

# Intermediate Python & a Glimpse into AI applications

Class 1

Pepe Bonet Giner

9<sup>th</sup> January 2023

# Index Class 1

---

Topic 1: What is the course about?

Topic 2: The AI revolution. AI for Business

Topic 3: The Backbone of this new world

Topic 4: Python Recap

Topic 5: Pandas

Topic 6: Documentation

# Who am I?

---



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



Maastricht  
University



B.Sc. Biotechnology

M.Sc.  
Computational Biology



Universitat  
Pompeu Fabra  
Barcelona

ETH zürich



P.h.D. @ BbgLab  
Computational Biology  
EMIBA



hynts

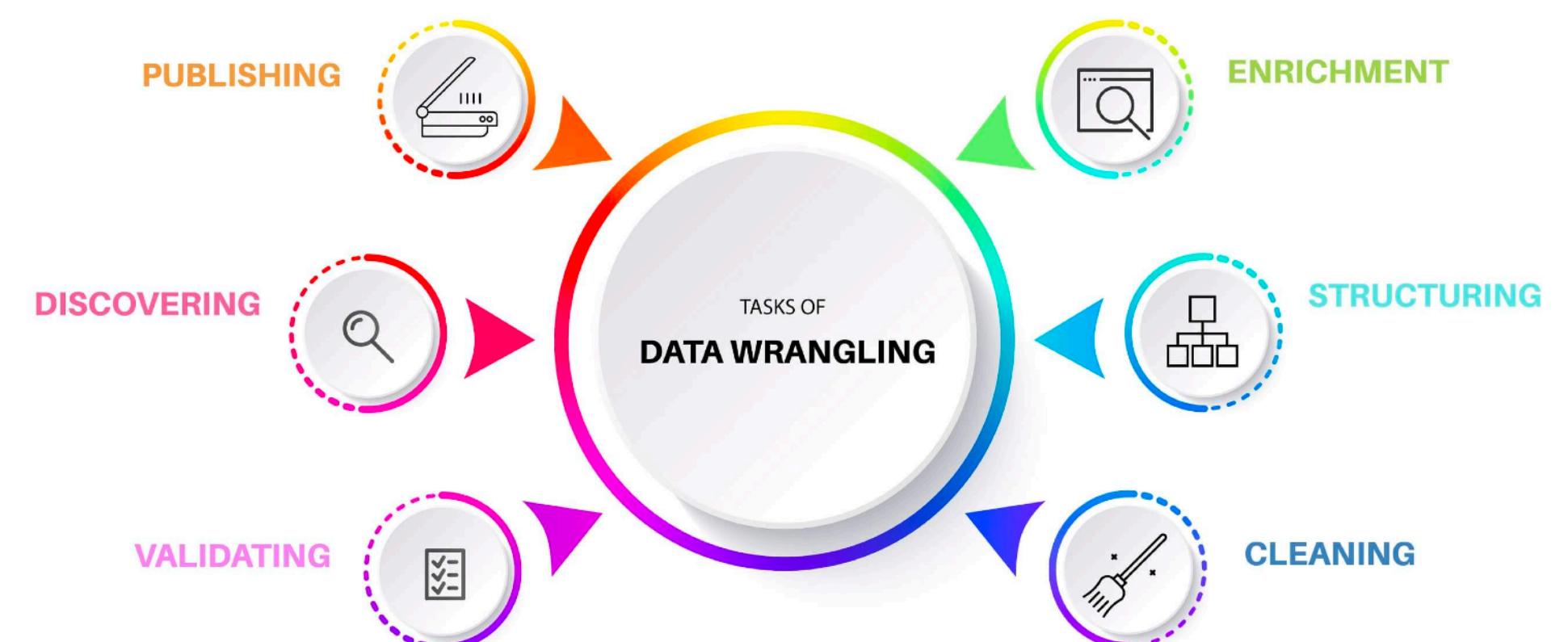
Hynts Analytics  
&  
Esade

# Topic 1: What is this course about?

# What are we going to learn?

---

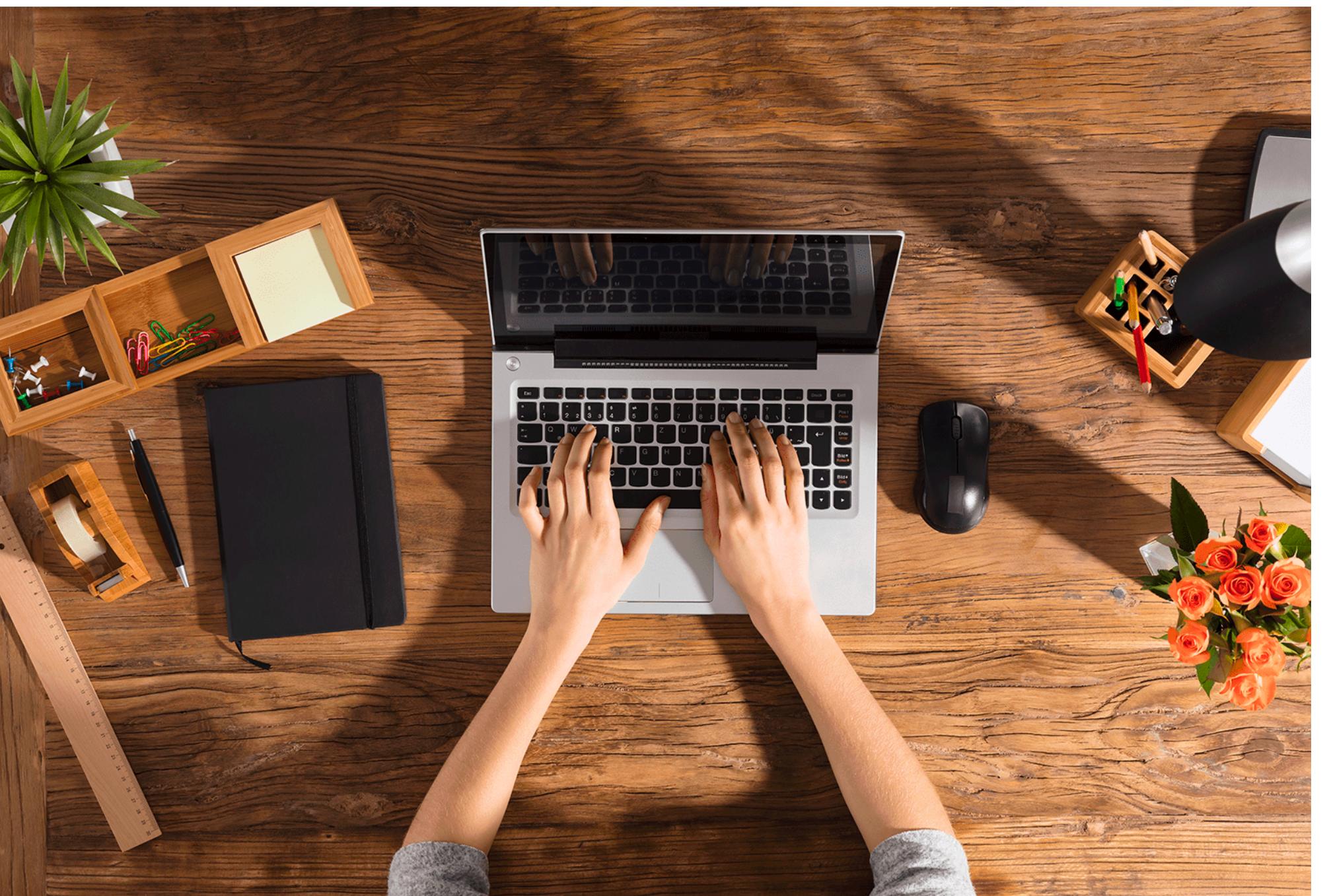
- Programming in the current world. The latest AI applications. Impact of AI in a variety of fields.
- Python fundamentals for data analysis: Pandas, Functions, Data Structures, Matplotlib
- Understanding your data with Python: Cleaning, transforming, and getting value from your data. Data Wrangling & Data Visualization
- Employ simple AI tools to get additional/predictive value from your data



# How is it going to be graded?

---

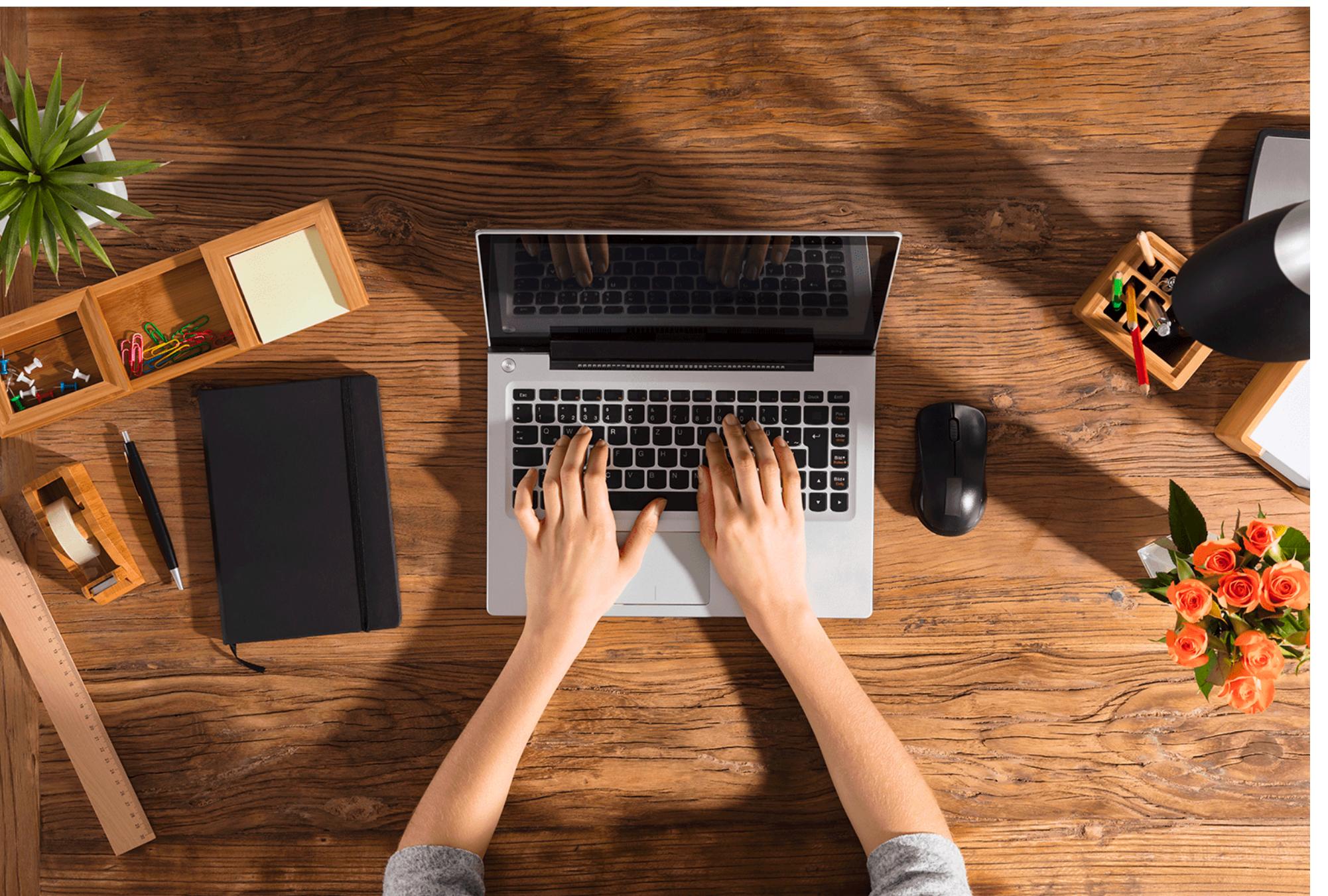
- **60% Active participation and class attendance:**
  - 15 % Attendance
  - 45 % coding exercises
  - Exercises will be uploaded to the Moodle after every session.



