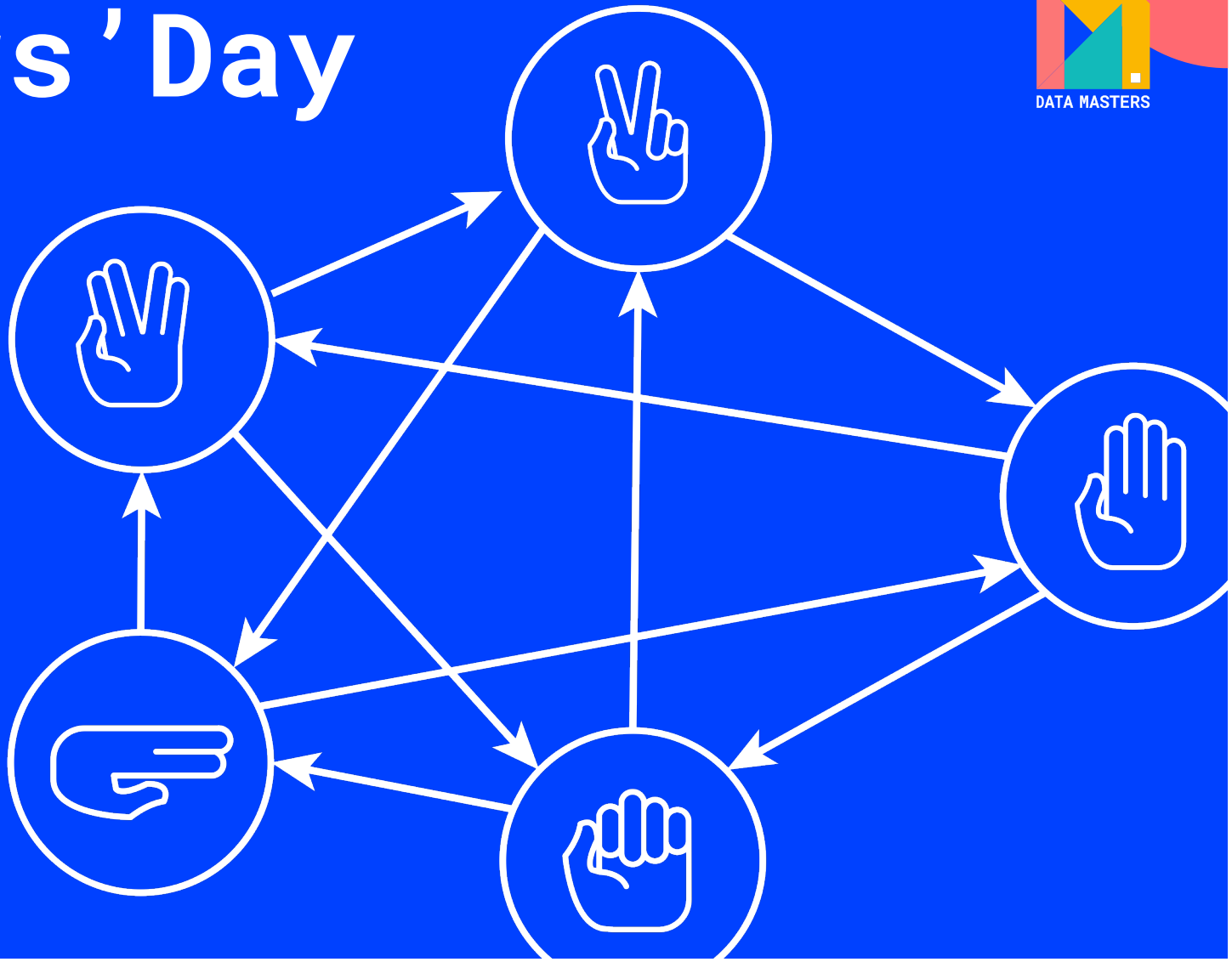
Beginners' Day



DATA MASTERS

PYCON(IT) 23

ABOUT ME

DATA MASTERS

- **Ingegnere informatico**
- **Front End developer** @ Frankhood dal 2011
- **Insegnante di informatica** @ ITT Panetti Pitagora
- **Machine Learning specialist** @ Datamasters dal 2020
- Linguaggi di sviluppo preferiti:

Python **Javascript** **julia**

PYCON(IT) 23 Beginners'Day

Mission

DATA MASTERS

Datamasters.it consente ai professionisti italiani di avanzare nella propria carriera, migliorare la propria comunità e cambiare il mondo condividendo e applicando le competenze più avanzate in ambito **Intelligenza Artificiale**, **Machine Learning** e **Data Science**.

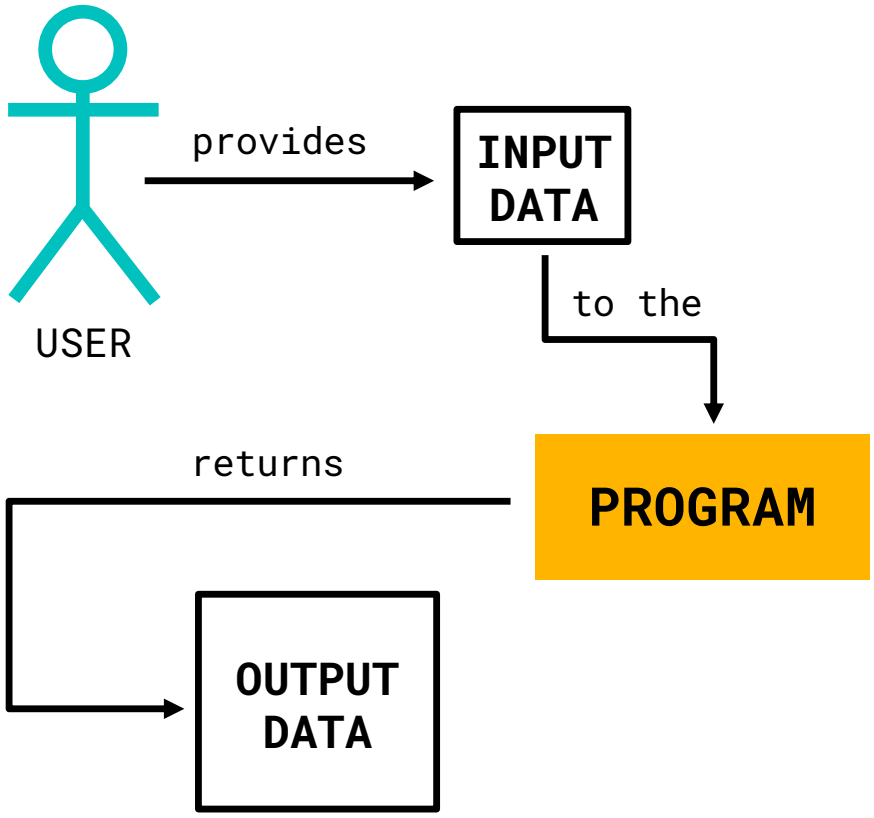
L'obiettivo è quello di forgiare la **nuova generazione di data scientist** e **machine learning engineer italiani**, una pipeline di talenti in grado di rispondere rapidamente all'enorme domanda di competenze specialistiche che investirà il mercato del lavoro.

INTRODUZIONE

DATA MASTERS

Tipicamente, un programma:

- **Chiede un input** all'utente
- **Esegue delle istruzioni** (tipicamente fa dei calcoli)
- **Ritorna** un valore in output



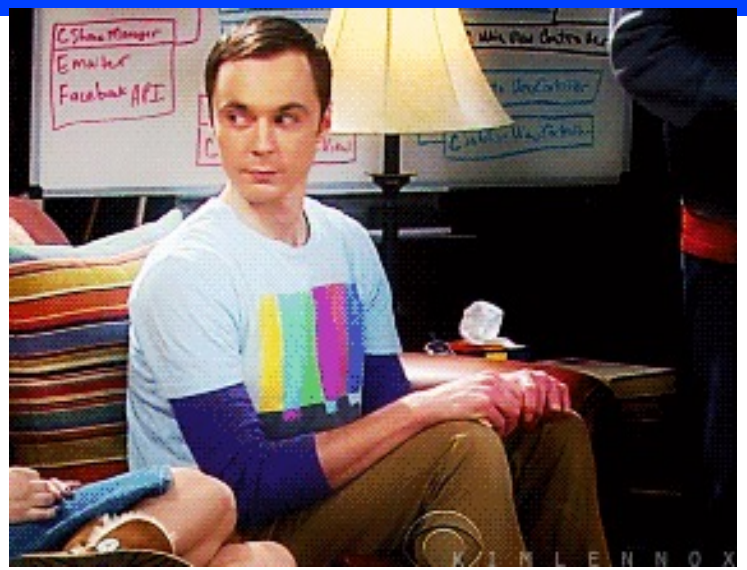
COSA FAREMO

DATA MASTERS

01 **Svilupperemo da zero** una nostra versione di Sasso-Carta-Forbice-Lizard-Spock

02 Partiremo dal classico **rock-paper-scissors**

03 Arriveremo ad **utilizzare un modello** di **machine learning** in grado di riconoscere le nostre mosse dalla webcam



PYCON(IT) 23 Beginners'Day

COSA FAREMO

DATA MASTERS

La domanda è:

COME LO FAREMO?

