

# Вступ до Data Science. Знайомство з бібліотекою Numpy

Заняття 1

Python для Data Science

# ПЛАН ЗАНЯТТЯ

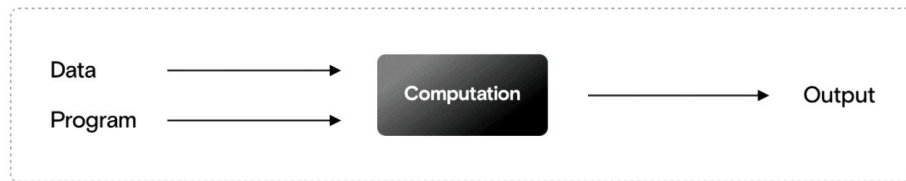


- Знайомство
- Формулювання задач машинного навчання
- Основні види задач машинного навчання
- Фреймворк роботи над задачами Data Science
- Знайомство з бібліотекою Numpy, її призначення і застосування

# MACHINE LEARNING VS TRADITIONAL PROGRAMMING

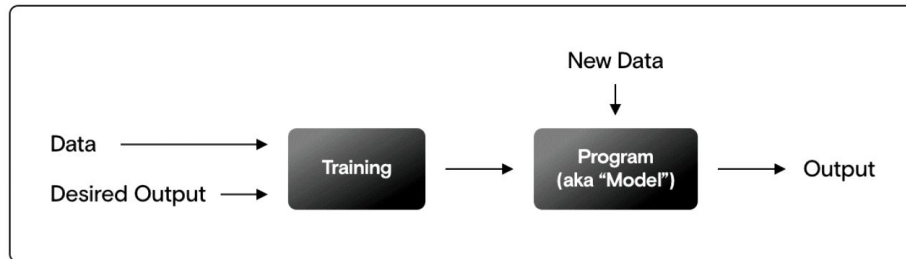
## Traditional Programming

Developers write rules (program) that produce an output.



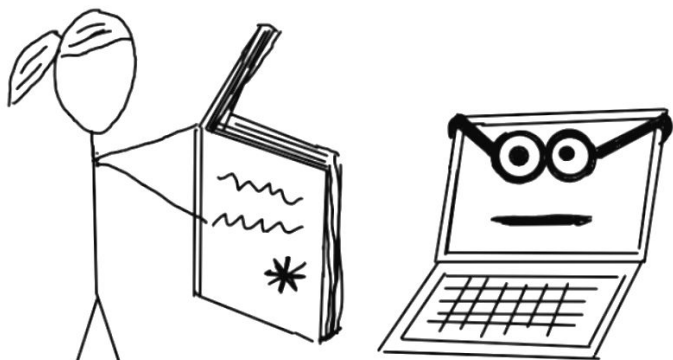
## Machine Learning

Developers write a training algorithm, that finds rules, which produce the desired output.



# MACHINE LEARNING VS TRADITIONAL PROGRAMMING

Without Machine Learning



\* VERY SPECIFIC INSTRUCTIONS

With Machine Learning

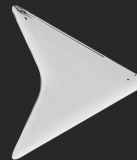


# MACHINE LEARNING

Data



Training



Model



# EXAMPLES OF MACHINE LEARNING TASKS

Data

Sun + wind + weekday

Chest X-Ray

Gene data

E-mail Text



Prediction

Electricity price (tomorrow)

COVID-19 likelihood

Chance of rare disease

Spam or not

# EXAMPLES OF MACHINE LEARNING TASKS

## THE MAIN TYPES OF MACHINE LEARNING

```
graph TD; Root[THE MAIN TYPES OF MACHINE LEARNING] --> Classical[CLASSICAL ML]; Root --> Reinforcement[REINFORCEMENT LEARNING]; Root --> Ensembles[ENSEMBLES]; Root --> NN[NEURAL NETWORKS AND DEEP LEARNING]; Ensembles -.-> NN; Classical --- C["Simple data<br/>Clear features"]; Reinforcement --- R["No data, but<br/>we have<br/>an environment<br/>to interact with"]; Ensembles --- E["When quality is<br/>a real problem"]; NN --- N["Complicated data<br/>Unclear features<br/>Belief in a miracle"]; Ensembles -.-> NN; NN --- NN2["Eternal<br/>competitors"];
```