# How is it going to be graded?

---

- **60% Active participation and class attendance:**
  - 15 % Attendance
  - 45 % coding exercises
  - Exercises will be uploaded to the Moodle after every session.
- **40% Final project:**
  - Python code on Jupyter Notebook, working and applying the concepts learned in class on a dataset, explaining the steps taken.
  - A 2-page report summarizing the results, the value extracted from data, and possible applications of AI.
  - Upload both to the Moodle max two weeks after the last lecture.



# Final work

---

Code in Jupyter Notebooks

Report Explaining Results

## Class 1. Intermediate Python & AI

### Table of contents

- Variables
- Lists
- If-else
- For loops
- While Loops
- Functions

### Variables

```
In [5]: a = 10  
        b = 5  
        print(a, b)
```

10 5

```
In [6]: a + b
```

```
Out[6]: 15
```



### Study of Police deaths in the USA from 1791 to 2022

Pepe Bonet Giner  
9<sup>th</sup> January 2023

- 1.- Introduction to the dataset
- 2.- Results obtained (in line with the code)
- 3.- Possible further AI applications

# Topic 2: The AI revolution. AI for Business

# There is something out there affecting us!

---

- Technology is impacting the world to an extent that we cannot grasp yet. This is having huge implications at different levels.

## Present

- Self-driving cars
- Social Media
- E-commerce
- Robots in factories



# There is something out there affecting us!

---

- Technology is impacting the world to an extent that we cannot grasp yet. This is having huge implications at different levels.

## Present

- Self-driving cars
- Social Media
- E-commerce
- Robots in factories

## Future

- Transportation: Autonomous Vehicles
- Customer Service: Chat Bots
- Education: Personalized Learning
- Healthcare: Personalized Medicine
- VR/AR

# There is something out there affecting us!

---

- Technology is impacting the world to an extent that we cannot grasp yet. This is having huge implications at different levels.

Present

- Self-driving cars
- Social Media
- E-commerce
- Robots in factories

Future

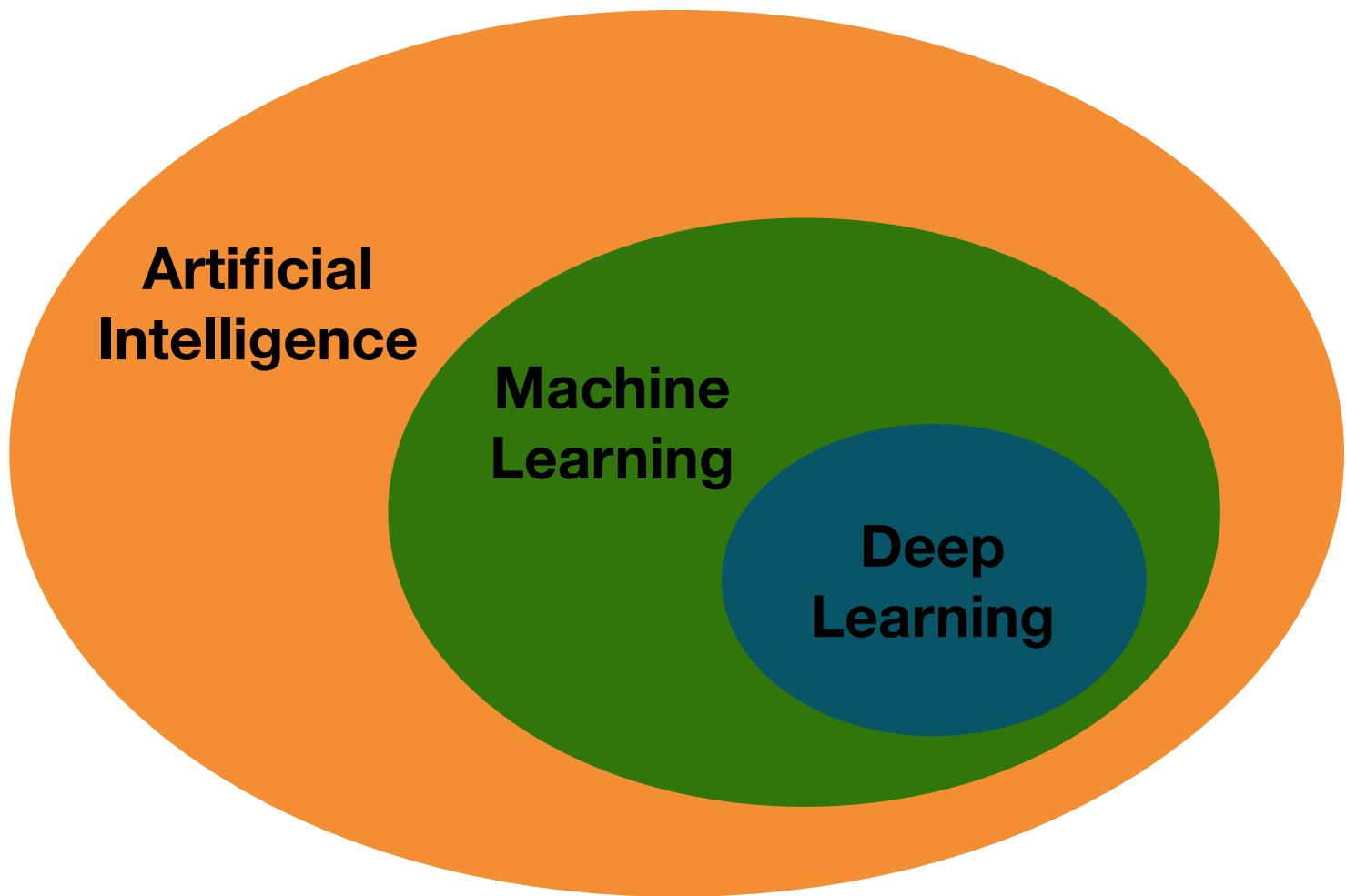
- Transportation: Autonomous Vehicles
- Customer Service: Chat Bots
- Education: Personalized Learning
- Healthcare: Personalized Medicine
- VR/AR

AI is behind many of these disruptions

# What is Artificial Intelligence (AI)?

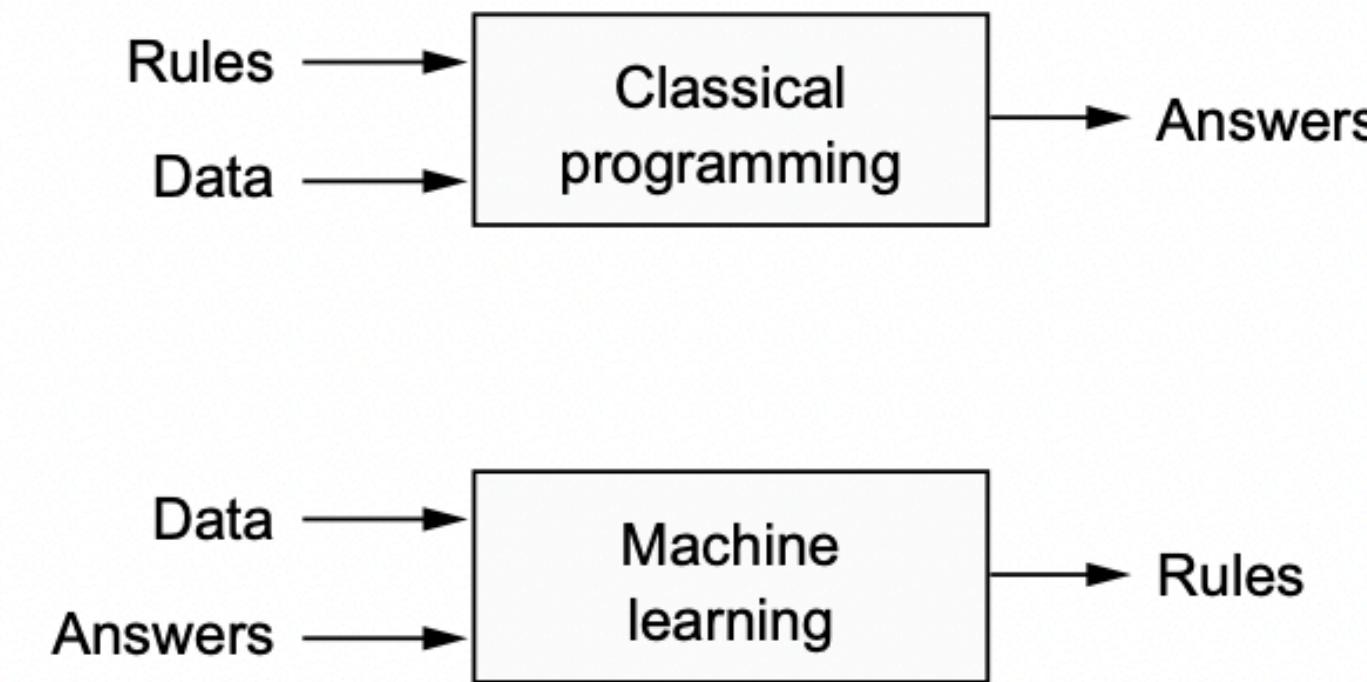
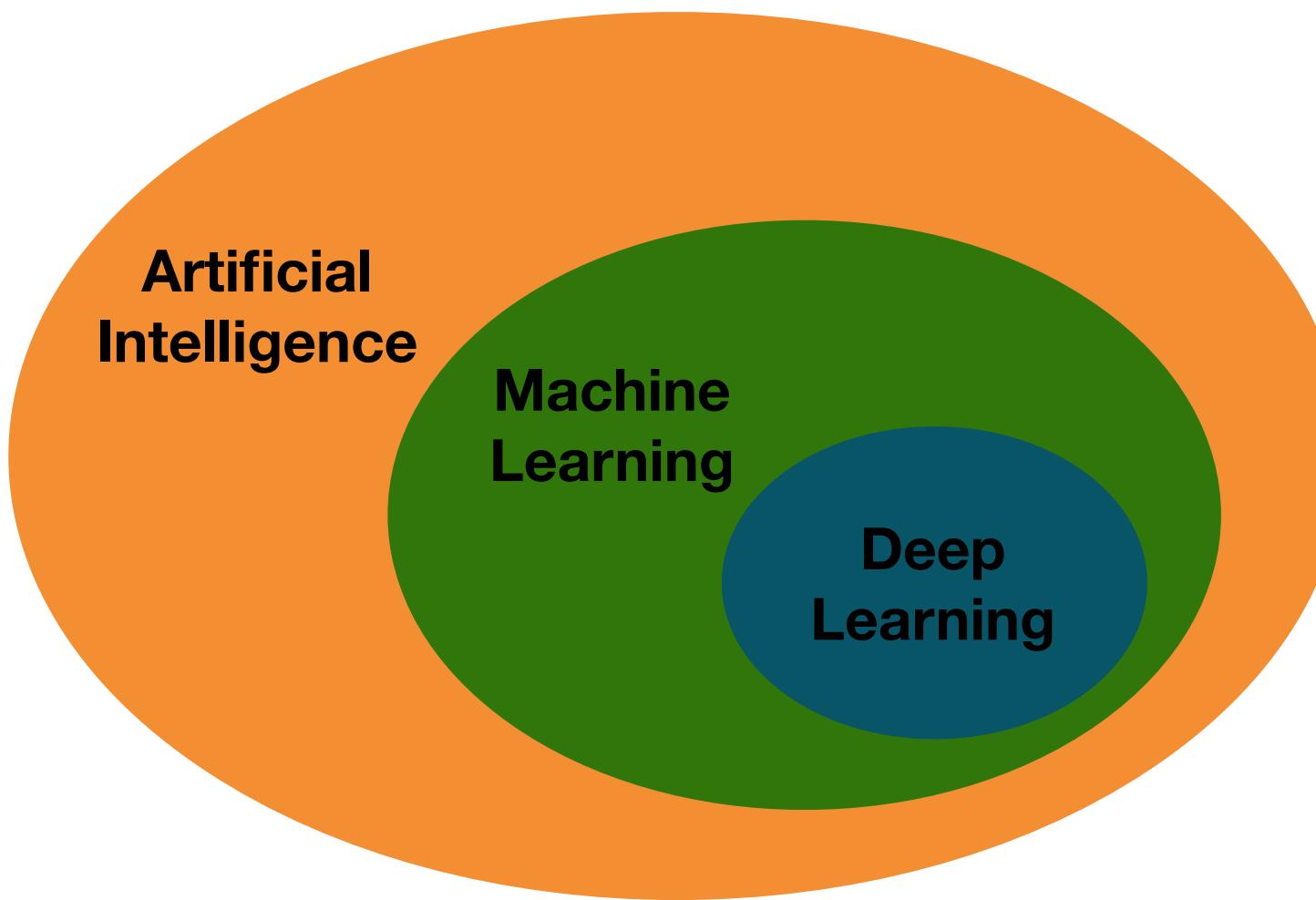
---