PYCON(IT) 23 Beginners'Day

PROBLEM

DATA MASTERS

Il problema è: “Trova un modo per mostrare i risultati del gioco di una manche di Sasso-Carta-Forbice di **un umano contro il computer**”

C’è molta ambiguità in questa affermazione. L’ambiguità è **un male!**

Cerchiamo di essere **più precisi**:

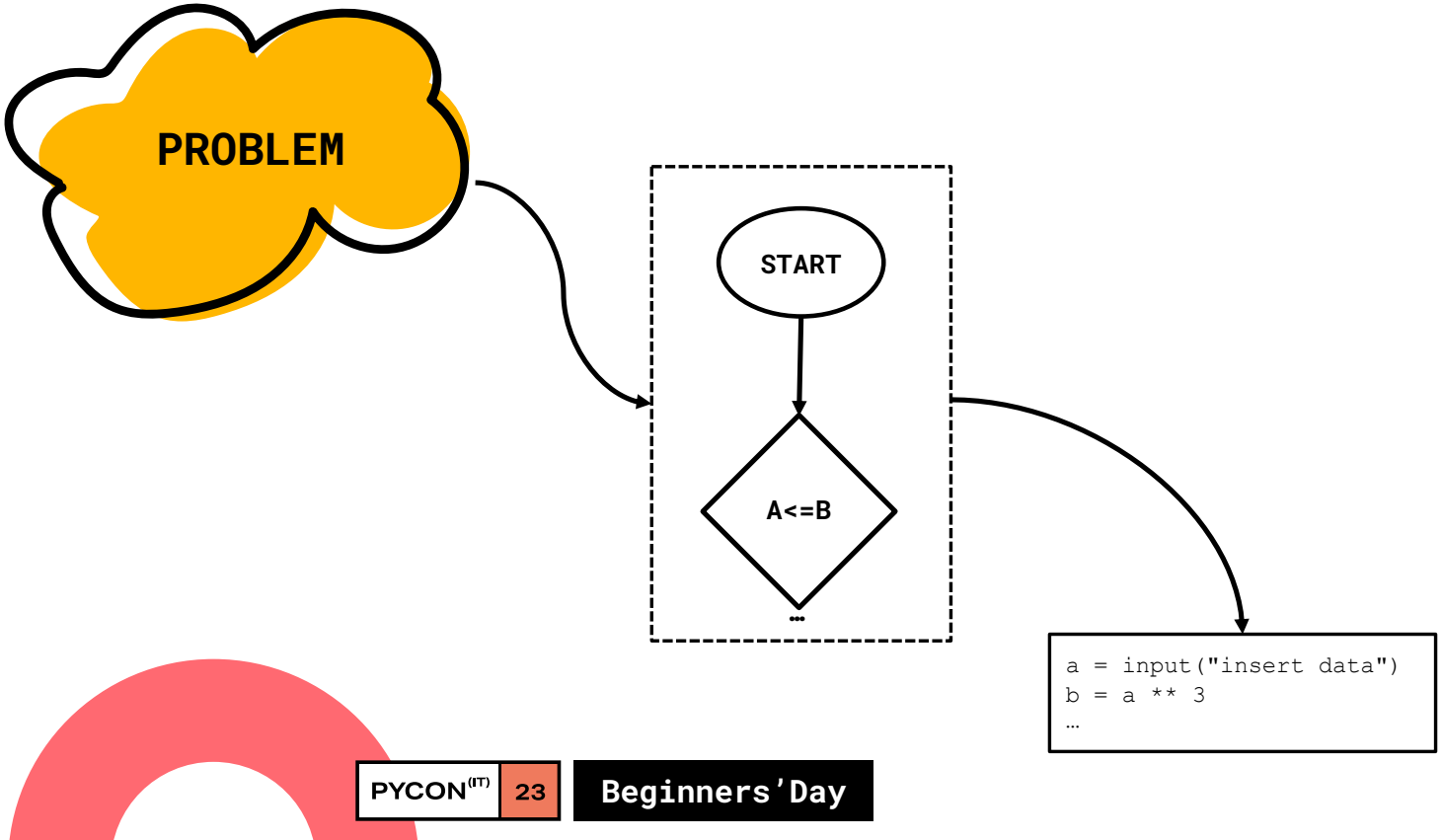
- Un software lavora con dei dati di input, li elabora in qualche modo e fornisce un output
- Come possiamo far giocare un umano contro il computer?
- Che tipo di dati dovremmo fornire in input al nostro programma?

Le istruzioni che risolvono il nostro problema sono le parti di un algoritmo.

Un algoritmo è una **sequenza di istruzioni molto semplice** per risolvere un problema.

DAL PROBLEMA AL CODICE
attraverso l'algoritmo

DATA MASTERS



High Level
(high abstraction)

Low Level
(low abstraction)

PYCON(IT) 23 Beginners'Day

ALGORITHM

L’algoritmo è **il cuore** del nostro programma

I dati di input rappresentano **la scelta dell’utente sotto forma di stringa**:

- Sasso
- Carta
- Forbice
- Lizard
- Spock

ALGORITMO


Carta
Sasso
Forbici
Lizard
Spock
Algoritmo
(il computer fa la sua mossa)
Hai vinto!
Hai vinto
il computer

COME PROCEDERE

DATA MASTERS

Suddividiamo un problema **complesso** (macro problem) in tanti problem **semplici**(tasks)

Macro-problema: a partire da un input preso dall'utente il programma deve far giocare il computer e stabilire chi ha vinto.

COSA FARÀ IL NOSTRO ALGORITMO?	• Far fare la mossa al computer (possibilmente senza barare ^__^) • Verificare chi ha vinto

Per fare ciò **l'algoritmo deve**:

- Creare una scelta completamente randomica e "al buio" che rappresenti la scelta del computer
- Confrontare in qualche modo le due scelte
- Restituire il vincitore della singola manche
- Generalizzazione: fare una partita fatta da N manche

COME PROCEDERE

DATA MASTERS

Task 1 | Prendere in input la scelta dell'utente, memorizzarla e fare la mossa del computer

Task 2 | Confrontare le mosse per stabilire un vincitore

Task 3 | Migliorare il gioco

Task 4 | Usare il Machine Learning

TASK 1

DATA MASTERS

Let's code!

TASK

Prendere in input la scelta dell'utente, memorizzarla e fare la mossa del computer

TASK 2

DATA MASTERS

È qui che dobbiamo confrontare le mosse per vedere chi ha vinto:

Cosa abbiamo a disposizione?	Due stringhe

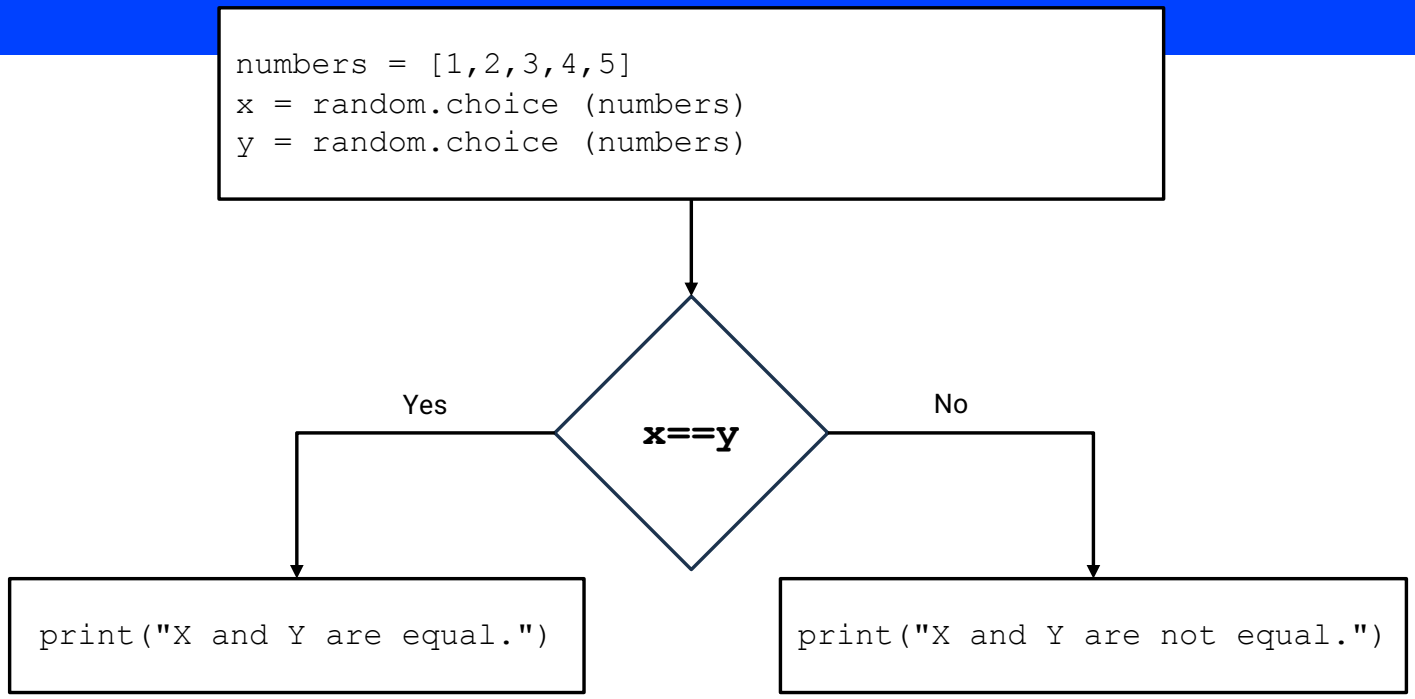
Che significa confrontare due stringhe (due parole)?

