT:** Once you're comfortable with `.query`, learn `.loc`.

Benefits of `.loc`:

- A. Can add /change a value in the filtered expression
- B. complex filtering logic

---

## PANDAS ESSENTIALS: FILTERING WITH .LOC [2]

---

### A. Add /change a value in the filtered expression with .loc

Example:

```
df.loc[df["column_to_filter"] == "criteria", "cats"] = "meow"
```

Explanation:

For all rows where the value in the column “column\_to\_filter” is strictly equal to “criteria”, ASSIGN the value “meow” in the column “cats”

Note:

If the column “cats” does not exist, it will be created on the fly.  
Any value not filtered will become “None”

---

## PANDAS ESSENTIALS: FILTERING WITH .LOC [3]

---

### B. Can do more complex filtering logic (multi criterias, mixing AND / OR)

Example:

```
df.loc[(df["column_to_filter"] == "criteria") | (df["second_criteria"] > 4)]
```

Explanation:

Filters any row having **either** “criteria” as the value in the column “column\_to\_filter”  
OR strictly more than 4 as the value in the column “second\_criteria”

Note: pipe ( | ) is used for logical OR, ampersand ( & ) is used for logical ANDS

---

**PANDAS: A DATA TOOLBOX**

---

# **TRANSFORMING**

# PANDAS: TRANSFORMATIONS [1]

**WHY:** To transform some existing column, or create a new column from an existing one

**HOW:** Use the ` `.apply` function. Here's the basic syntax:

*<dataframe>[<column-selection>].apply(<function>)*

The DataFrame object + the column selection. It returns a single column (pd.Series). The transformation will be applied on *\*each element\** of the column

The .apply function

A function that will be applied on EACH ELEMENT of the source object

Legend

- Source Object
- Function / Method
- Argument

**The main advantage of ` `.apply` is its versatility. Pandas**

# PANDAS: TRANSFORMATIONS [2]

## EXAMPLE:

```
def transform_to_date(elem: dt.datetime):
    # extract the date from a datetime.
    # date is a method of the datetime object, that's why it has to be called with `()`
    return elem.date()
```

```
df['date'] = df['utc'].apply(transform_to_date)
df.head(2)
```

|   | username  | subreddit        | utc                 | date       |
|---|-----------|------------------|---------------------|------------|
| 0 | kabanossi | photoshopbattles | 2016-12-26 10:24:59 | 2016-12-26 |
| 1 | kabanossi | GetMotivated     | 2016-12-26 10:23:14 | 2016-12-26 |

---

PANDAS: A DATA TOOLBOX

---

# AGGREGATING

# PANDAS: AGGREGATIONS [1]

Legend

**WHY:** To transform some existing column, or create a new column from an existing one

**HOW:** Use the ` `.apply` function. Here's the basic syntax:

```
<dataframe>.groupby(<column_to_reduce>).agg({“target_column”: “function”})
```

Source DataFrame

The .groupby function

A column (or list of columns). The dataframe will be reduced until only UNIQUE ELEMENTS of that column are left.

The .agg function

Which column we want to perform computation on. Ex: if we perform an average, this will define the values we average.

This function will be applied on each element of “target\_column”, and generate a single value for each unique element of “column\_to\_reduce”



Source Object



Function / Method



Argument

---

## PANDAS: GROUP BY [SQL EQUIVALENT]

---

**PANDAS**

**Df.groupby("country")["id"].count()**

**OR**

**Df.groupby("country").agg({"id": "count"})**

**SQL**

**SELECT country, count(id)  
FROM table  
GROUP BY country**

---

PANDAS: A DATA TOOLBOX

---

# HANDS ON

OPEN THE NOTEBOOK **INTRO-TO-PANDAS.IPYNB**

FROM THE FOLDER **“01 - PANDAS”** (IN THE REPO YOU  
DOWNLOADED)

---

**PANDAS: A DATA TOOLBOX**

---

# **MORE PANDAS**

---

## PANDAS: VECTORIZED OPERATIONS

---

Pandas is fast because it executes its operations on “vectors”

Vectorized operations are operations performed on entire columns at once.

To keep pandas fast avoid looping through rows.