- AI: Effort to automate intellectual tasks normally performed by humans



# What is Artificial Intelligence (AI)?

---

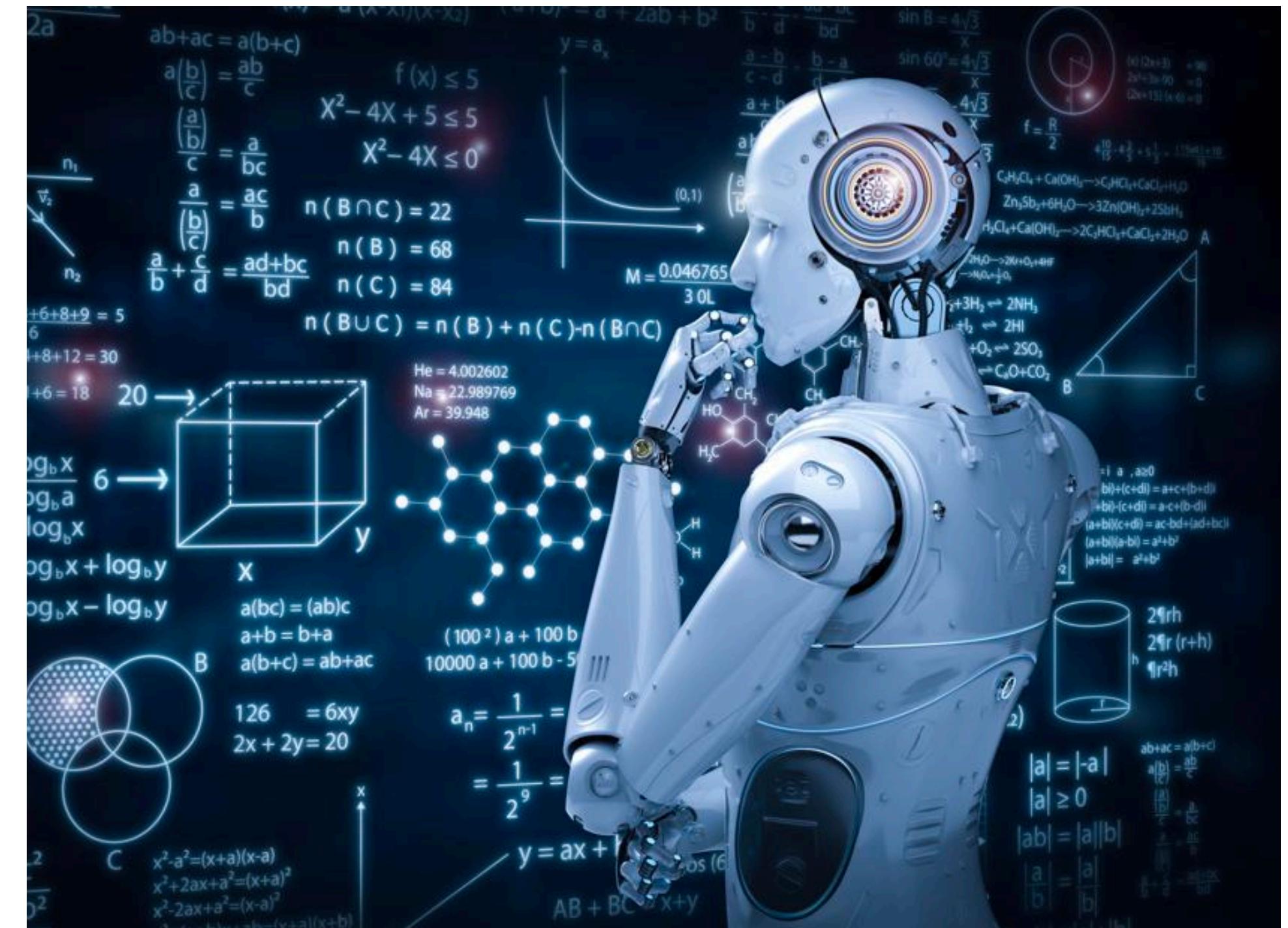


- ~ 1950-1980 Symbolic AI. Rules + Data → Answers
- ML: Data + Answers → Rules
- Ability to learn from past data to make future decisions

# What this field has achieved so far?

---

- Near-human-level image classification
- Near-human-level speech transcription
- Near-human-level handwriting transcription
- Dramatically improved machine translation
- Dramatically improved text-to-speech conversion
- Google assistant - Alexa
- Near-human-level autonomous driving
- Superhuman game-playing (AlphaGo, Alphazero, AlphaStar, etc...)



This field is constantly evolving: some of the major AI-powered research milestones in the world

---



## New AI Model Translates 200 Languages, Making Technology Accessible to More People

July 6, 2022

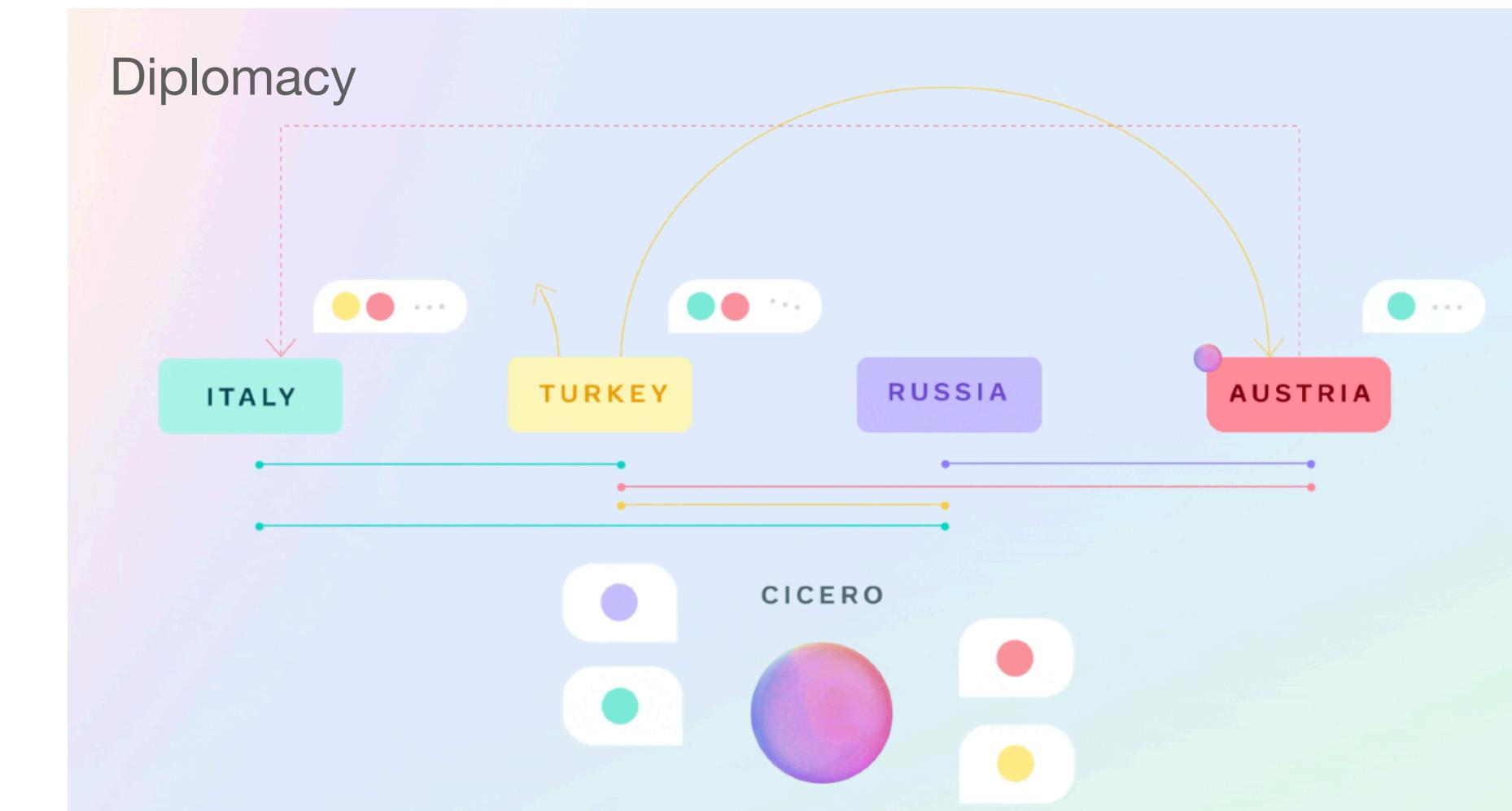
Meta AI Introduces 'Make-A-Video': An Artificial Intelligence System That Generates Videos From Text

By [Ashish kumar](#) - October 3, 2022

RESEARCH

## CICERO: An AI agent that negotiates, persuades, and cooperates with people

November 22, 2022



# This field is constantly evolving: some of the major AI-powered research milestones in the world

---



 Research

## Discovering novel algorithms with AlphaTensor

October 5, 2022

Article | [Open Access](#) | [Published: 15 July 2021](#)

## Highly accurate protein structure prediction with AlphaFold

P How do I build an intermediate python course

 To build an intermediate Python course, you can follow these steps:

1. Determine the target audience for your course. This will help you tailor the content to the appropriate level and focus on the topics that will be most relevant and useful to your students.
2. Outline the main topics that you want to cover in the course. Some potential topics for an intermediate Python course might include object-oriented programming, error handling, data manipulation and visualization, and working with external libraries and APIs.
3. Create a lesson plan for each topic, including a list of objectives, a list of required materials, and a detailed breakdown of the content that you will cover in each lesson.
4. Prepare any necessary teaching materials, such as slides, code examples, and exercises for your students to work on.
5. As you teach the course, be sure to incorporate plenty of hands-on practice and interactive activities to help your students learn the material more effectively.
6. Assess your students' progress through quizzes, exams, and projects, and provide feedback to help them learn and improve.
7. Stay up-to-date on new developments in the Python ecosystem and incorporate relevant updates into your course as needed.