**CLASSICAL ML**

Simple data  
Clear features

**REINFORCEMENT  
LEARNING**

No data, but  
we have  
an environment  
to interact with

**ENSEMBLES**

When quality is  
a real problem

**NEURAL NETWORKS  
AND DEEP  
LEARNING**

Complicated data  
Unclear features  
Belief in a miracle

*Eternal  
competitors*

# EXAMPLES OF MACHINE LEARNING TASKS

## THE MAIN TYPES OF MACHINE LEARNING

Simple data  
Clear features

**CLASSICAL ML**

No data, but  
we have  
an environment  
to interact with

**REINFORCEMENT  
LEARNING**

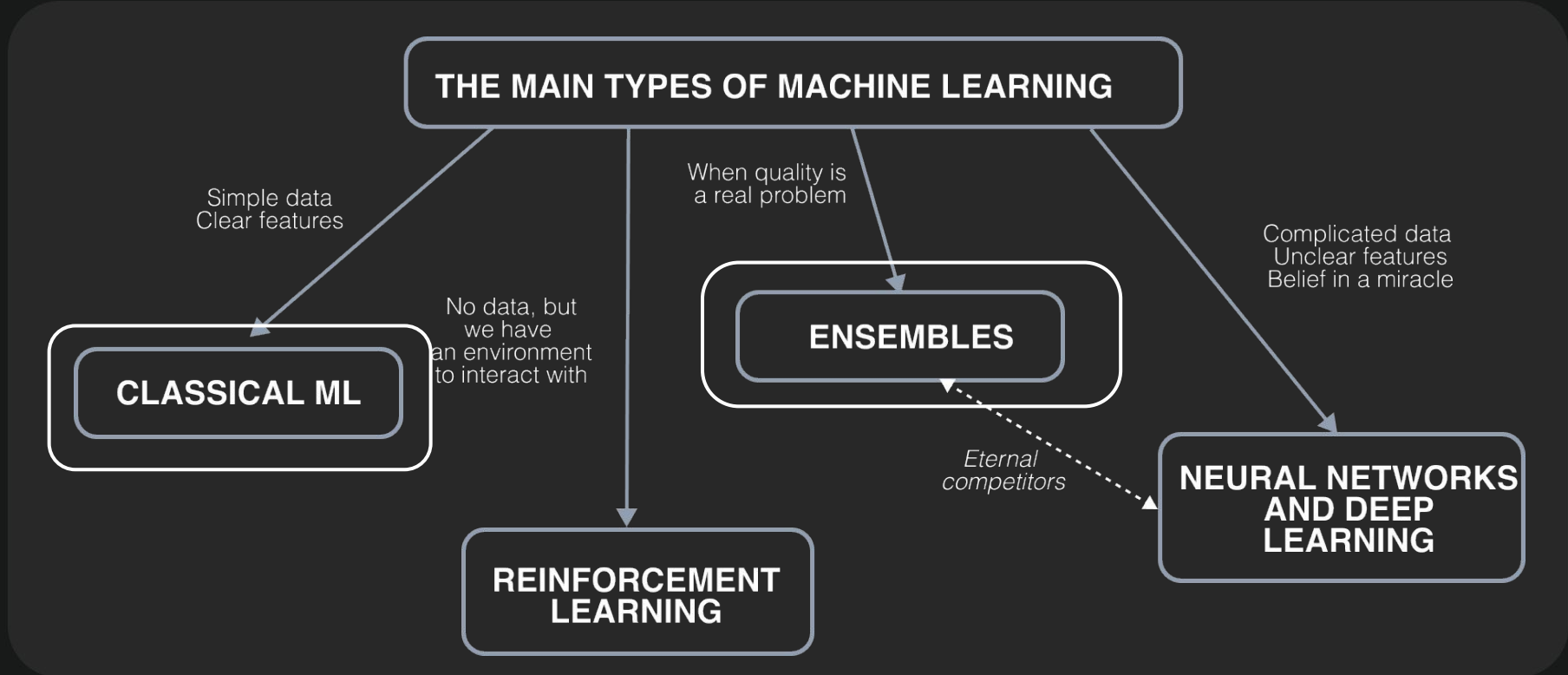
When quality is  
a real problem

**ENSEMBLES**

Complicated data  
Unclear features  
Belief in a miracle

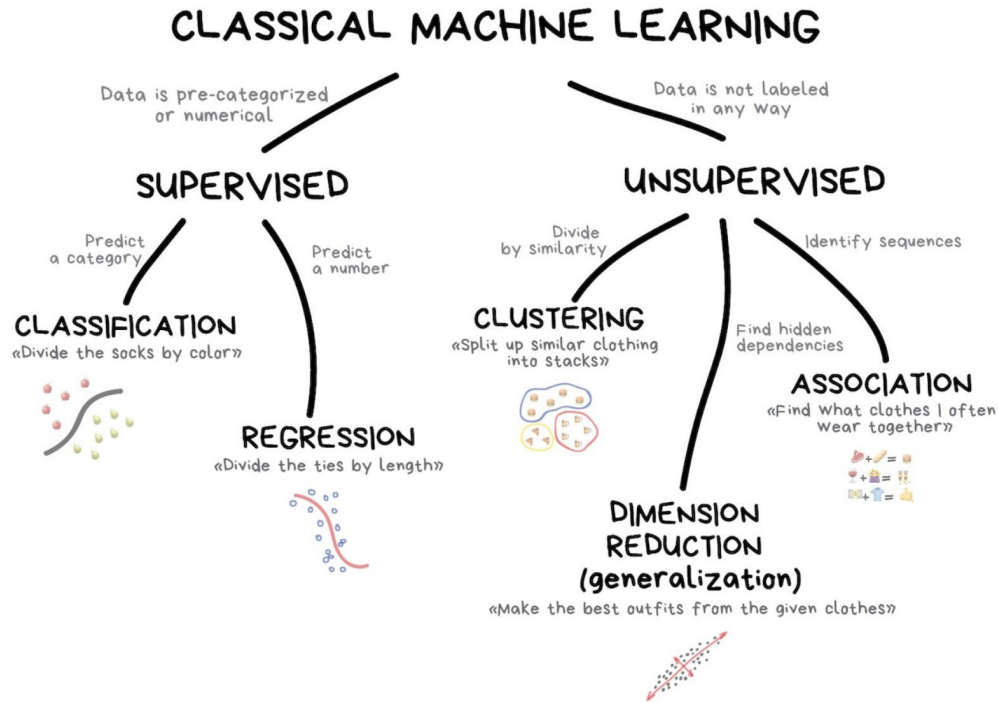
**NEURAL NETWORKS  
AND DEEP  
LEARNING**

*Eternal  
competitors*



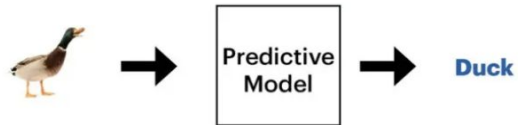
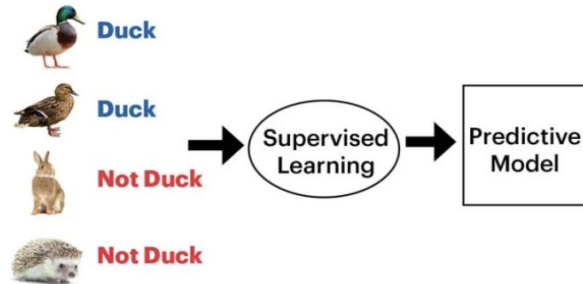