It's a pandas anti-pattern, it will be much SLOWER than standard python

---

## PANDAS: HOW TO THINK ABOUT DATA VIZ

---

It's easy to spend hours creating very nifty charts in python.  
They do look cool, but I'd suggest not investing too much time in  
learning how to create charts:

1. It can be VERY time consuming to create charts if you want to control the look in depth
2. The code is typically not very reusable
3. You're not improving your coding skills.
4. Many charts can be done easily with GUI

***Don't spend more time creating charts than working the data***

---

# PANDAS: HOW TO THINK ABOUT DATA VIZ

---

***You're working on an EDA (exploratory data analysis)***

Use pandas “.plot” methods, as it takes no effort. Install the “seaborn” library to prevent your eyes from bleeding

***You're working on a deliverable (slides, paper, etc..)***

For a business audience, consider Tableau / Excel, unless there is a strong need to give a “sciency” vibe.

***You're automating a report (charts need to be generated automatically)***

Code is necessary. No choice.

***You're working on a dashboard / web application that displays charts***

Consider hosted dashboards (hosted tableau, mode, metabase, holistics, ...). If your needs are too specific, have a look at plotly.

---

## PANDAS: WHEN TO USE OTHER TOOLS

---

### ***Excel***

Yes, it's not code. But if you have some quick calculations to do on a small file, or if you need a very dynamic interaction with your data (ie: lots of filtering), don't hesitate to save your dataframe as an excel file.

### ***Databases***

A lot of pandas operations can be done by a database (merging, filters, aggregations). Many of them will be faster on a database, so learn SQL, and extract just the data you need from the database (rather than complete tables)

# PANDAS: MOST IMPORTANT DOC REFERENCE

---

[Group By: split-apply-combine](#)

[Indexing and selecting data](#)

[Tom Augspurger: Modern Pandas](#)

Most important part of  
the pandas doc

The best  
pandas  
tutorial

---

INTERMEDIATE PYTHON FOR DATA

---

# PERFORMANCE

---

## INTERMEDIATE PYTHON FOR DATA

---

**ALWAYS OPTIMIZE  
YOUR TIME !!!**

---

# INTERMEDIATE PYTHON FOR DATA

---

## WHEN PERFORMANCE MATTERS

- You can perceive the slowness and it can be fixed faster than the time it wastes for you
- You understand how to fix it
- It's preventing from getting results quickly (waiting hours to get the model ready)

## WHEN PERFORMANCE DOES NOT MATTERS

- “I can make this 3 x faster” >> pride
- It just saves machine time (worth much less than dev time)

---

## PERFORMANCE PRACTICAL TIPS

---

- ▶ Use a very small subset of your dataset when you're bug fixing (ie: still figuring some syntax, not yet reasoning about the data itself)
- ▶ Add timers to your (slow) loops, so you know when to expect your code to complete (do you need to wait 20 minutes or 20 days ?)
- ▶ When you're working with an algo that can use multiple CPU, always double check CPU usage on your machine to make sure that it's actually using them

*These tips may look obvious, but I consistently see data scientists and analysts losing hours because they don't respect them.*

---

INTERMEDIATE PYTHON FOR DATA

---

# MACHINE LEARNING PIPELINE

---

## **WHAT IS A PIPELINE AND WHY WE NEED IT**

---

- ▶ A pipeline is a set of automated processes that allows to automatically execute tasks.
- ▶ When starting on an ML model, most data scientists start by their code in a notebook, cell by cell, without functions.
- ▶ This works perfectly for EDA (Exploratory Data Analysis), but is not an ideal way to build a machine learning model.

---

## KEY BENEFITS OF BUILDING MODELS AS PIPELINES [1/2]

---

- ▶ 1. Less bugs: by compartmenting your code into `functions`, each `function` will take care of a single part of the pipeline, making bugs discovery/fixing much easier
- ▶ 2. Readability: by breaking your code in small pieces, it becomes more readable, easier to document and share - essential if you're working in a team
- ▶ 3. Flexibility: much easier to adjust parts of the model without risking breaking / impacting the rest
- ▶ 4. Reproducibility: it's quite easy to produce unstable code in notebooks (execution order of cells is not fixed, easy to shadow variables)