ALTRA DOMANDA → è possibile *far durare di più il gioco?*

TASK 2

DATA MASTERS

COSTRUTTO
if/else

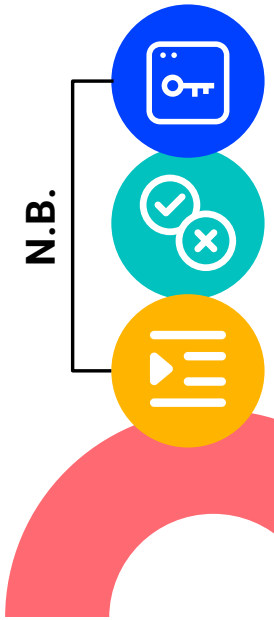


TASK 2

DATA MASTERS

COSTRUTTO
if/else

```
if espressione:
    do_something()
else:
    do_something_else()
```



N.B.

`if` ed `else` sono **parole chiave** di Python

`espressione` è una **variabile booleana** *(può essere solo True o False)*

Occhio **all'indentazione**

TASK 2

DATA MASTERS

OPERATORI DI
confronto

Operatore	Domanda a cui risponde
`a > b`	Il valore di a è **maggiore stretto** del valore di b?
`a < b`	Il valore di a è **minore stretto** del valore di b?
`a <= b`	Il valore di a è **minore o uguale** al valore di b?
`a >= b`	Il valore di a è **maggiore o uguale** al valore di b?
`a == b`	Il valore di a è **uguale** al valore di b?
`a != b`	Il valore di a è **diverso** dal valore di b?

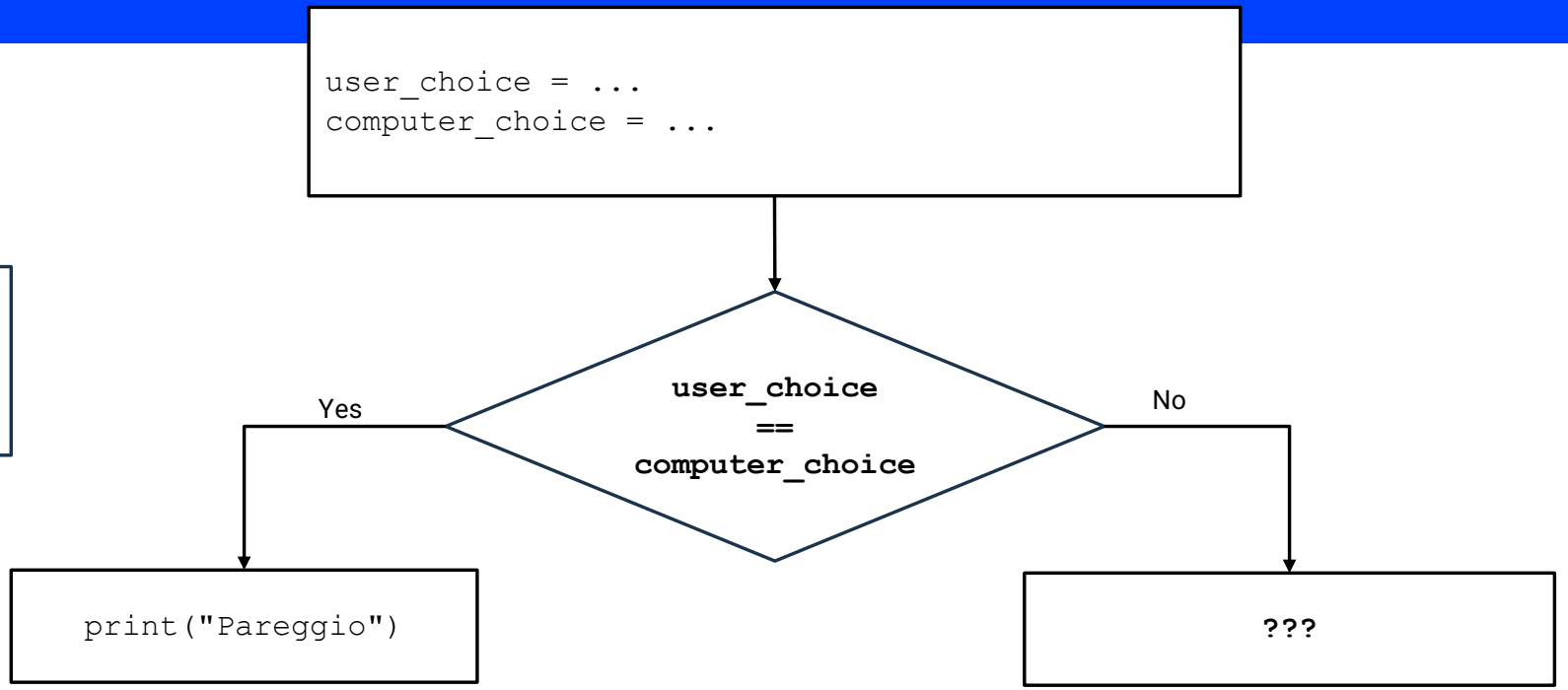
N.B.
I valori contenuti in `a` e `b` possono essere di **qualsiasi tipo**

Posso confrontare fra loro valori numerici, **stringhe**, altri booleani, etc.

TASK 2

DATA MASTERS

COSTRUTTO
if/else



TASK 2

DATA MASTERS

COSTRUTTO
if/else

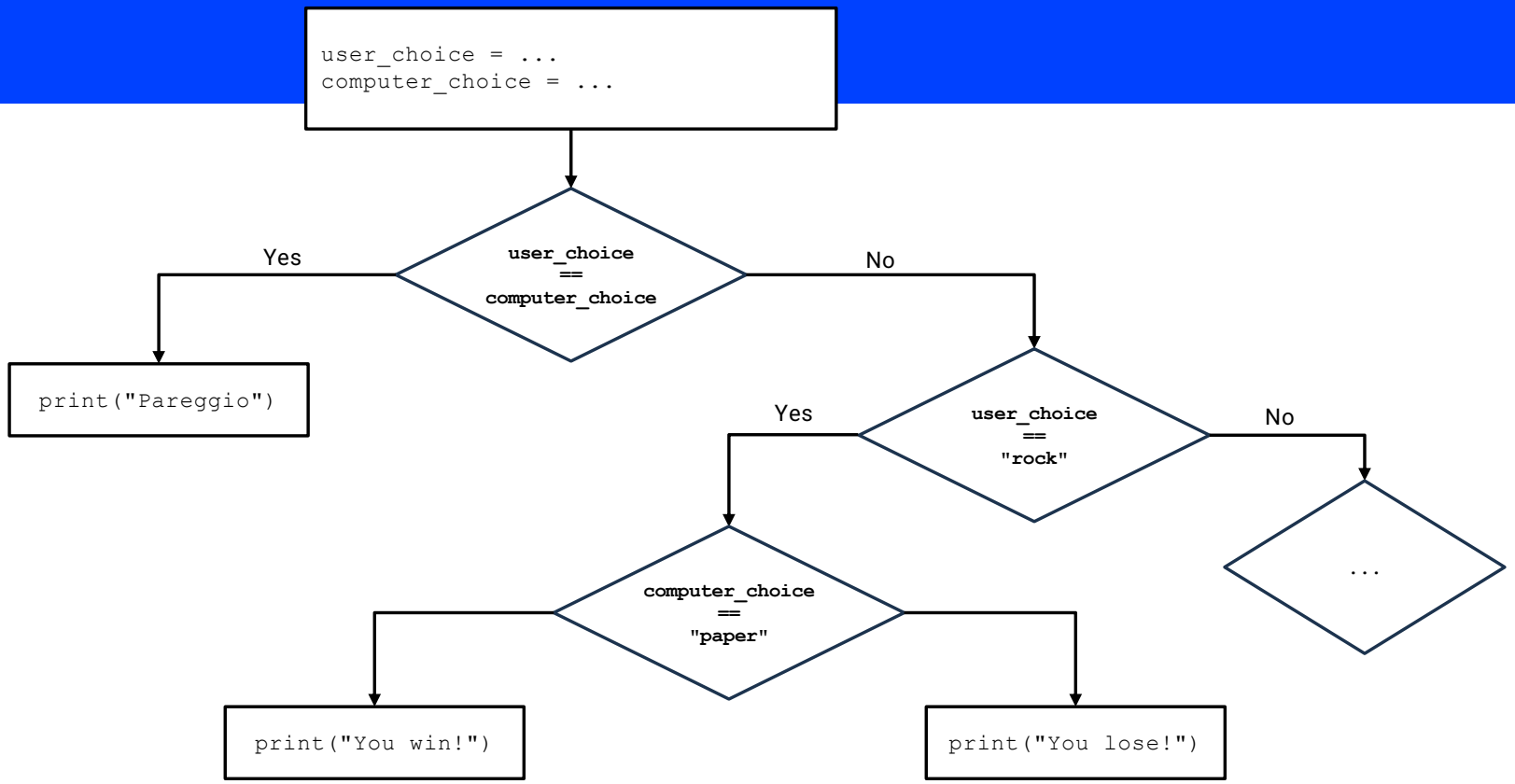
```
if espressione:
    do_something()
elif:
    do_something_instead()
elif:
    …
else:
    do_something_else()
```