# CLASSICAL MACHINE LEARNING

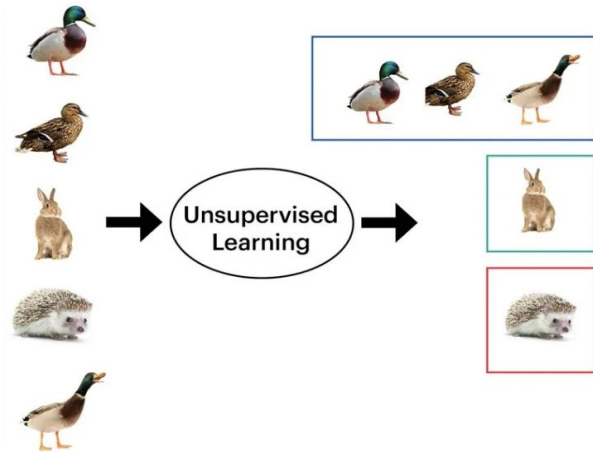


# CLASSICAL MACHINE LEARNING

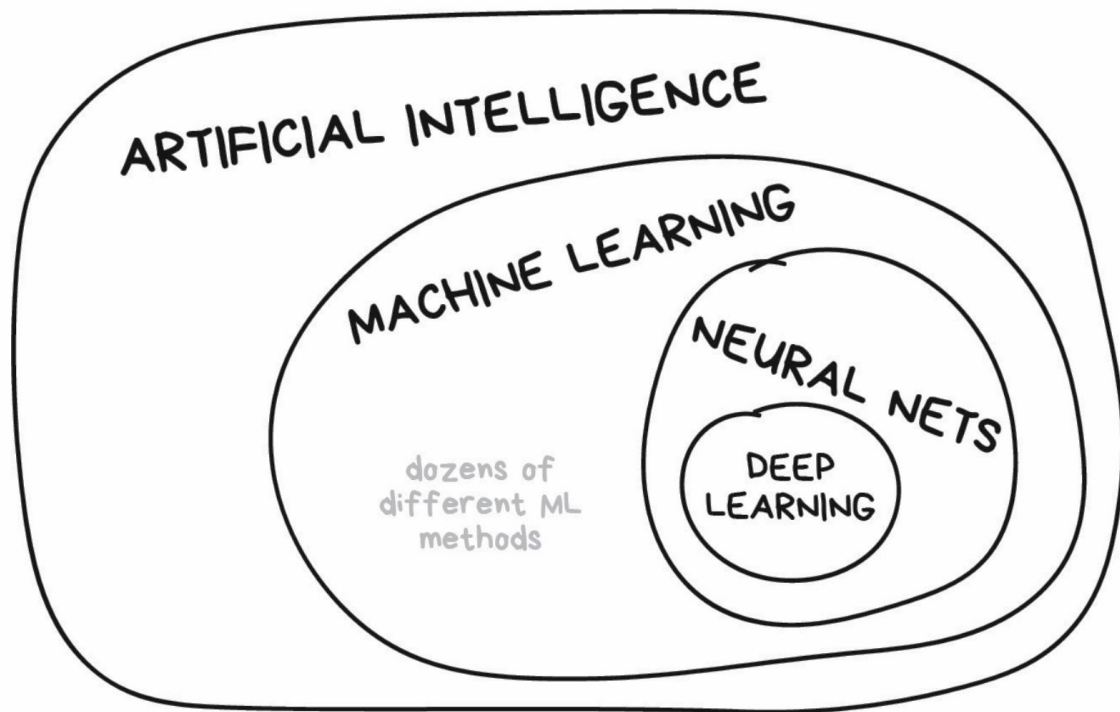
## Supervised Learning (Classification Algorithm)