I hope this helps! Let me know if you have any other questions.

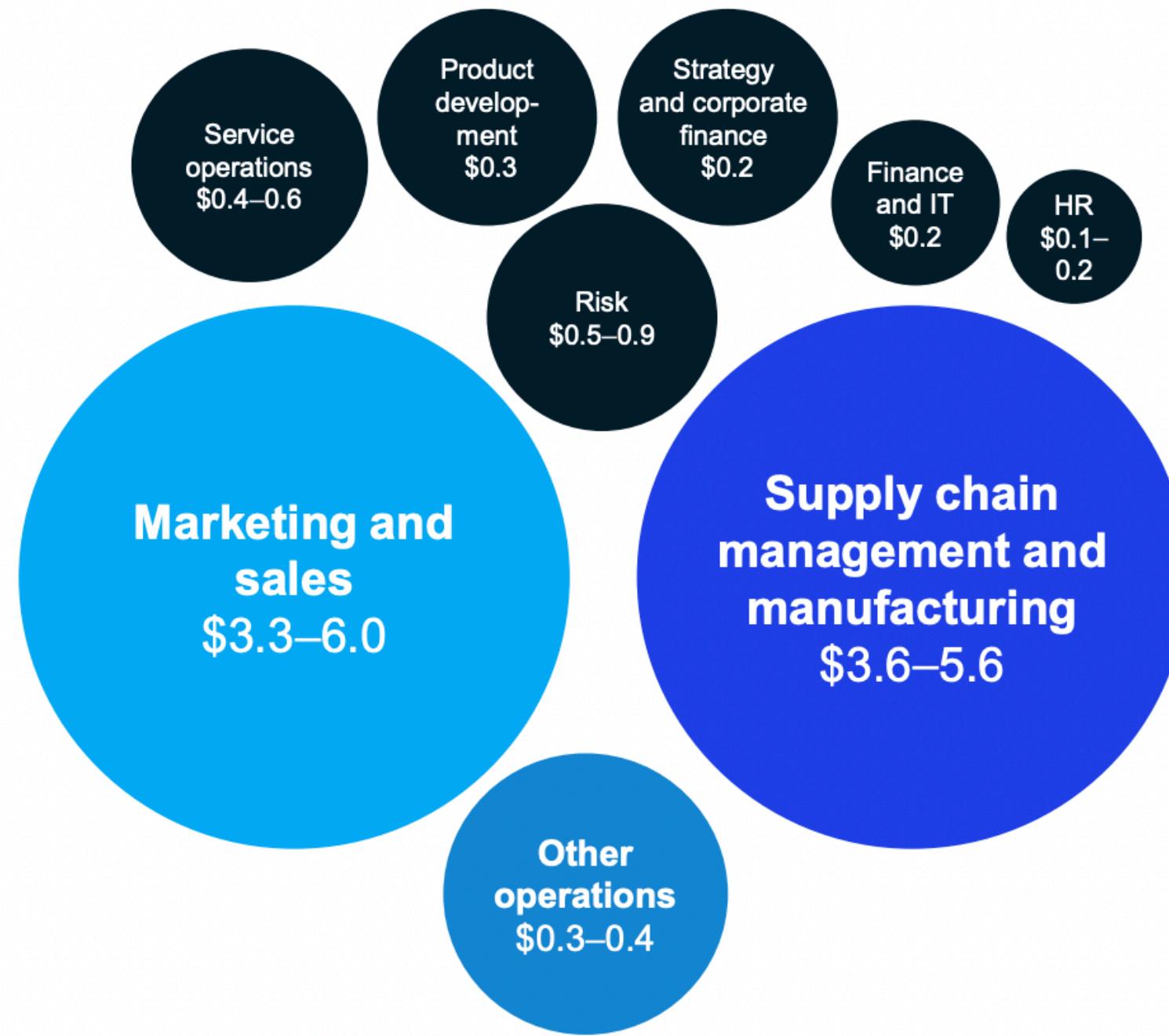
# This disruption applies to any field. Business in general

---

The potential value at stake from AI is  
\$10 to \$15 trillion ...

Global annual potential, forecast

Value at stake, \$ trillion

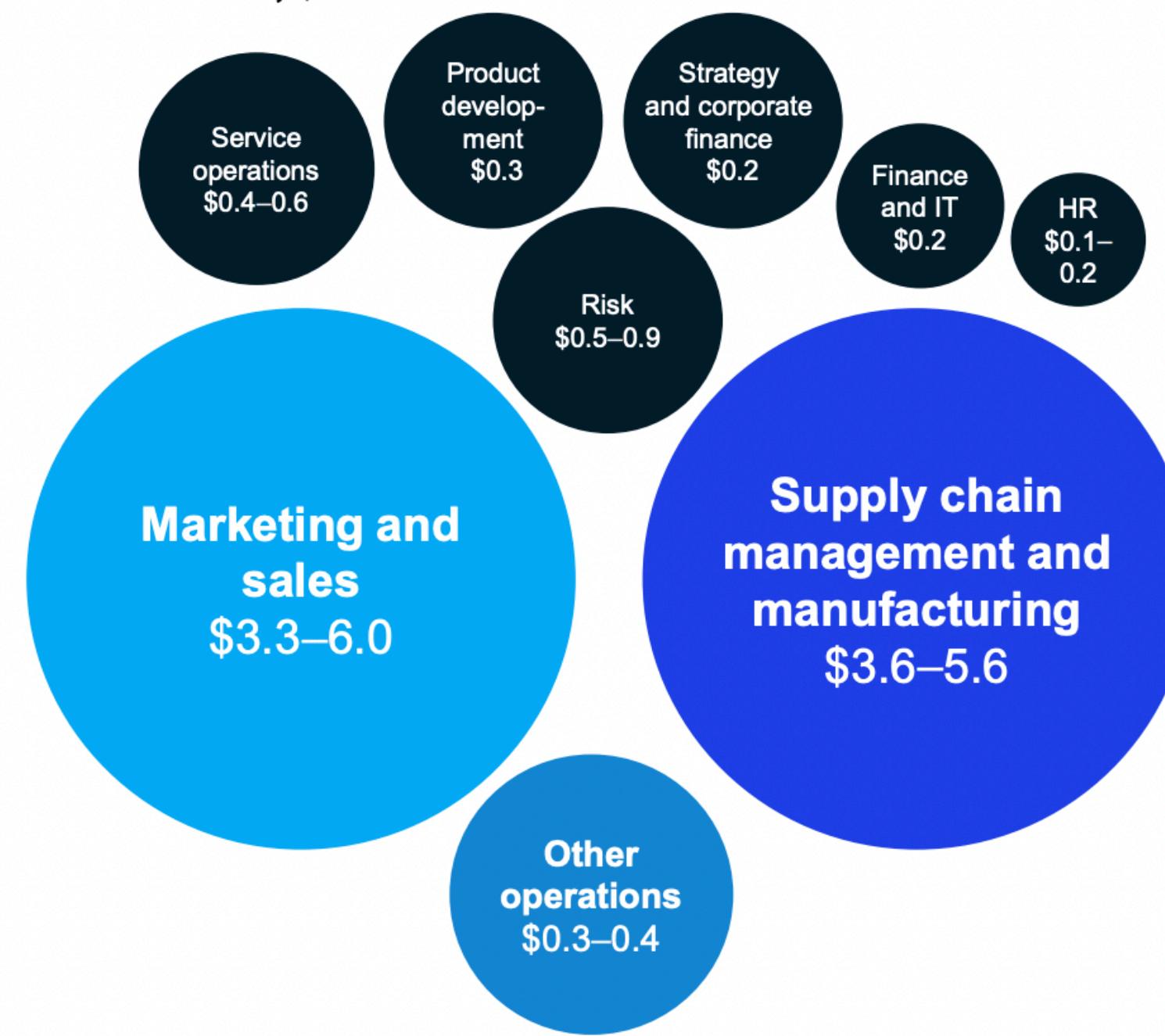


# This disruption applies to any field. Business in general

The potential value at stake from AI is \$10 to \$15 trillion ...