Un semplice if/else **non ci basta**, ci sono altre condizioni da verificare.

- Se il giocatore e il computer hanno scelto la stessa mossa: **PAREGGIO**
- **Altrimenti se** il giocatore ha scelto **sasso**:
  - Se il computer ha scelto **carta**: vince il computer
  - Se il computer ha scelto **forbici**: vince il giocatore
- **Altrimenti se** il giocatore ha scelto **carta**:
  - Se il computer ha scelto **forbici**: vince il **computer**
  - Se il computer ha scelto sasso: vince il **giocatore**
- **Altrimenti se** il giocatore ha scelto **forbici**:
  - Se il computer ha scelto **sasso**: vince il **computer**
  - Se il computer ha scelto **carta**: vince il **giocatore**

TASK 2

DATA MASTERS



TASK 2

DATA MASTERS

Task 1 ✓ Prendere in input la scelta dell'utente, memorizzarla e fare la mossa del computer

Task 2 **Confrontare le mosse per stabilire un vincitore**

Task 3 Migliorare il gioco

Task 4 Usare il Machine Learning

TASK 2

DATA MASTERS

Let's code!

TASK

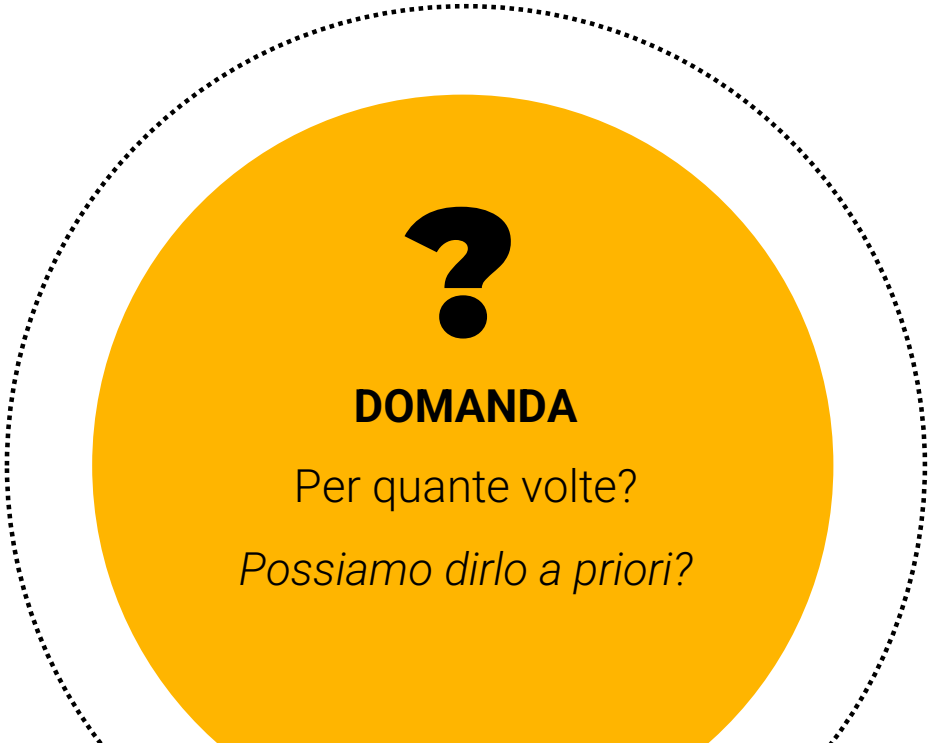
Confrontare le mosse per stabilire un vincitore

TASK 2
Improvement

DATA MASTERS

È possibile far durare una partita **più di una singola manche di gioco?**

- Una singola manche di gioco è fatta da:
  - Prendi l'input dell'utente
  - Genera la mossa del computer
  - Confronta le mosse per capire chi ha vinto
  - Stampa a schermo il vincitore
- La chiave è pensare di **ripetere** questo flusso

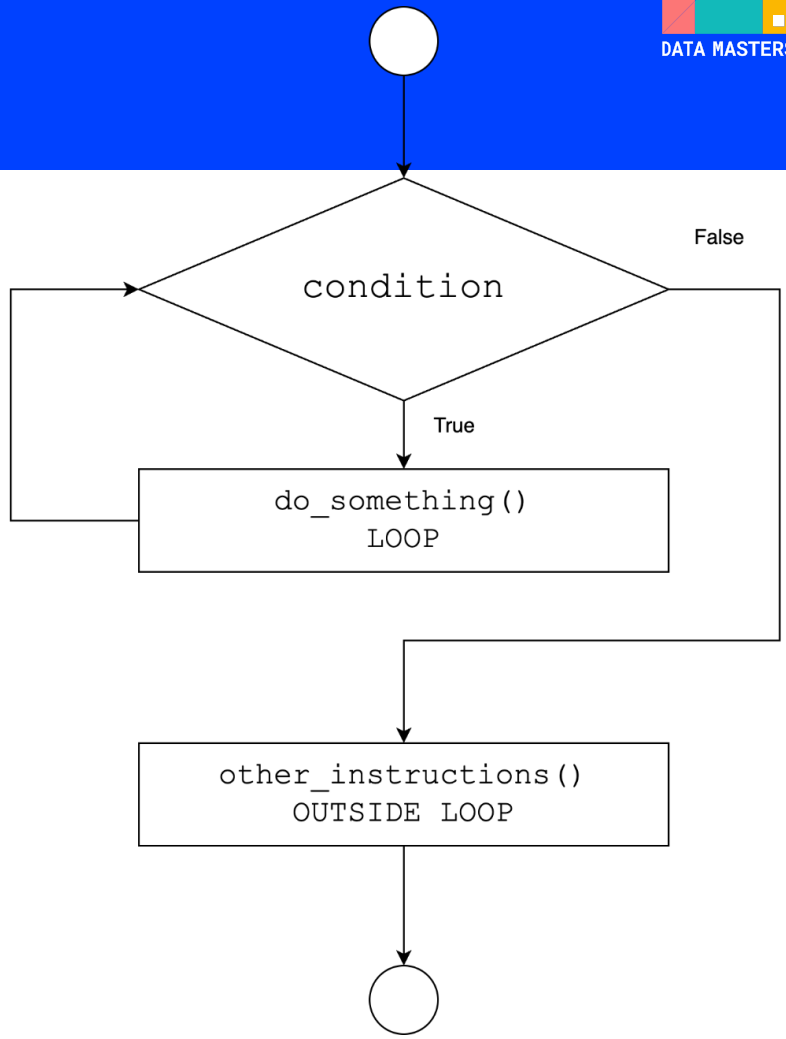


TASK 2

DATA MASTERS

COSTRUTTO
ciclo while

```
while condition:
    do_something()
```



- `while` è una parola chiave del linguaggio
- viene eseguito a loop tutto ciò che è indentato
- quello che è allo stesso livello di indentazione del `while` è fuori dal loop
- `condition` è un valore booleano
- il ciclo termina quando:
  - `condition` assume un valore False
  - viene eseguita un'istruzione `break`

TASK 2
Improvement

DATA MASTERS

TASK 2a

Let's code!