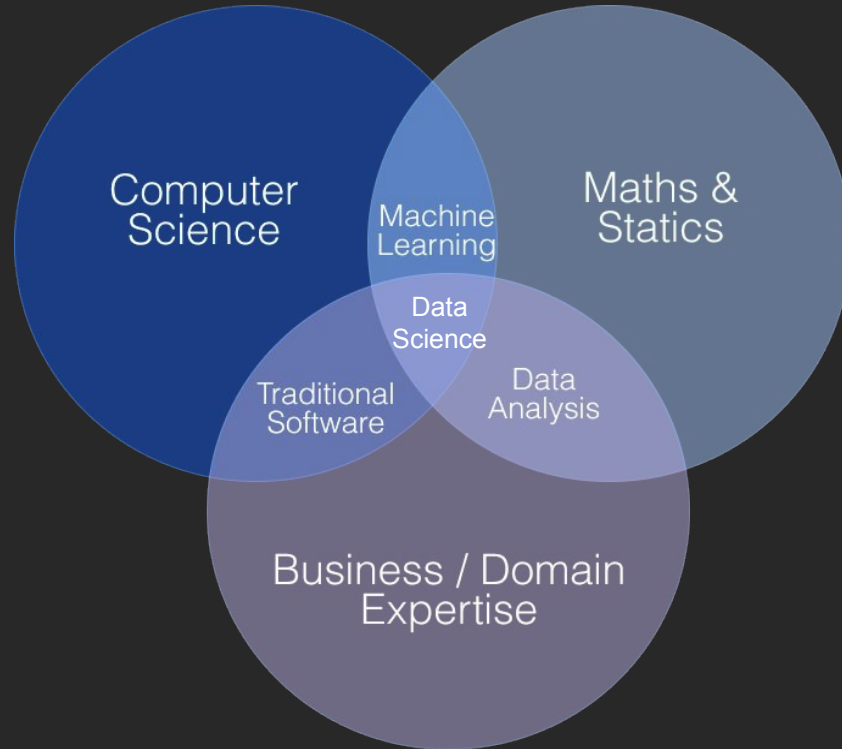
## Unsupervised Learning (Clustering Algorithm)