---

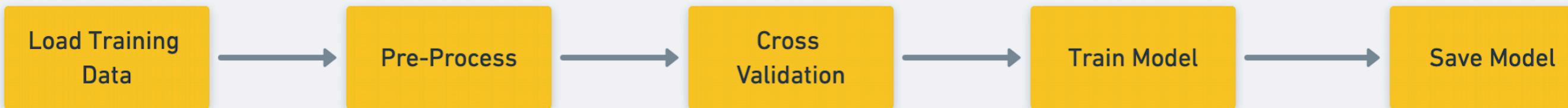
## A PRACTICAL IMPLEMENTATION

---

- ▶ A PIPELINE IS:
- ▶ A SERIE OF FUNCTION
- ▶ WHERE EACH FUNCTION
- ▶ USES THE RESULT OF THE PREVIOUS FUNCTION AS ITS INPUT

# EXAMPLE OF PIPELINES

## TRAIN



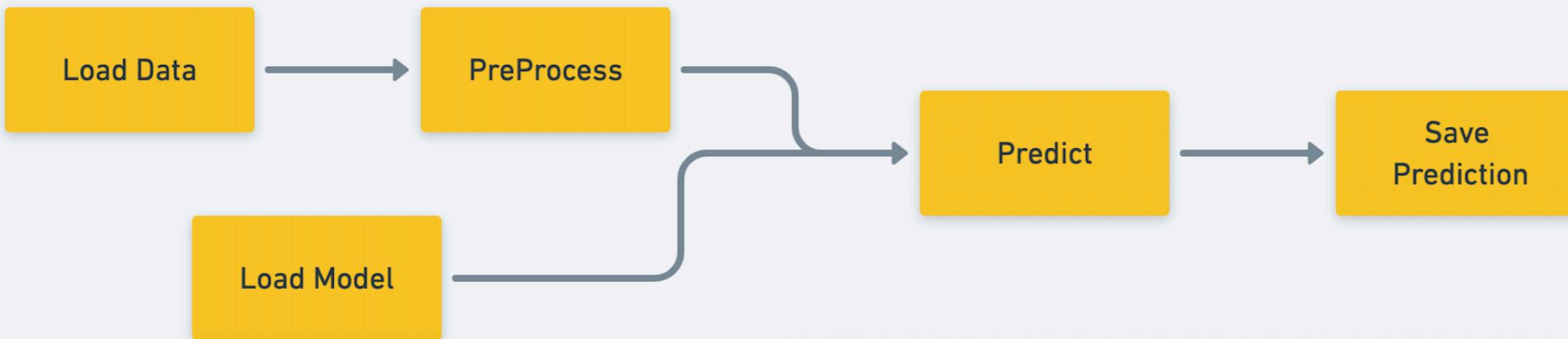
## EXAMPLE OF PIPELINES

### HYPER PARAMETERS OPTIMIZATION



# EXAMPLE OF PIPELINES

## PREDICT



---

## NOTEBOOK EXAMPLE

---

Open the notebook **ml-pipeline.ipynb**

From the folder **“02 - ml pipeline”** (in the repo you downloaded)

---

INTRODUCTION TO PYTHON

---

# WHAT'S NEXT

---

# INSTALLING PYTHON

---

## WINDOWS: ANACONDA

Download Anaconda

<https://www.anaconda.com/distribution/>

## MAC OS: BREW

On Mac, use Anaconda if you plan to use python ONLY for data science. Else use Brew, it's extra effort but will save you trouble.

<https://docs.python-guide.org/starting/install3/osx/>

---

## **KEEP LEARNING - PRACTICE**

---

- I. Work with the other notebooks
  - Flights Delays & Youtube
  - They include some examples and some exercises
- II. Download some data and perform your own notebooks
  - <https://www.kaggle.com/datasets>
  - Data from your job

**PRACTICE UNTIL YOU CAN USE  
THE SYNTAX SEEN IN THE INTRO NOTEBOOK  
WITHOUT TOO MUCH EFFORT**

---

## **KEEP LEARNING - MEET**

---

- ▶ **MEETUPS:**
  - ▶ PyData
  - ▶ Python User groups
  - ▶ DataScience SG
  
- ▶ **PERSONAL PROJECTS**

---

## INTRODUCTION TO PYTHON

---

# Q&A



[ni.paris@gmail.com](mailto:ni.paris@gmail.com)