Ripetere la manche di gioco per un numero indefinito di volte

Hint: ad ogni manche chiediamo all'utente se vuolecontinuare a giocare

TASK 3
Miglioriamo il gioco

DATA MASTERS

Abbiamo una versione base, ma possiamo fare di più, **molto di più!**

Prossimi step

IntEnum

DATA MASTERS

In Python, un enum (enumeration) è un modo per **strutturare dei dati** e associare dei **nomi** a dei **valori costanti.**
Un po'come usare delle variabili, ma in maniera **più strutturata.**

```
from enum import IntEnum
class Action(IntEnum):
    Rock = 0
    Paper = 1
    Scissors = 2
```

- **`IntEnum`** è un `enum` in cui possiamo associare dei **valori interi**
- **`class`** è una parola chiave del linguaggio
- Posso accedere al valore «0» con:
  - `Action.Rock`
  - `Action(0)`

IntEnum

DATA MASTERS

Ma… perché dobbiamo complicarci la vita?

- Per ragioni di **leggibilità** del codice: il codice diventa più generico ed auto-esplicante per la lettura da parte di altri sviluppatori
- Evitare **bug**: cosa accadrebbe se lo sviluppatore digitasse «dock» al posto di «rock» in una linea di codice?
  Evitare **magic values**
- Supporto IDE: utile per velocizzare la fase di sviluppo

In generale, ottimizzare le cose durante lo sviluppo **evita mal di testa dopo**

FUNZIONI

DATA MASTERS

- È un gruppo di istruzioni **aggregate** insieme
- Può lavorare con dei dati di input e opzionalmente restituire un output
- Un metodo per:
  - Evitare di riscrivere il codice (principio DRY: Don't Repeat Yourself)
  - Raggruppare in un'unica istruzione più istruzioni
- In sostanza è un modo semplice per **ottimizzare** il nostro codice

IntEnum

DATA MASTERS

Una funzione ha due momenti nella sua vita:

- **Definizione**
- **Uso** (o **invocazione**)

```
def average(numbers_list):
    a = sum(numbers_list)
    n = len(numbers_list)
    avg = a / n
    return avg
…

print(average([5, 2, 4, 3]))
```

- **`def`** e **`return`** (opzionale) sono delle parole chiave del linguaggio
- **Definiamo** la funzione quando scriviamo il suo comportamento, eventuali valori in input, eventuali valori in output
- **Invochiamo** la funzione usando il suo nome e le parentesi tonde
- Eventuali argomenti di input vanno specificati fra le parentesi tonde

TASK 3

DATA MASTERS

Let's code!

TASK

- usare un IntEnum per memorizzare le possibili scelte di gioco
- usiamo delle funzioni

TASK 3

- È tempo di semplificare i controlli andando ad **eliminare un po'di quegli if**
- L'idea è che **meno codice scriviamo, meno rischiamo di sbagliare**
- Per semplificare la struttura dei controlli utilizziamo una struttura dati detta **dizionario**
- È un modo diverso di organizzare i dati, **coppie chiave/valore**
  - Il valore può essere un qualsiasi tipo di dato:
    - Nativo (int, float, string)
    - Lista, tupla
    - Altro dizionario

IntEnum

DATA MASTERS

```
person = {
    'first_name': 'Giuseppe',
    'last_name': 'Mastrandrea',
    'age': 38,
    'children': [
        'Giulia', 'Francesca'
    ]
}
```

- Si può creare racchiudendo fra parentesi graffe l'insieme di coppie chiave/valore
- Le coppie chiave valore sono separate da virgola
- Possiamo usare i valori dell'enum per creare un dizionario delle mosse vincenti per ciascuna mossa

TASK 3

Let's code!

TASK

- usare un dizionario per specificare (per ogni mossa) quali sono le mosse vincenti
- usiamo il dizionario e la parola chiave `in` per semplificare i controlli

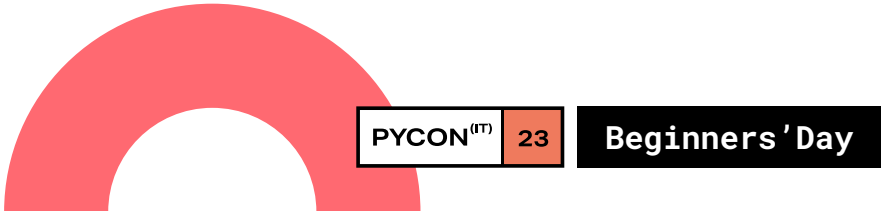
TASK 3

A questo punto, con le ottimizzazioni che abbiamo fatto, possiamo concludere la prima versione del gioco aggiungendo le nuove mosse: **Lizard e Spock**

Al posto di scrivere altri rami di if ed esporci a possibilità di errore ci basta modificare:

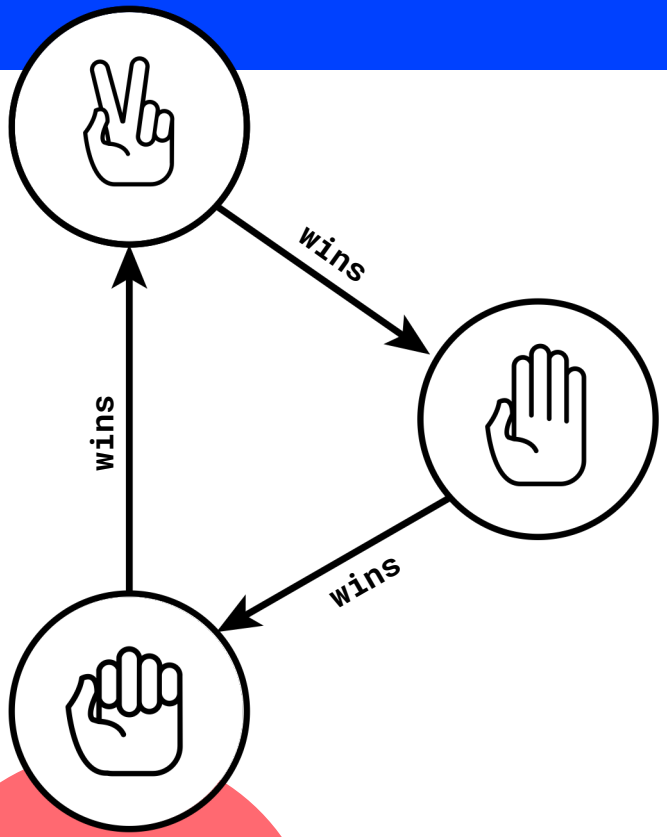
- La classe **`actions`**
- Il dizionario **`victories`**

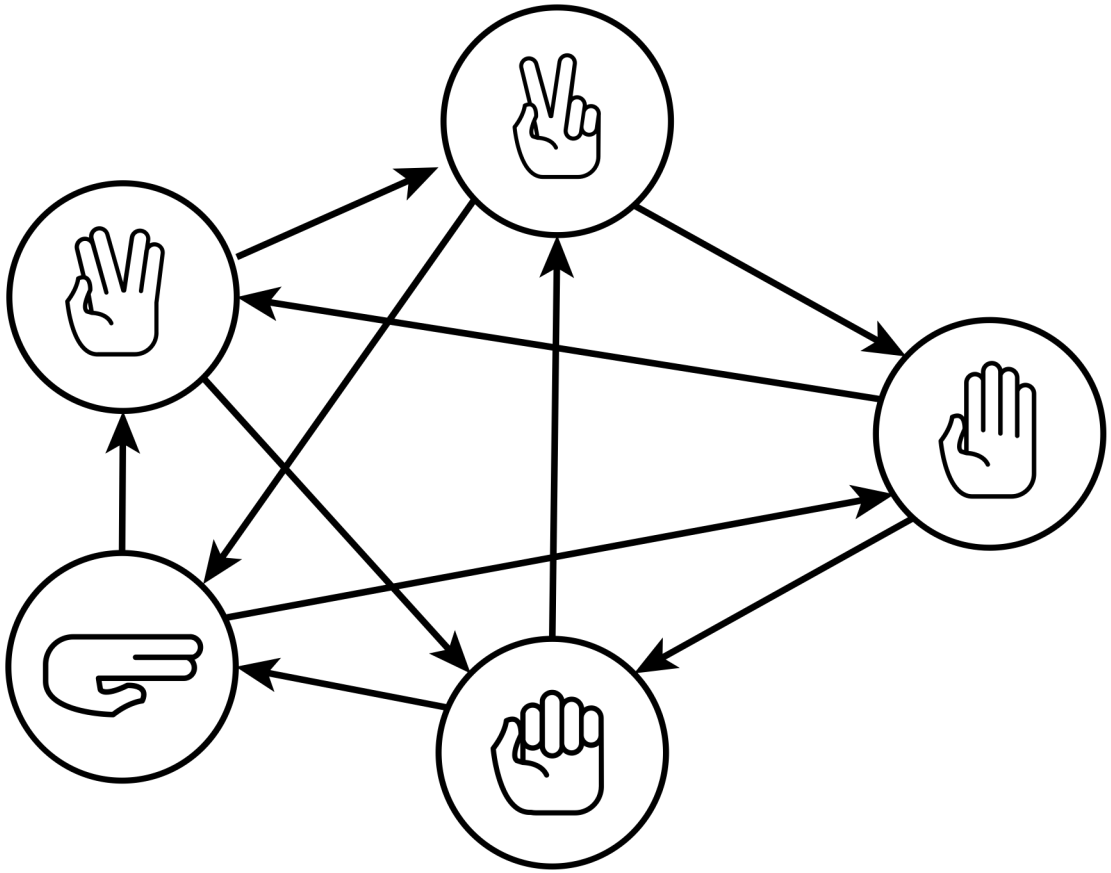
Non c'è bisogno di altro!