# AI VS ML VS NN



# DATA SCIENCE

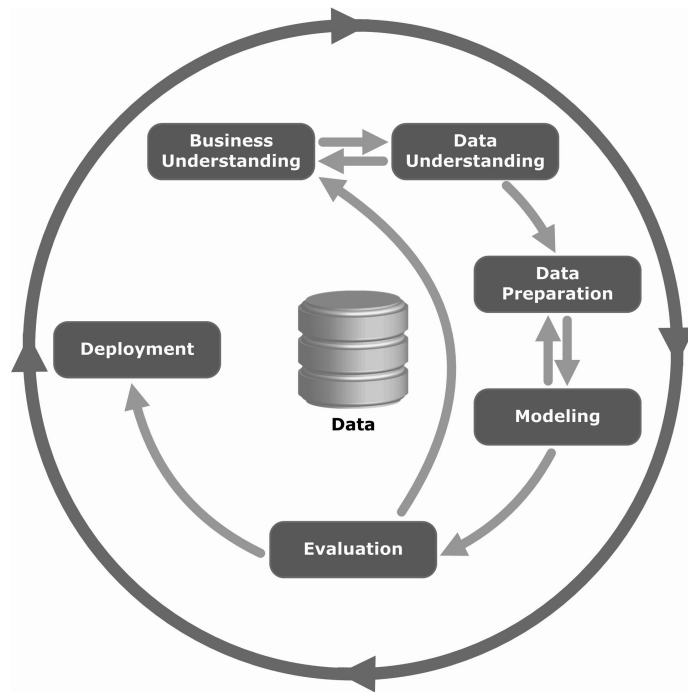


# CRISP-DM

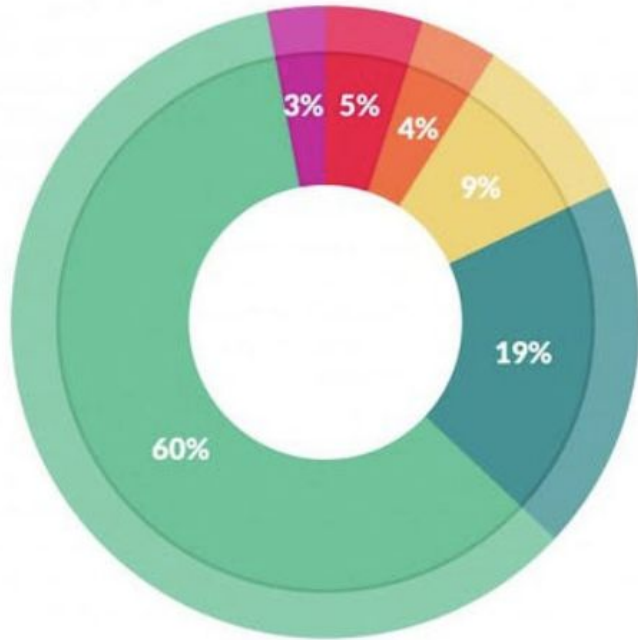
**CRISP-DM** — Cross-industry standard process for data mining

Міжгалузевий стандартний процес інтелектуального аналізу даних.

**Дані — це центральна точка.**



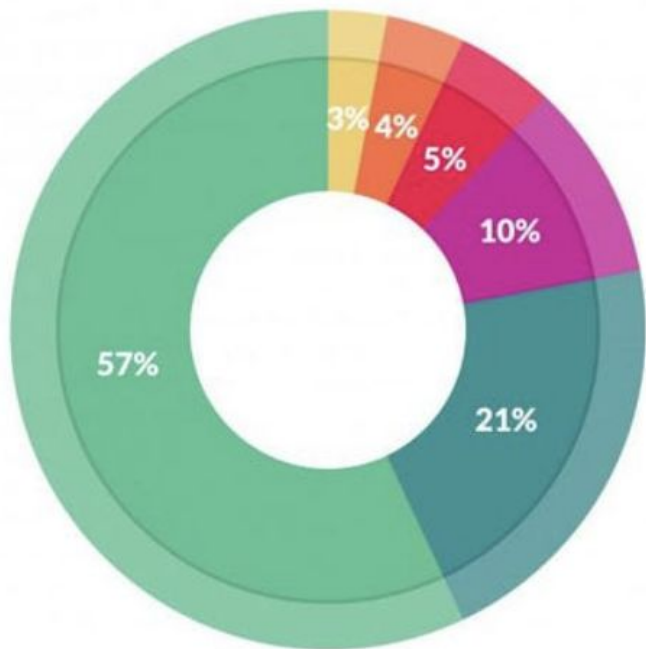
# DATA IS A KEY



## What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets; 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

# ... BUT THE LEAST ENJOYABLE KEY



What's the least enjoyable part of data science?

- Building training sets: 10%
- Cleaning and organizing data: 57%
- Collecting data sets: 21%
- Mining data for patterns: 3%
- Refining algorithms: 4%
- Other: 5%