Global annual potential, forecast

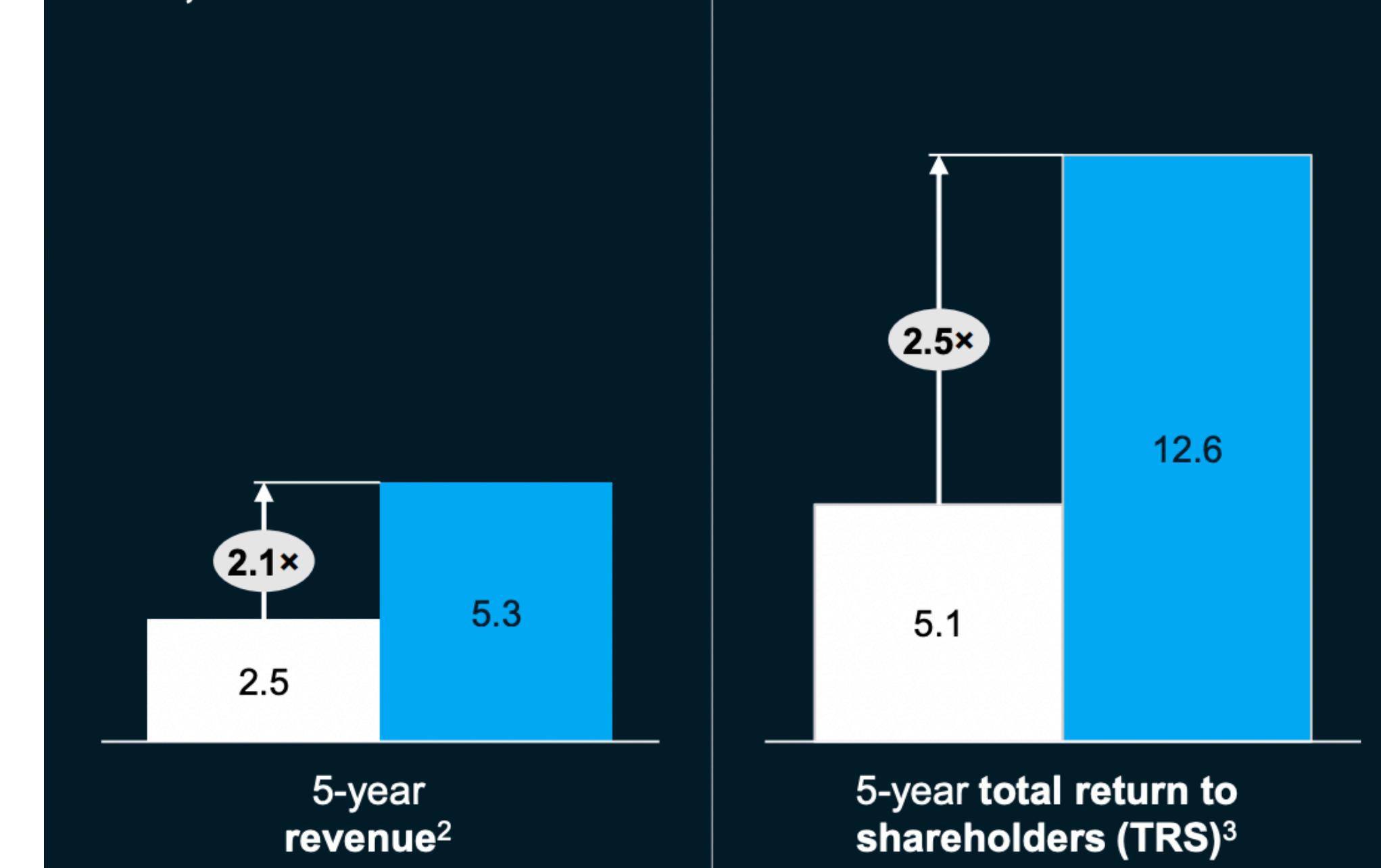
Value at stake, \$ trillion



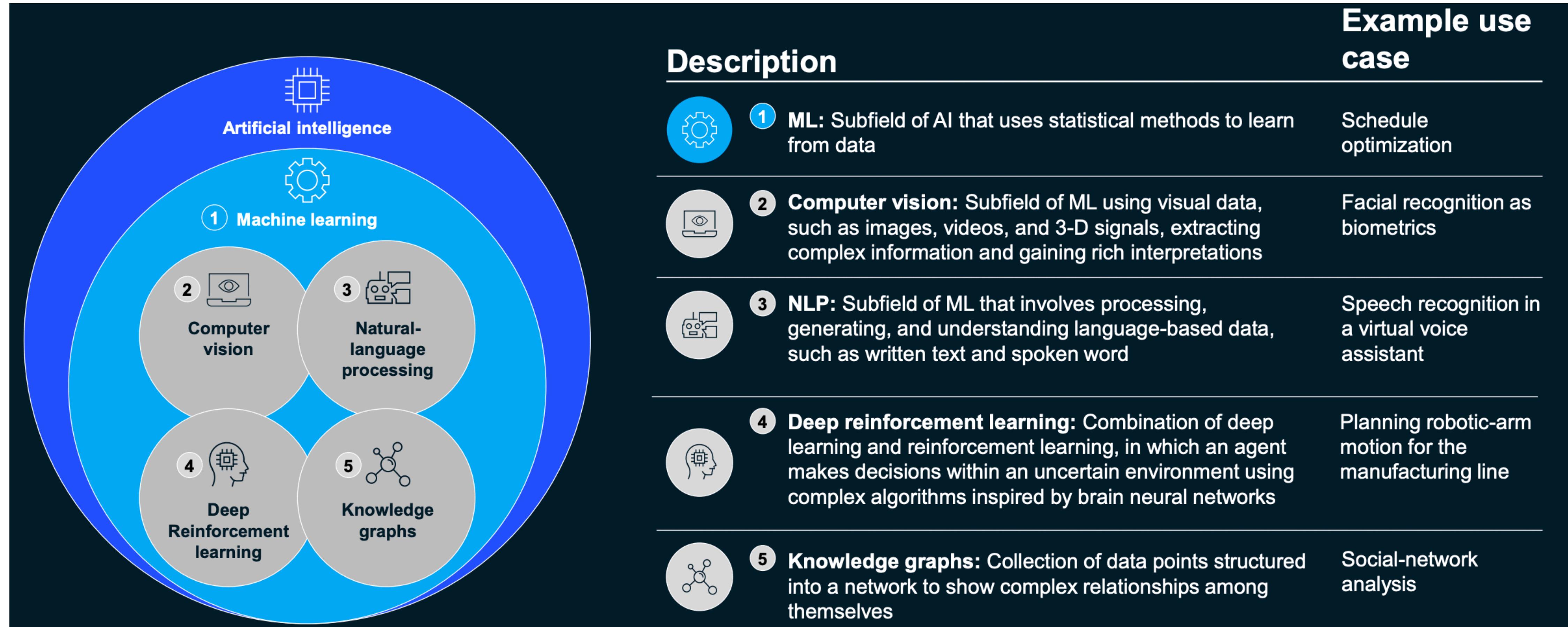
... and leaders adopting AI exhibit stronger financial performance

AI maturity and financial performance ■ Others ■ Analytics leaders<sup>1</sup>

CAGR, %



# This disruption applies to any field. Business in general



# My Take

---

- AI is changing the world and you will have to live in it
- Chances are your work may be affected by it. It is on you to start early with it.
- I think it should be a must to (Harvard HBS):
  - Know about it
  - Understand what it is useful for
  - How it applies to companies now
  - The implications that can arise



# Topic 3: The backbone of this new world

# What is driving this AI revolution?

## Advances in Hardware



CPU

GPU

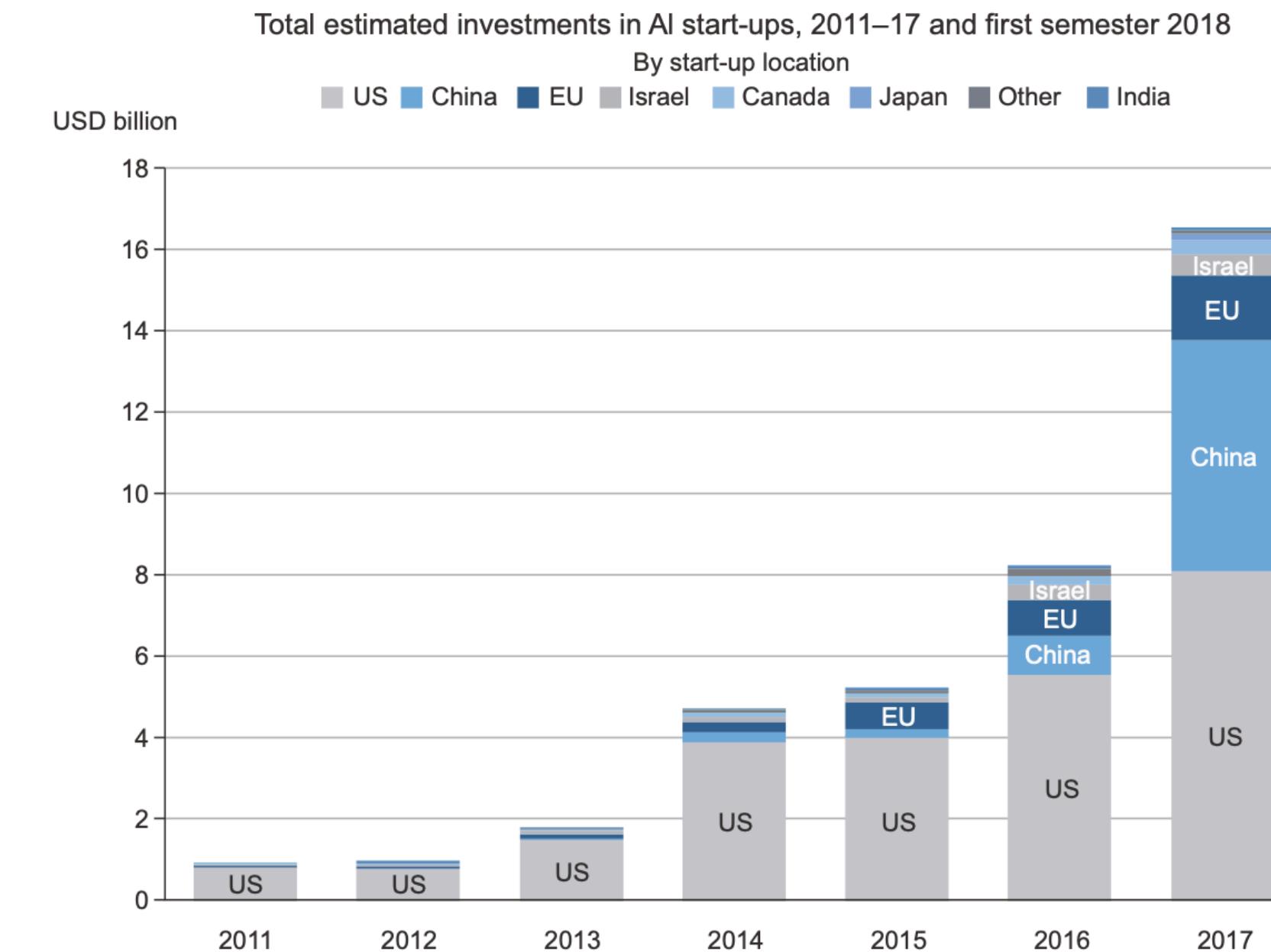
TPU

## Cloud Computing



Google Cloud

## Investment



## Research



# Programming languages are behind this revolution as well

---

- We need to tell the computers to do all of these things
- Programming languages allow us to **communicate with computers**
- A programming language is a **notation system** for writing **computer programs**



# What is Python?

---



- Versatile, and powerful **programming language**
- Concise, and **easy to learn**, use and read
- **Large python community** (Widely used)
- Large number of libraries (Big data, Machine Learning)
- **High demand** in the job market

```
import tensorflow as tf
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.applications.inception_v3 import InceptionV3

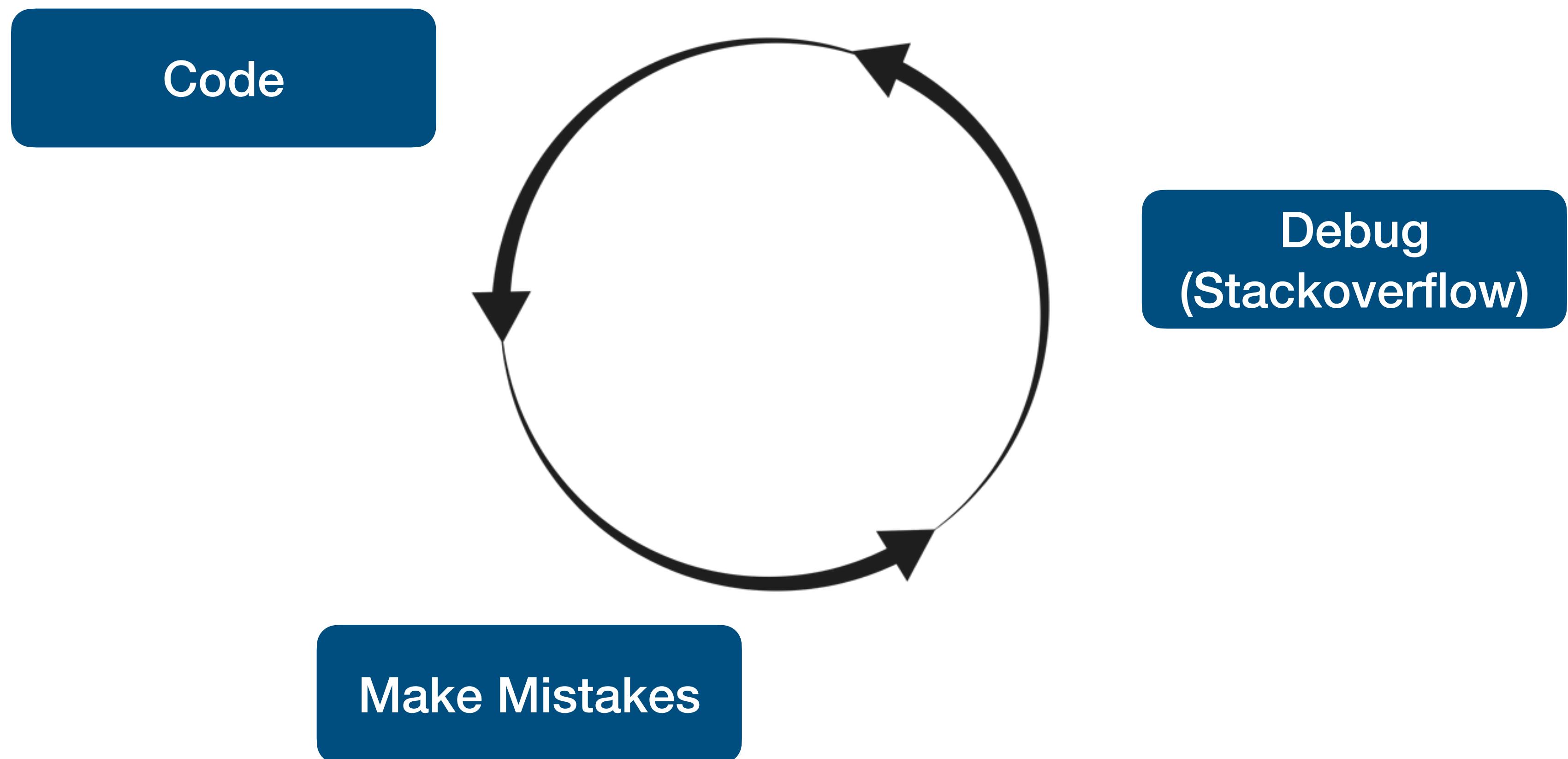
class ConvBlock(Model):
    def __init__(self, filters, kernel_size):
        super(ConvBlock, self).__init__()
        self.conv = Conv2D(filters, kernel_size, padding="same",
                          kernel_initializer="lecun_normal")
        self.bn = BatchNormalization()
        self.relu = ReLU()

    def call(self, inputs):
        x = self.conv(inputs)
        x = self.bn(x)
        x = self.relu(x)
        return x
```