TASK 3

DATA MASTERS





TASK 3

DATA MASTERS

Let's code!

TASK

Aggiungere le nuove mosse alla classe Action e al dizionario

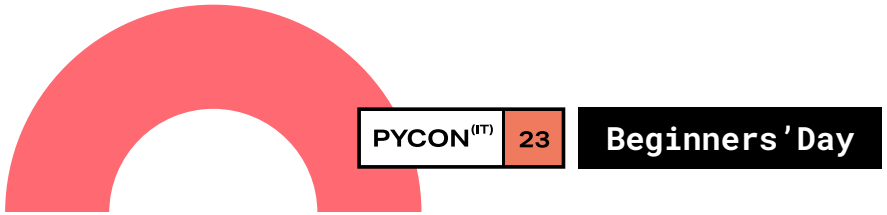
TASK 3

DATA MASTERS

Perché non usare **l'ASCII art?**

- Nuove nozioni: **raw string**
- Multiline string

Let's code!



TASK 3

DATA MASTERS

E se volessimo utilizzare delle **immagini** al posto dell'ascii art?

Let's code!

Multiline string

TASK 3

DATA MASTERS

Let's code!

Usiamo le funzionalità di **Jupyter** e **Colab** per **aggiungere un menu a tendina** evidando che l'utente inserisca la mossa da tastiera

TASK 4

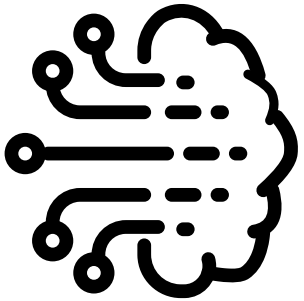
DATA MASTERS

Tempo di aggiungere un po’di **Machine Learning!**

Useremo il Machine Learning per **riconoscere la mossa dell’utente** utilizzando la webcam

Crash course su MACHINE LEARNING

DATA MASTERS



Machine Learning: campo dell'intelligenza artificiale (AI) che si concentra sullo sviluppo di algoritmi e modelli che consentono ai computer di apprendere e fare **previsioni basate sui dati**, senza essere esplicitamente programmati.

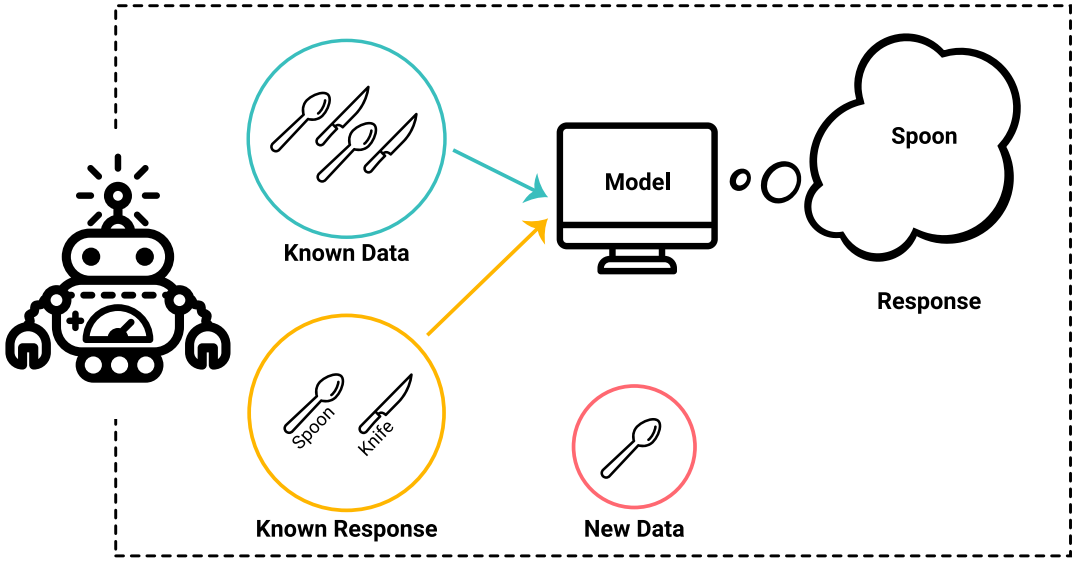
Come è possibile?

I modelli di machine learning lavorano con **grandi quantità di dati** per identificare pattern nei dati e fare delle predizioni per il futuro quando vedono **nuovi dati.**

PYCON(IT) 23 Beginners'Day

Crash course su MACHINE LEARNING

DATA MASTERS



- Fase di **train** (addestramento): i modelli «studiano» serie di dati storici **etichettati** provando a fare delle predizioni su di essi e ripetendo il processo finchè l'errore che commettono è più basso possibile

- Dopo l'addestramento riceveranno in input dei nuovi dati ***(formattati esattamente come quelli di addestramento)*** e faranno su di essi delle predizioni

PYCON(IT) 23 Beginners'Day

Crash course su MACHINE LEARNING

DATA MASTERS

Task di **Classificazione**

ML Model
(Classifier)

0: 0.2% confidence
1: 2.5% confidence
2: 1.2% confidence
3: 92,5% confidence
4: 0.88% confidence
5: 0.3% confidence
6: 0.2% confidence
7: 0.5% confidence
8: 1.5% confidence
9: 0.2% confidence

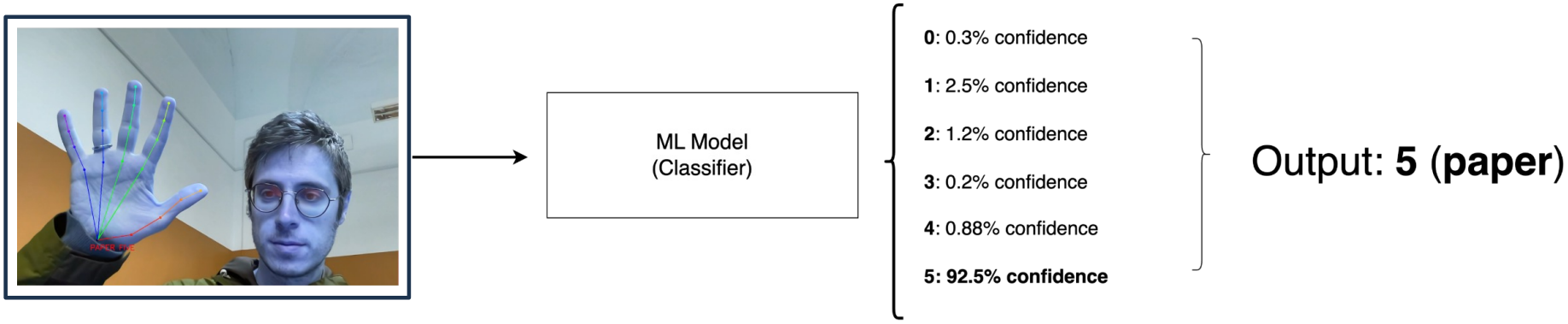
Output: **3**

PYCON(IT) 23 Beginners'Day

Crash course su MACHINE LEARNING

DATA MASTERS

Task di **Classificazione**

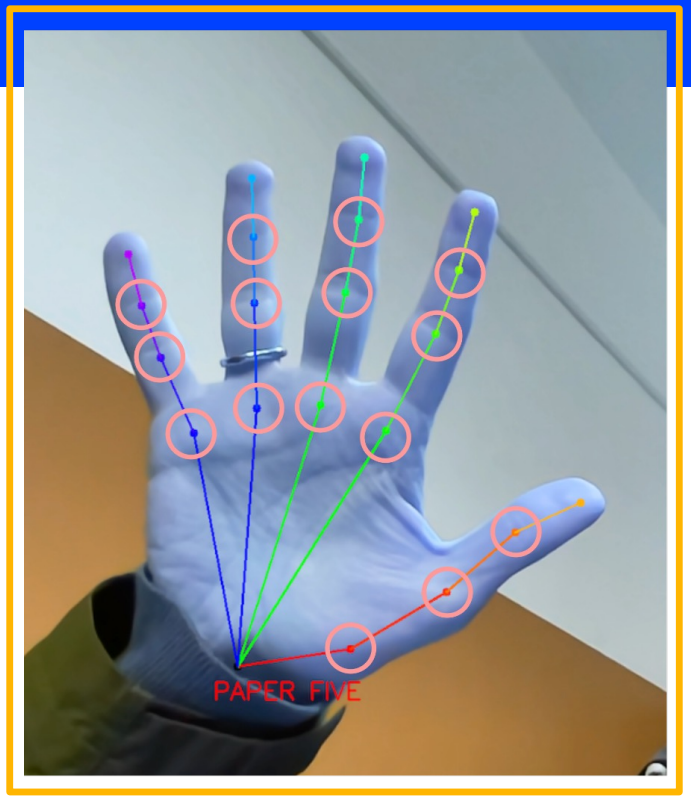


Crash course su MACHINE LEARNING

DATA MASTERS

Quali saranno i dati di **input** nel nostro caso?