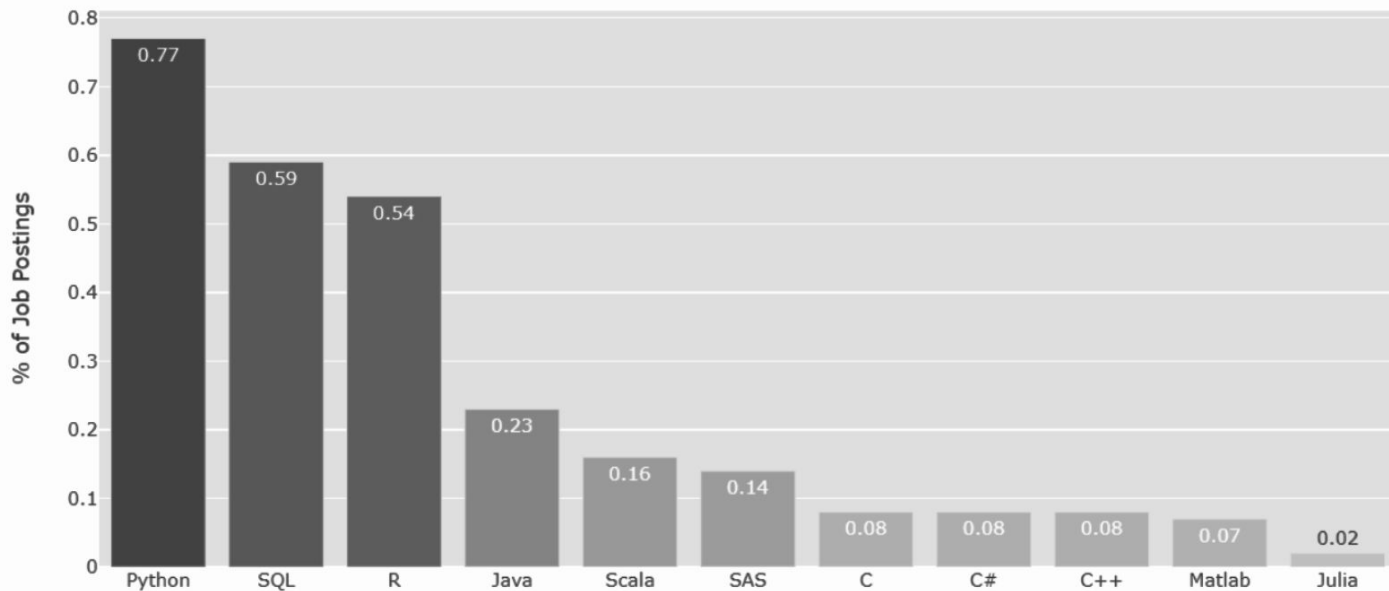
**ЗАПИТАННЯ**





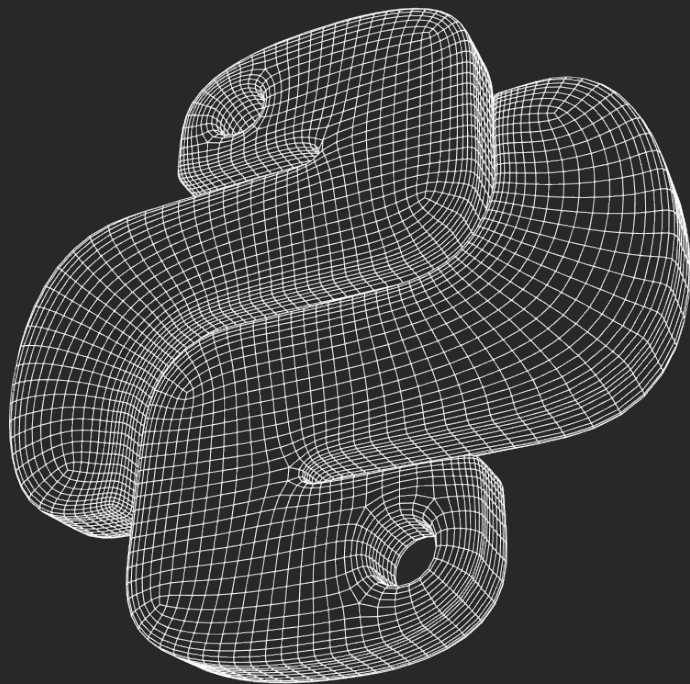
# ЧОМУ САМЕ PYTHON ДЛЯ DATA SCIENCE?

Top Programming Languages for Data Scientists



# TOP PYTHON LIBRARIES FOR DATA SCIENCE

- NumPy
- Pandas
- SciPy
- Matplotlib
- scikit-learn
- Seaborn
- Plotly
- Statsmodels
- XGBoost
- ... and many more



# NUMPY

NumPy, скорочення від **Numerical Python**, є основним пакетом, необхідним для високопродуктивних наукових обчислень і аналізу даних в екосистемі Python.

Це основа, на якій побудовані майже всі інструменти вищого рівня, як-от Pandas і scikit-learn.