# How do we learn it?

---

## 1. Learn by doing (The coding cycle)



# How do we learn it?

---



COURSES

ABOUT

## The online coding school that invests in you

Train remotely to become a software engineer or data scientist and pay nothing upfront until you are earning \$50k or more.

[START APPLICATION](#)



# Python Backbone

- Variables
- Programming mind (indentation)
- Lists, Dictionaries
- Loops: for, if, elif, else, while
- Functions

```
In [7]: my_list = [1, 5, 12, 91, 102]
my_list_length = len(my_list)
for i in range(0,my_list_length):
    print(i, my_list[i] * my_list[i])
```

```
0 1
1 25
2 144
3 8281
4 10404
```

# Python Libraries

- Pandas
- Numpy
- Scikit Learn
- Tensorflow
- Keras



# Let's start!

---

- We will use Jupyter Notebooks (JN) to code in python
- JN is a web-based interactive computing platform.
- The notebook combines live code, equations, narrative text, and visualizations, to provide a friendly user interface to code and share our work.
- Start your first notebook and let's code!

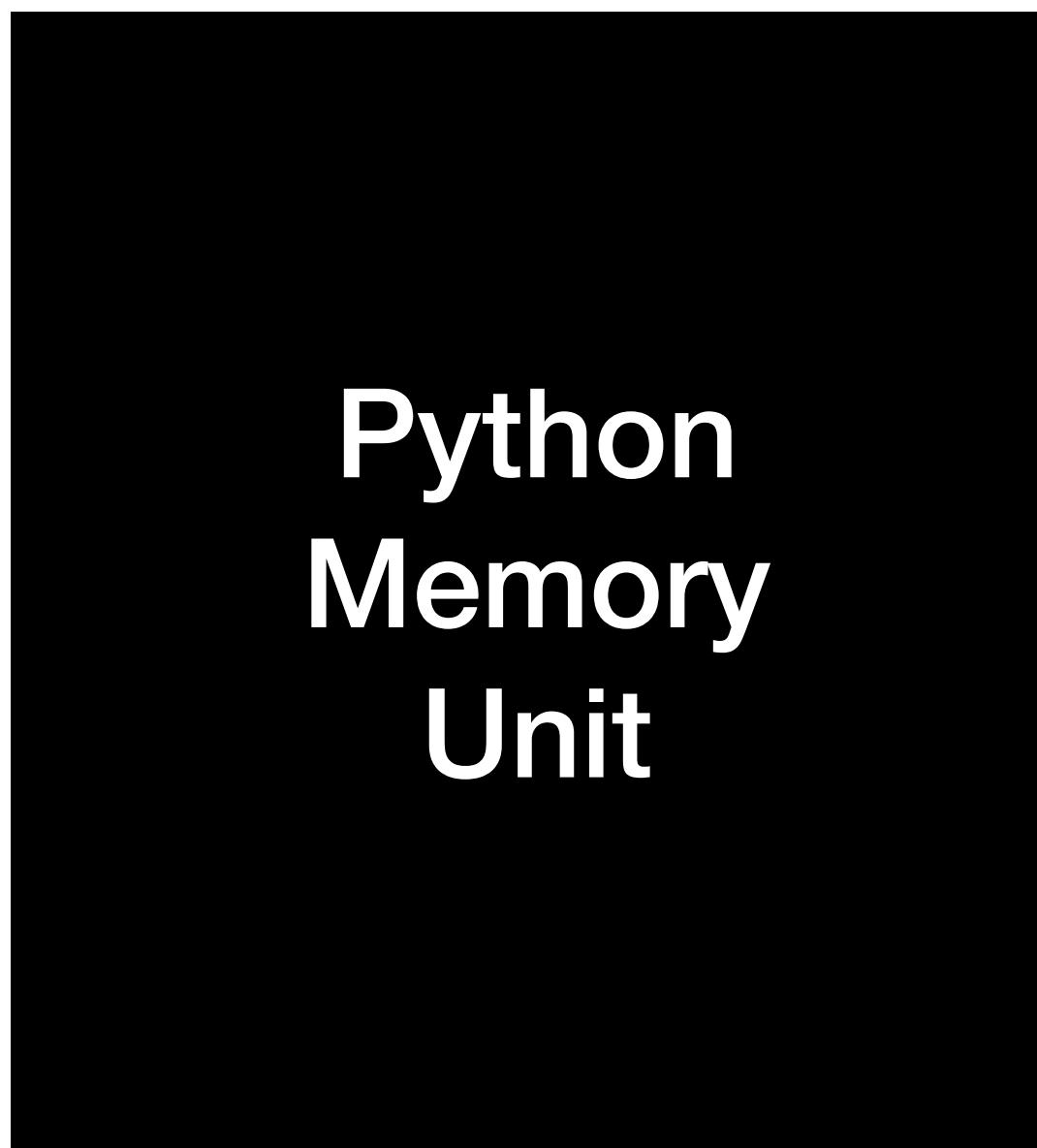


# Topic 4: Python Basics Recap

# Variables

---

- What is a variable? A place for storing data values (Numbers, text, DataFrames, Lists, dictionaries...)
- Types of variables: float, int, string/text, boolean, tuple, dict



# Tips for Variables

---

- Name your variables properly
  - Not long names but expressive (meaningful names related to the variable)
  - Allows you to recognize what the variable stores
- Example: **df** is a convention for naming a data frame. Why not: **my\_data**

Instead, you can assign variables a value like below and then this will be stored:

```
In [4]: a = 20
In [5]: b = 10
In [6]: print(a, b)
```

20 10

```
In [8]: type(b)
Out[8]: int

In [9]: e = 'Hello'
In [10]: type(e)
Out[10]: str

In [11]: f = 1.0
In [12]: type(f)
Out[12]: float
```

# Operations with variables

---

- Python can essentially be a calculator
- Regular operators: +, -, /, \*, ^, \*\*, %
- The results from operations can be saved as new variables

```
In [16]: a * b
```

```
Out[16]: 200
```

```
In [17]: a + b
```

```
Out[17]: 30
```

The result from operations can be saved as a new variable.

```
In [18]: c = a + b  
print(c)
```

```
30
```

```
In [19]: g = c * 10 * a
```

```
In [20]: print(g)
```

```
6000
```

```
In [21]: g / 10
```

```
Out[21]: 600.0
```

```
In [22]: g % 10
```

```
Out[22]: 0
```

# Lists

---

- One of the building blocks of Python for storing information
- A way of storing multiple items in a single variable

```
Sports = ['Football', 'Basketball', 'Hockey']
```

```
Students = ['Iñigo', 'Tomas', 'Margarita', 'Antonella']
```

- List indexing: Students[0]
- List operations: *append, remove, count...*

# Exercise

---

- Create a list that you like, append an element and output its second element

# Dictionaries

---

- Another building block of Python for storing information
- Dictionaries are used to store data values in key-value pairs
  - `my_dict = { key : value}`
  - To index dictionaries: `my_dict[key]`
  - List operations: *update, pop, keys, values*

```
my_dict = {
    'Players': ['Messi', 'Ronaldo', 'Maradona'],
    'Goals' : [10, 8, 6]
}
```

```
my_dict
{'Players': ['Messi', 'Ronaldo', 'Maradona'],
 'Goals': [10, 8, 6]}
```

```
my_dict['Players']
['Messi', 'Ronaldo', 'Maradona']
```

```
my_dict.update({'Assists' : [12, 3, 5]})
```

```
my_dict
{'Players': ['Messi', 'Ronaldo', 'Maradona'],
 'Goals': [10, 8, 6],
 'Assists': [12, 3, 5]}
```

# Exercise

---

- Create a dictionary, update it with a new key-value pair and print the keys of your dictionary