- Il dataset sarà una file CSV (una tabella) fatta da:
  - **109** righe
  - **16** colonne
- Ogni riga rappresenta un'immagine diversa di una mano
- Ogni colonna rappresenta un **angolo** fra i 15 evidenziati in figura
- La 16esima colonna rappresenta **la label** (il numero corrispondente a quel determinato insieme di angoli)

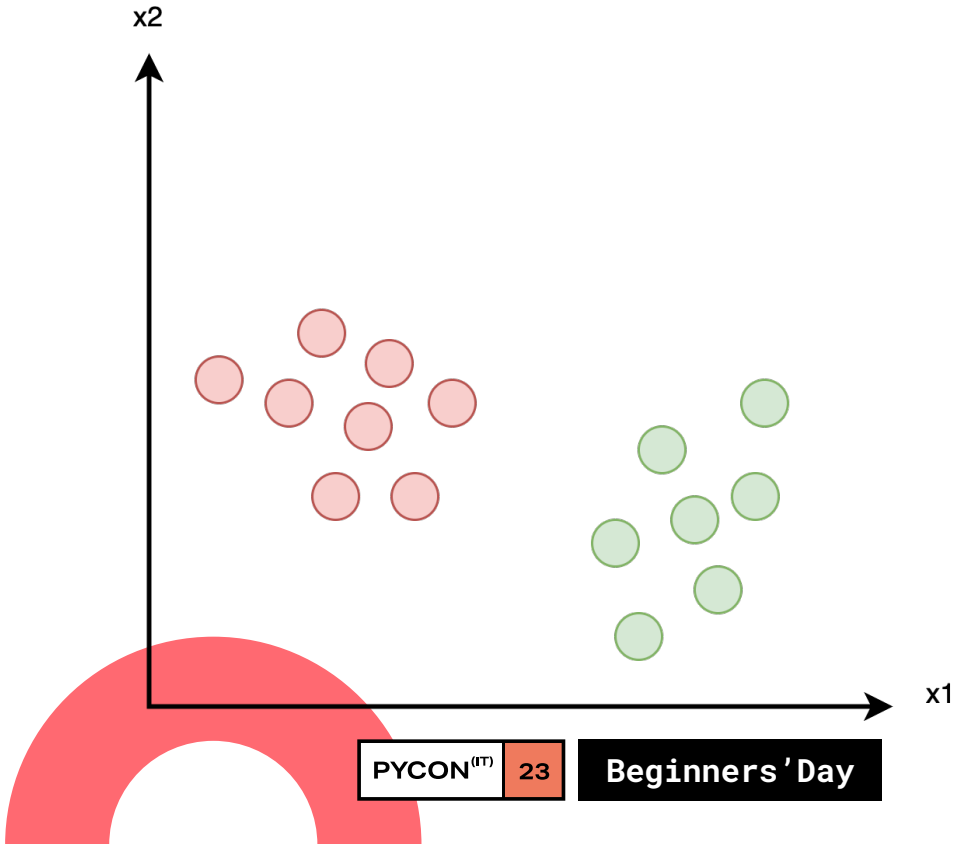


Crash course su MACHINE LEARNING

- Quindi il flusso sarà:
  - Acquisiamo il flusso di frame dalla webcam (video)
  - Estraiamo i **keypoint** dal video acquisito
  - I keypoint sarebbero gli **angoli** che corrispondono alla gesture dell'utente
  - Ottengo quindi una serie di **15 numeri** esattamente come le feature di input del dataset
  - **Manca la 16-esima colonna**: è quella che il modello deve prevedere!
  - Algoritmo utilizzato: KNN (K-Nearest Neighbors)

Crash course su MACHINE LEARNING

DATA MASTERS



Esempio

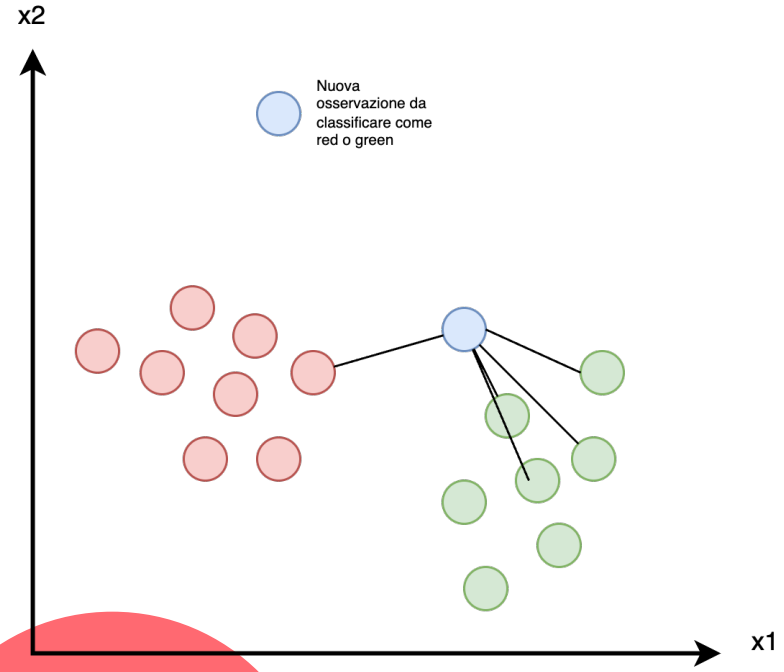
Dataset con solo **2 features**

- x1 e x2: **feature**
- colore: classe finale di appartenenza

Crash course su MACHINE LEARNING

DATA MASTERS

KNN su 2 feature e 1 classe di output:



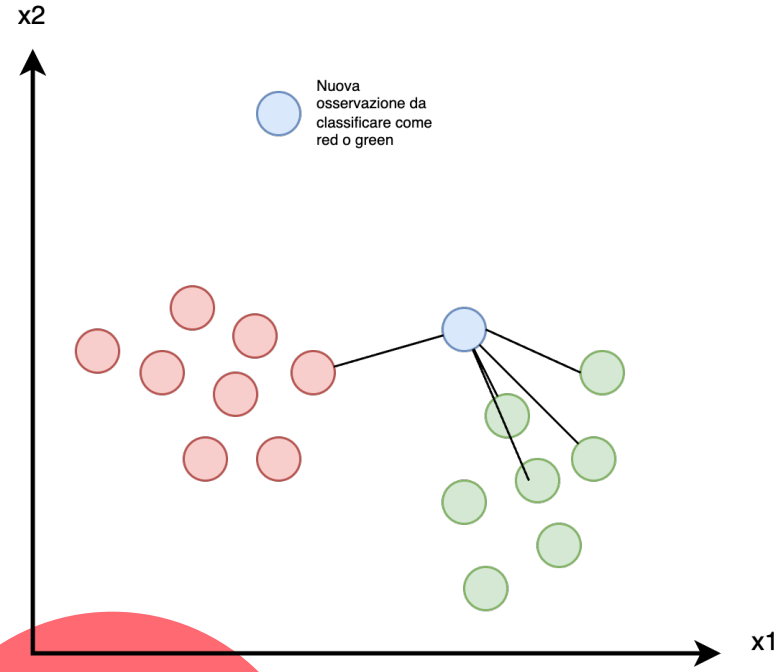
Classificare una nuova osservazione:

- Settare un valore di K (es. pari a 5)
- Calcolare le distanze fra la nuova osservazione e **tutte le osservazioni del dataset**
- Prendere i K valori più bassi
- Vedere quale è la classe più presente fra i K valori più vicini: sarà il valore della predizione finale

Crash course su MACHINE LEARNING

DATA MASTERS

KNN su 2 feature e 1 classe di output:



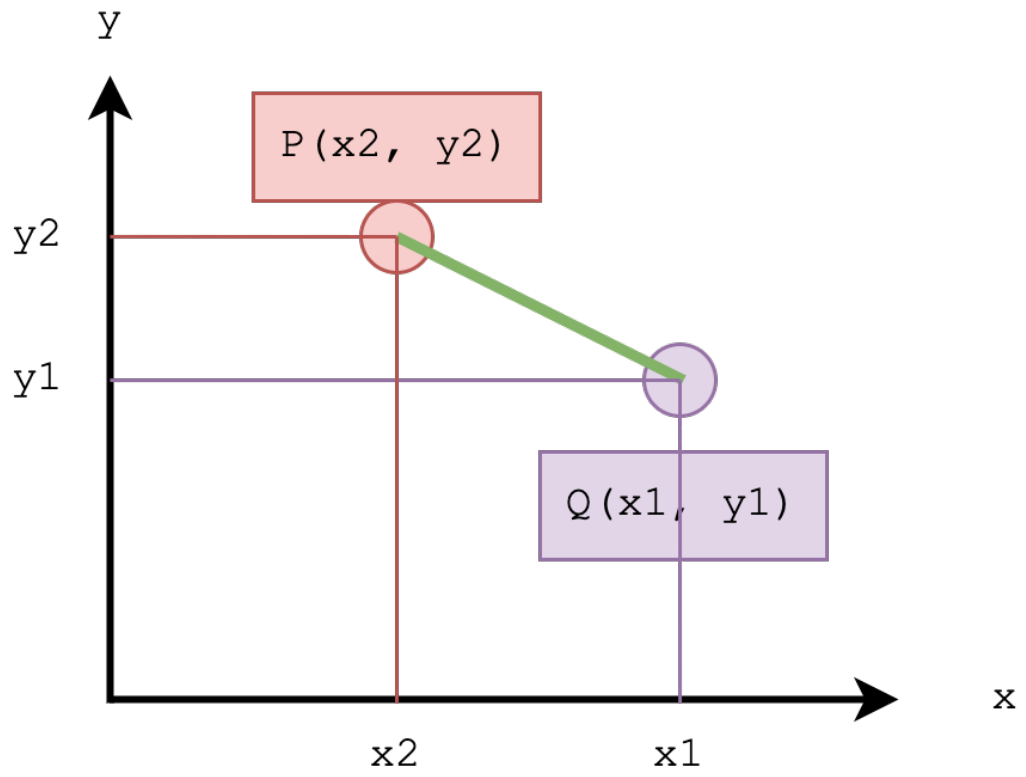
- Come misurare la distanza?
- **Distanza euclidea**
  - in 2 dimensioni è semplicemente l'applicazione del teorema di Pitagora

TASK 2

DATA MASTERS

- **Euclidean distance**
- We use **Pythagoras Theorem**
- The distance is the green line (hypotenuse) of the triangle in figure

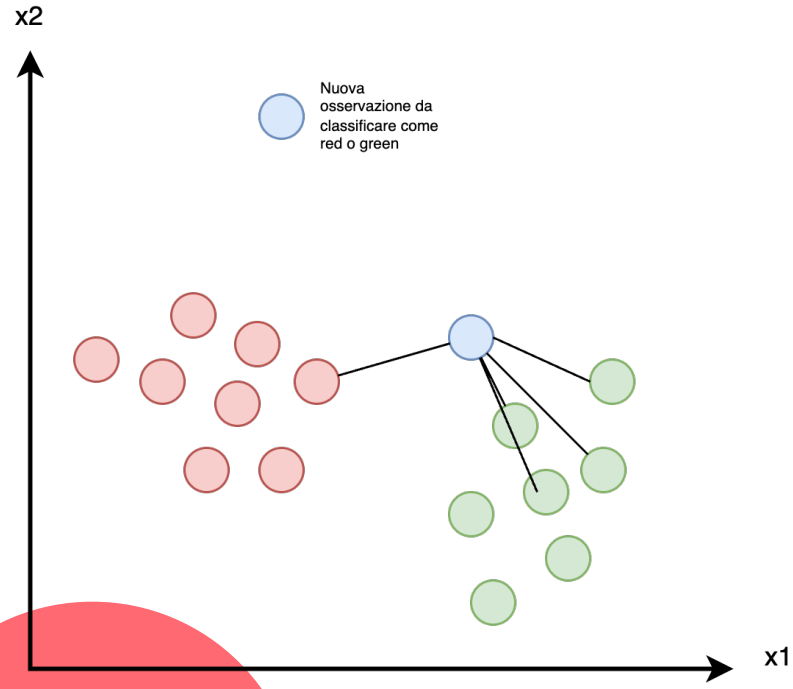
$$d_{P,Q} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



Crash course su MACHINE LEARNING

DATA MASTERS

KNN su 2 feature e 1 classe di output:



- Formula generale: $d_{P,Q} = \sqrt{\sum_{i}^{N}(P_i - Q_i)^2}$
- Nel nostro caso N = 15
- Fra i 5 vicini della nuova osservazione abbiamo
- 4 osservazioni con classe verde
- 1 osservazione con classe rossa
- 4 > 1 -> **vince la classe verde**

TASK 4

DATA MASTERS

Usiamo le librerie e i modelli per:

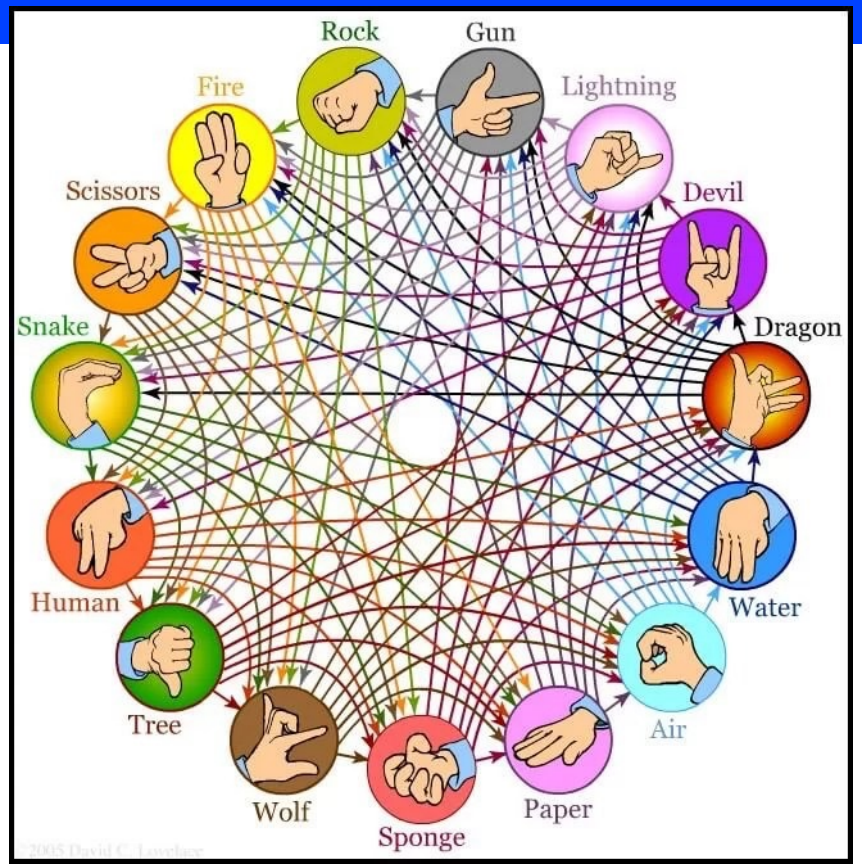
Let's code!

- Leggere i dati dalla webcam
- Rilevare la gesture
- Mapparla ad una delle mosse che ci interessano
- Completare così la partita!

That's it!

DATA MASTERS

NOW
you're ready
for **this!**

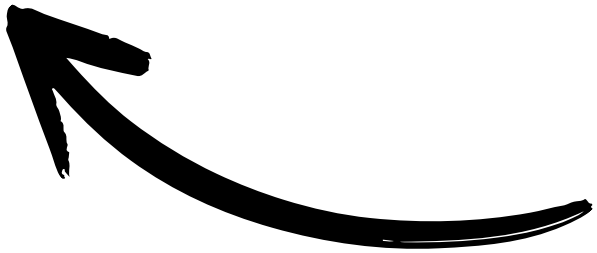


PYCON(IT) 23 Beginners' Day

Join us on Discord

DATA MASTERS

launchpass.com/datamasters

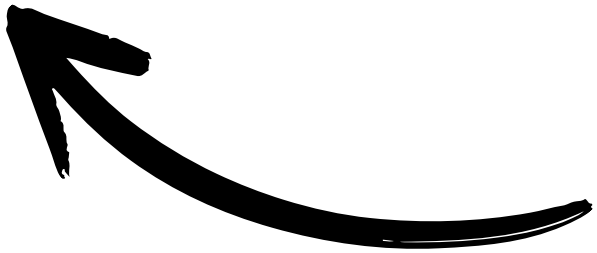


PYCON(IT) 23 Beginners' Day

Last but not least

DATA MASTERS

https://forms.gle/tGwbDDfmKLMMxe7p7



PYCON(IT) 23 Beginners'Day