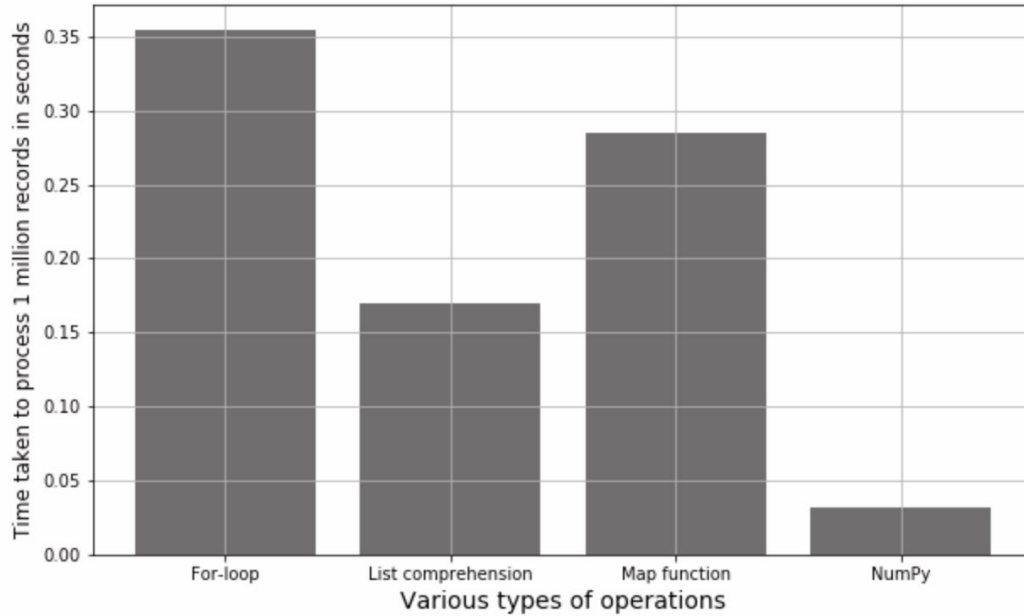
TensorFlow використовує масиви NumPy як фундаментальний будівельний блок, на основі якого вони побудували свої об'єкти Tensor і графічний потік для завдань глибокого навчання (Deep Learning) (де активно використовують операції лінійної алгебри над довгим списком/вектором/матрицею чисел).

# NUMPY

NumPy надає два фундаментальні об'єкти:

- N-dimensional array object (ndarray) є однорідною колекцією «елементів», проіндексованих за допомогою N цілих чисел. Є два важливих параметра, які визначають N-вимірний масив:
  - форма масиву
  - тип елемента, з якого складається масив
- A universal function object (ufunc). NumPy надає безліч математичних функцій, які працюють з ndarray object. Від алгебраїчних функцій, як-от додавання та множення, до тригонометричних функцій, як-от  $\sin$  і  $\cos$ . Усі ufunc завжди виконуються поелементно.

# NUMPY IS FAST. VECTORIZATION AND BROADCASTING



Bar chart of comparative speeds of execution of simple mathematical operations

# VECTORIZATION

Vectorization (векторизація) — це потужна здатність у NumPy виражати операції над цілими масивами, а не їх окремими елементами.

Під час циклічного перегляду масиву чи будь-якої структури даних у Python виникає багато накладних витрат. Векторизовані операції в NumPy делегують внутрішній цикл високооптимізованим функціям C і Fortran, що робить код Python чистішим і швидшим.

# BROADCASTING (ТРАНСЛЯЦІЯ)

Термін broadcasting (трансляція) описує, як NumPy обробляє масиви різної форми під час арифметичних операцій. З урахуванням певних обмежень менший масив «broadcast» («транслюється») через більший масив, щоб вони мали сумісні форми. Трансляція забезпечує засоби векторизації операцій з масивами, щоб цикли відбувалися в C замість Python.

# BROADCASTING (ТРАНСЛЯЦІЯ)

Shape: (3, 2)

0	1
2	4
10	10

Shape: (2, )

4	5
4	5
4	5

-

=

Shape: (3, 2)

-4	-4
-2	-1
6	5



**/LIVE CODING:**



Q&A

???