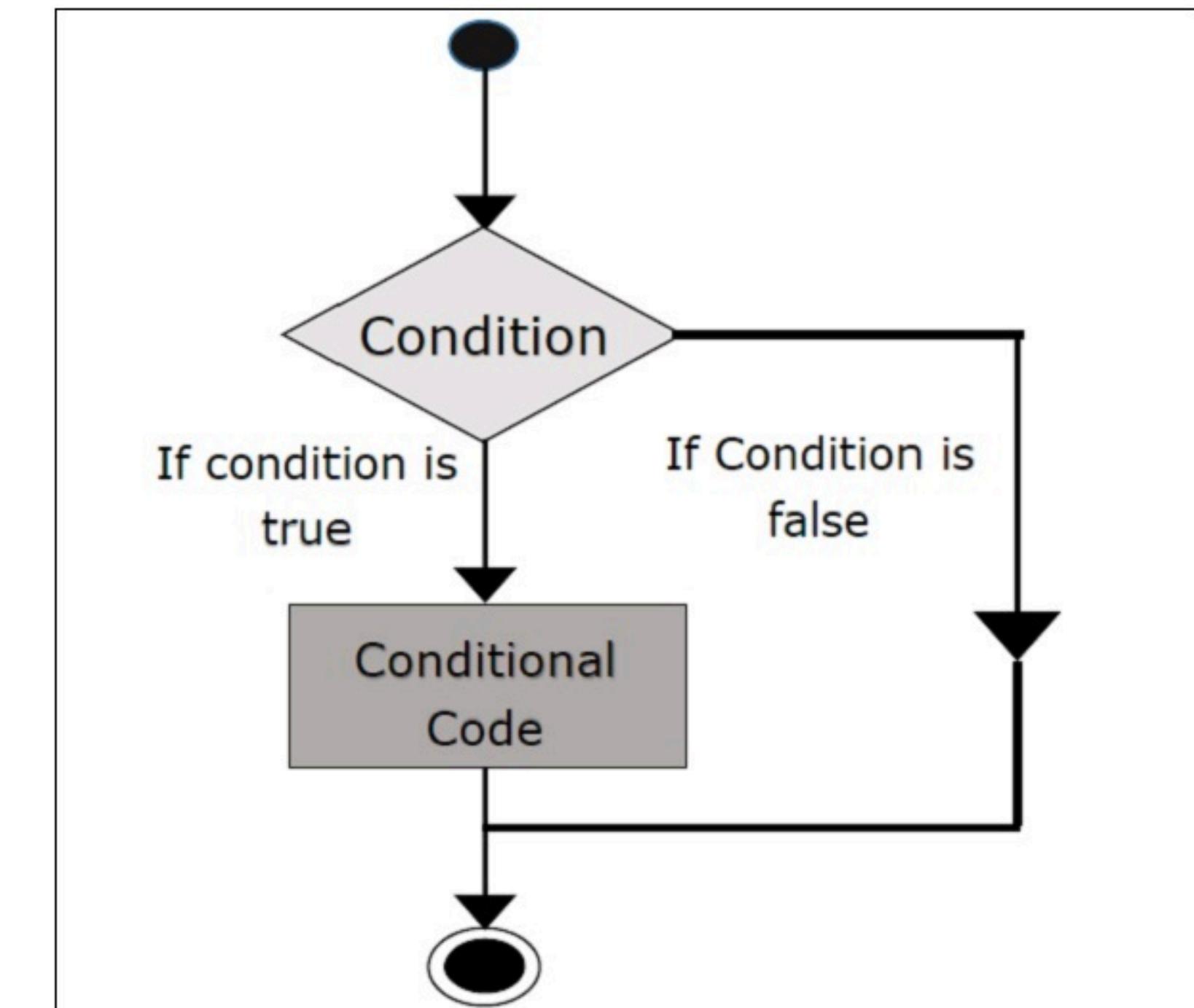
# If-Else statements

---

- Evaluate test expressions and execute the code only when the test condition is true.
- Suppose two variables **a** and **b**. We can assess:
  - $a == b$ ,  $a != b$ ,  $a > b$ ,  $a < b$ ,  $a \leq b$ ,  $a \geq b$

```
In [26]: if a == 10:  
    print('The variable a is equal to 10')  
else:  
    print('The variable a is different than 10')
```

The variable a is different than 10



# Exercise

---

- Evaluate whether your previous list contains more than 3 items.
  - Help: `len(my_list) > 3`
  - If it is: then print yes. Otherwise (else) print no.

# For loops

---

- For loops are used to iterate over sequences like lists, arrays, dictionaries, or others.

```
In [63]: students = ['Margarita', 'Antonella', 'Tomas', 'Iñigo']
for el in students:
    print(el)
```

```
Margarita
Antonella
Tomas
Iñigo
```

```
In [61]: for el in students:
    if el == 'Iñigo':
        print('This student is Iñigo')
    else:
        print('This is other student')
```

```
This is other student
This is other student
This student is Iñigo
This is other student
This is other student
```

# Exercise

---

- Create a list with numbers in it. `my_list = [0, 1, 2, 3]`
- Iterate over it with a for loop, subtract 2 to every number, and print every number at every iteration.
- Plus points if: Start a new list and save the new numbers on it and print it after the for loop

# While Loops

---

- While loops are used to execute a set of statements as long as a condition is true.
- They are not widely used

```
In [70]: aa = 5
while aa != 0:
    print(aa)
    aa -= 1
print(f'The new value of aa is {aa} as it should be')
```

```
5
4
3
2
1
The new value of aa is 0 as it should be
```

# Introduction to functions

---

- A function is a block of code that only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

```
In [78]: def square_root(x):  
    return x ** 0.5
```

```
In [79]: square_root(36)
```

```
Out[79]: 6.0
```

```
In [80]: square_root(16)
```

```
Out[80]: 4.0
```

```
In [71]: def sum_two(my_list):  
    new_list = []  
    for el in my_list:  
        new_list.append(el + 2)  
    return new_list
```

```
In [73]: list1 = [0, 1, 2, 3]  
list2 = [0, -1, -2, -3]
```

```
In [76]: sum_two(list1)
```

```
Out[76]: [2, 3, 4, 5]
```

```
In [77]: sum_two(list2)
```

```
Out[77]: [2, 1, 0, -1]
```

# Exercise

---

- Create a function that computes and returns the square of any number that is passed to the functions.

# Topic 5: Pandas

# What is Pandas?

---

- Python package that provides:
  - Fast
  - Flexible
  - Expressive data structures
- To make working with "relational" or "labeled" data both easy and intuitive



## Notebook lessons

- 1.- Load a csv/tsv/excel file and explore it
- 2.- Basic Data Exploration
- 3.- Column & row operations
- 4.- Groupby, apply & Lambda function

# How do python libraries work

---



Step 0  
Load the package

In [2]: `import pandas as pd`

Python Memory  
Unit

Loads all pandas  
functions

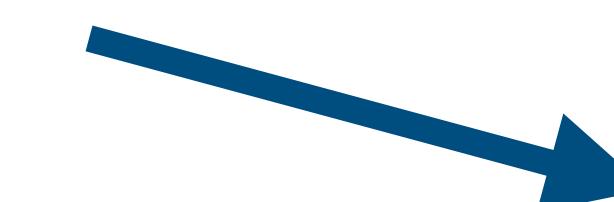
Python  
Programming  
Unit

`import pandas as pd`

# Step 1: Load a csv / tsv / excel file

---

- To work with a dataset in Jupyter Notebook we need to load it.
- We can load any file type: csv, tsv, excel...
- This allows us to start exploring the data



## Step 1 Load dataset in Jupyter notebook

```
In [2]: df = pd.read_csv('../datasets/Dataset_credit_scoring.tsv', sep='\t')
```

```
In [3]: df_excel = pd.read_excel('../datasets/Dataset_credit_scoring.xlsx')
```

```
In [12]: df_excel.head()
```

```
Out[12]:
```

	Occupation	Age	Loan_Salary_ratio	Outcome
0	Industrial	34	2.96	Repay
1	Professional	41	4.64	Default
2	Professional	36	3.22	Default
3	Professional	41	3.11	Default
4	Industrial	48	3.80	Default

Rows

Columns

# Exercise

---

- Download the dataset from the Moodle, save it in the same folder as your jupyter, load it in your Jupyter, and do head() of the first 4 rows.

# Step 2: Data frame exploration

---

- One the most important things is to learn how to explore the data before taking actions
- To do so there are a set of simple functions that can be employed:
  - `columns`, `index`, `describe()`, `info()`, `dtypes`, `head()`, `tail()`

```
df.columns
```

```
Index(['Rank', 'Name', 'Age', 'End_Of_Watch', 'Day_Of_Week', 'Cause',
       'Department', 'State', 'Tour', 'Badge', 'Weapon', 'Offender',
       'Summary'],
      dtype='object')
```

```
df.shape
```

```
(25623, 13)
```

```
df.index
```

```
RangeIndex(start=0, stop=25623, step=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25623 entries, 0 to 25622
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
 ---  -- 
 0   Rank              25623 non-null   object 
 1   Name              25623 non-null   object 
 2   Age               22946 non-null   float64
```

# Exercise

---

- Who is the largest age? And the minimum age?

# Step 3: Operations with Columns & Rows

---

- Knowing how to handle the data and easily access the information is essential.
- Exploration will set our next steps to clean the data and build a model

**What are the main causes of the murder?**

```
df['Cause'].value_counts()
```

Gunfire	12869
Automobile crash	2507
Heart attack	1187
Motorcycle crash	1157
Vehicular assault	976
Struck by vehicle	911
COVID19	812

**Which is the most common rank?** ↗

```
df['Rank'].value_counts()
```

Patrolman	3841
Police Officer	3514
Deputy Sheriff	3215
Officer	1699
Sergeant	1420
...	
Narcotic Agent II	1
Jail Officer	1
Acting Marshal	1
Deputy II	1
Assistant Principal Keeper	1
Name: Rank, Length: 618, dtype: int64	

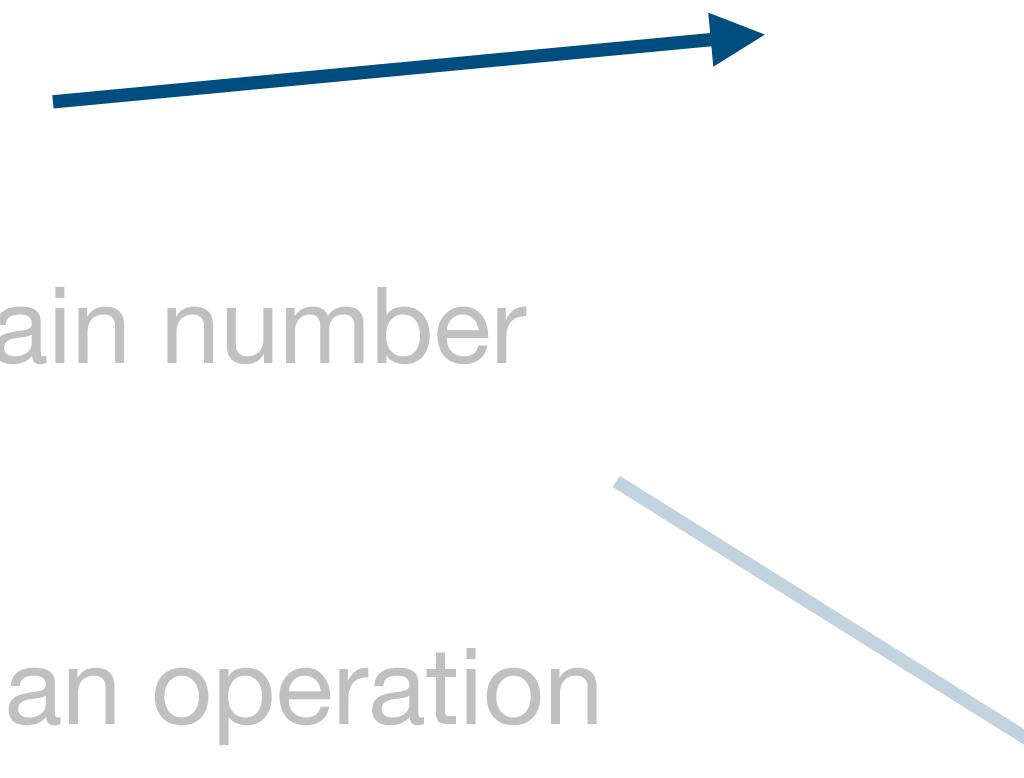
# Exercise

---

- How many causes of the murder do we have?
- How many ranks are in the dataset?
- Which is the 9th most common rank?
- Which department has the most police deaths?

# Step 3: Operations with Columns & Rows

- Separate DataFrames
- Filter Data by selecting a certain number of columns or rows
- Create new columns through an operation



```
less_50 = df[df['Age'] <= 50]
less_50.head(2)
```

	Rank	Name	Age	End_Of_Watch	Day_Of_Week	Cause
3	Marshal	Robert Forsyth	40.0	1794-01-11	Saturday	Gunfire
5	Sheriff	Robert Maxwell	45.0	1797-11-12	Sunday	Gunfire

```
less_50['Age'].median(), less_50['Age'].max()
```

(35.0, 50.0)

```
new_df = df[['Rank', 'Name', 'Age']]
new_df.head()
```

	Rank	Name	Age
0	Constable	Darius Quimby	NaN
1	Sheriff	Cornelius Hogeboom	53.0
2	Deputy Sheriff	Isaac Smith	NaN
3	Marshal	Robert Forsyth	40.0
4	Deputy Sheriff	Robert Berwick	NaN

```
new_df = df.iloc[0:10]
new_df.shape
```

(10, 13)

```
new_list = []

for el in df['Age'].tolist():
    if el > 50:
        new_list.append(1)
    else:
        new_list.append(0)

df['Binary age'] = new_list
df.head(2)
```

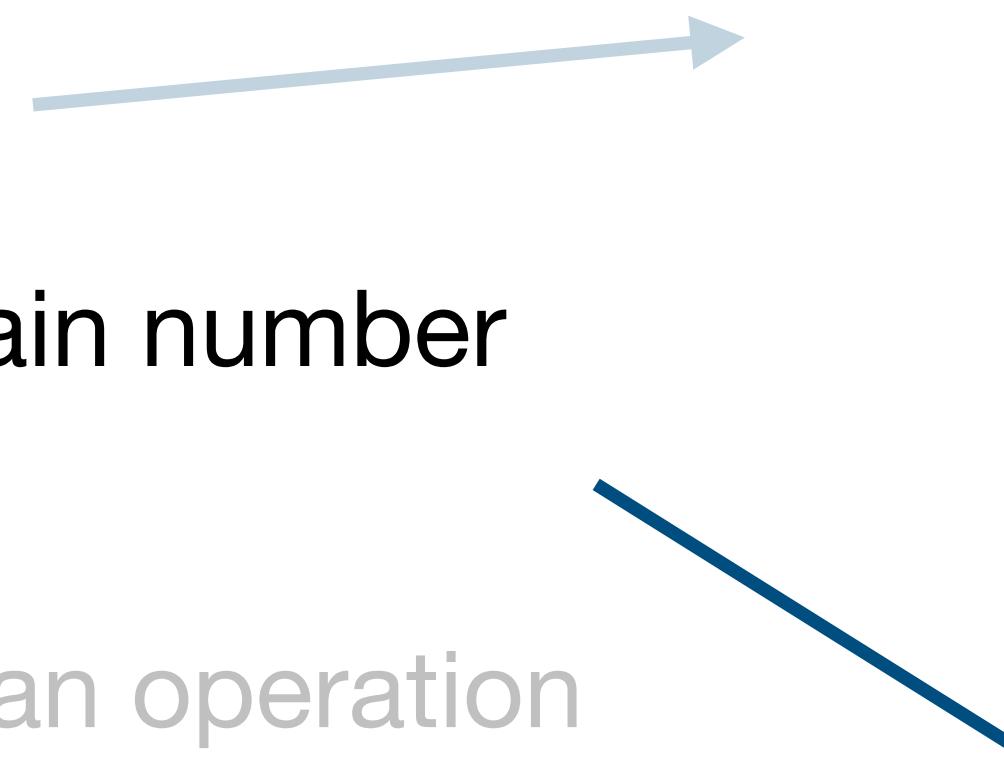
# Step 3: Operations with Columns & Rows

- Separate DataFrames
- Filter Data by selecting a certain number of columns or rows
- Create new columns through an operation

```
new_list = []

for el in df['Age'].tolist():
    if el > 50:
        new_list.append(1)
    else:
        new_list.append(0)

df['Binary age'] = new_list
df.head(2)
```



```
less_50 = df[df['Age'] <= 50]
less_50.head(2)
```

	Rank	Name	Age	End_Of_Watch	Day_Of_Week	Cause
3	Marshal	Robert Forsyth	40.0	1794-01-11	Saturday	Gunfire
5	Sheriff	Robert Maxwell	45.0	1797-11-12	Sunday	Gunfire

```
less_50['Age'].median(), less_50['Age'].max()

(35.0, 50.0)
```

```
new_df = df[['Rank', 'Name', 'Age']]
new_df.head()
```

	Rank	Name	Age
0	Constable	Darius Quimby	NaN
1	Sheriff	Cornelius Hogeboom	53.0
2	Deputy Sheriff	Isaac Smith	NaN
3	Marshal	Robert Forsyth	40.0
4	Deputy Sheriff	Robert Berwick	NaN

```
new_df = df.iloc[0:10]
new_df.shape
```

(10, 13)

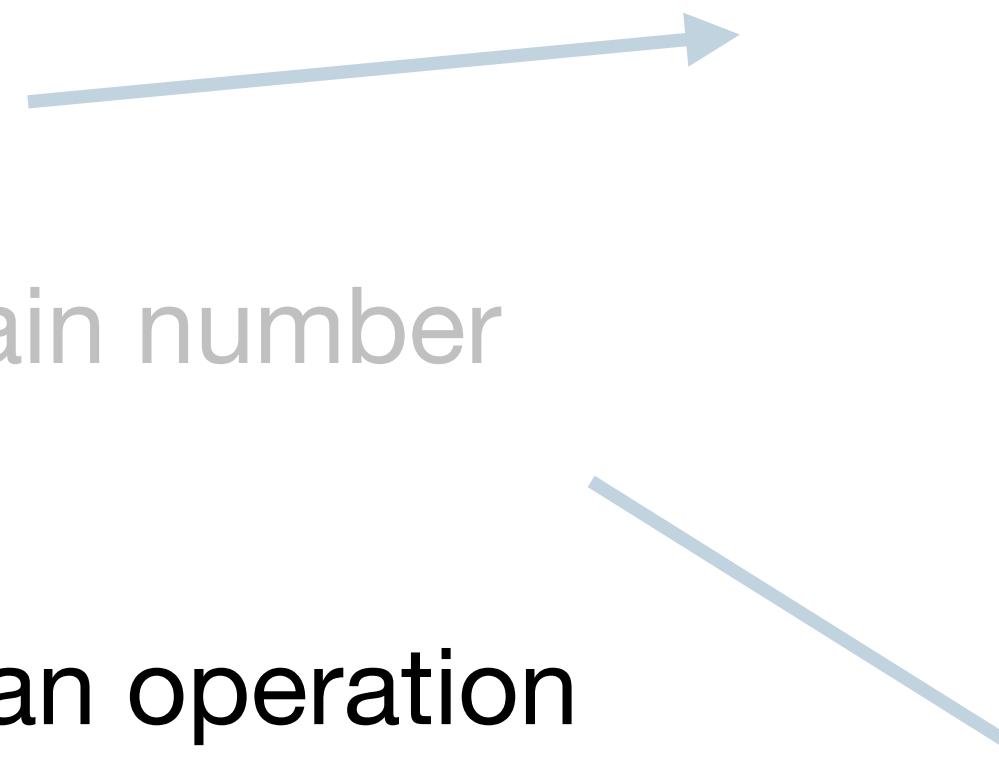
# Step 3: Operations with Columns & Rows

- Separate DataFrames
- Filter Data by selecting a certain number of columns or rows
- Create new columns through an operation

```
new_list = []

for el in df['Age'].tolist():
    if el > 50:
        new_list.append(1)
    else:
        new_list.append(0)

df['Binary age'] = new_list
df.head(2)
```



```
less_50 = df[df['Age'] <= 50]
less_50.head(2)
```

	Rank	Name	Age	End_Of_Watch	Day_Of_Week	Cause
3	Marshal	Robert Forsyth	40.0	1794-01-11	Saturday	Gunfire
5	Sheriff	Robert Maxwell	45.0	1797-11-12	Sunday	Gunfire

```
less_50['Age'].median(), less_50['Age'].max()

(35.0, 50.0)
```

```
new_df = df[['Rank', 'Name', 'Age']]
new_df.head()
```

	Rank	Name	Age
0	Constable	Darius Quimby	NaN
1	Sheriff	Cornelius Hogeboom	53.0
2	Deputy Sheriff	Isaac Smith	NaN
3	Marshal	Robert Forsyth	40.0
4	Deputy Sheriff	Robert Berwick	NaN

```
new_df = df.iloc[0:10]
new_df.shape
```

(10, 13)

# Step 3: Operations with Columns & Rows

- Separate DataFrames
- Filter Data by selecting a certain number of columns or rows
- Create new columns through an operation

```
new_list = []

for el in df['Age'].tolist():
    if el > 50:
        new_list.append(1)
    else:
        new_list.append(0)

df['Binary age'] = new_list
df.head(2)
```

```
less_50 = df[df['Age'] <= 50]
less_50.head(2)
```

	Rank	Name	Age	End_Of_Watch	Day_Of_Week	Cause
3	Marshal	Robert Forsyth	40.0	1794-01-11	Saturday	Gunfire
5	Sheriff	Robert Maxwell	45.0	1797-11-12	Sunday	Gunfire

```
less_50['Age'].median(), less_50['Age'].max()

(35.0, 50.0)
```

```
new_df = df[['Rank', 'Name', 'Age']]
new_df.head()
```

	Rank	Name	Age
0	Constable	Darius Quimby	NaN
1	Sheriff	Cornelius Hogeboom	53.0
2	Deputy Sheriff	Isaac Smith	NaN
3	Marshal	Robert Forsyth	40.0
4	Deputy Sheriff	Robert Berwick	NaN

```
new_df = df.iloc[0:10]
new_df.shape
```

(10, 13)

# Exercise

---

- There was a mistake when introducing the age of the people. We need to correct them. They ask us to create a new column named `corrected_age` that contains the original age minus 2. Can you do that?

# Groupby + apply + Lambda Function

- The groupby operation can be used to group large amounts of data and compute operations on these groups separately.
- "split-apply-combine" operation:
  - Splitting the data into groups based on some criteria
  - Applying a function to each group independently
  - Combining the results into a data structure

```
In [188]: df[df['Outcome'] == 'Repay']['Age'].mean()
```

```
Out[188]: 43.0
```

```
In [189]: df[df['Outcome'] == 'Default']['Age'].mean()
```

```
Out[189]: 38.5
```



```
In [194]: df.groupby('Outcome')['Age'].mean()
```

```
Out[194]: Outcome
           Default    38.5
           Repay     43.0
           Name: Age, dtype: float64
```

```
In [193]: df.groupby('Outcome')['Age'].apply(lambda x: x.sum()/x.count())
```

```
Out[193]: Outcome
           Default    38.5
           Repay     43.0
           Name: Age, dtype: float64
```

Step-by-step in the Jupyter Notebooks

# Exercise

---

- How many different ranks are there if we group by day of the week and age range?

# Topic 6: Documentation

# Datacamp Courses

---

## Python Fundamentals

Are you ready to gain the foundational skills you need to become a Python programmer? In this track, you'll learn the Python basics you need to start on your programming journey, including how to clean real-world data ready for analysis, use data visualization libraries, and even how to write your own Python functions.

Your instructor Hugo will introduce you to how companies worldwide use Python to gain a competitive edge. Through hands-on coding exercises you'll then learn how to store, manipulate, and explore data using NumPy. Then it's time to level-up as you learn how to visualize your data using Matplotlib, manipulate DataFrames and dictionaries using pandas, and write your own functions and list comprehension. Start this track to add these essential Python skills to your data science toolbox.

[Switch Track](#)

Python 15 hours 4 Courses 1 Skill Assessment

## Data Scientist with Python

Gain the career-building Python skills you need to succeed as a data scientist. No prior coding experience required.

In this track, you'll learn how this versatile language allows you to import, clean, manipulate, and visualize data—all integral skills for any aspiring data professional or researcher. Through interactive exercises, you'll get hands-on with some of the most popular Python libraries, including pandas, NumPy, Matplotlib, and many more. You'll then work with real-world datasets to learn the statistical and machine learning techniques you need to train decision trees and use natural language processing (NLP). Start this track, grow your Python skills, and begin your journey to becoming a confident data scientist.

[Resume Track](#)

Python 88 hours 23 Courses 6 Projects 3 Skill Assessments

# Free Books to learn Python & data analytics

---

