PUE ACADEMY Day
8 de mayo - Madrid 2019

#PUEAcademyDay19

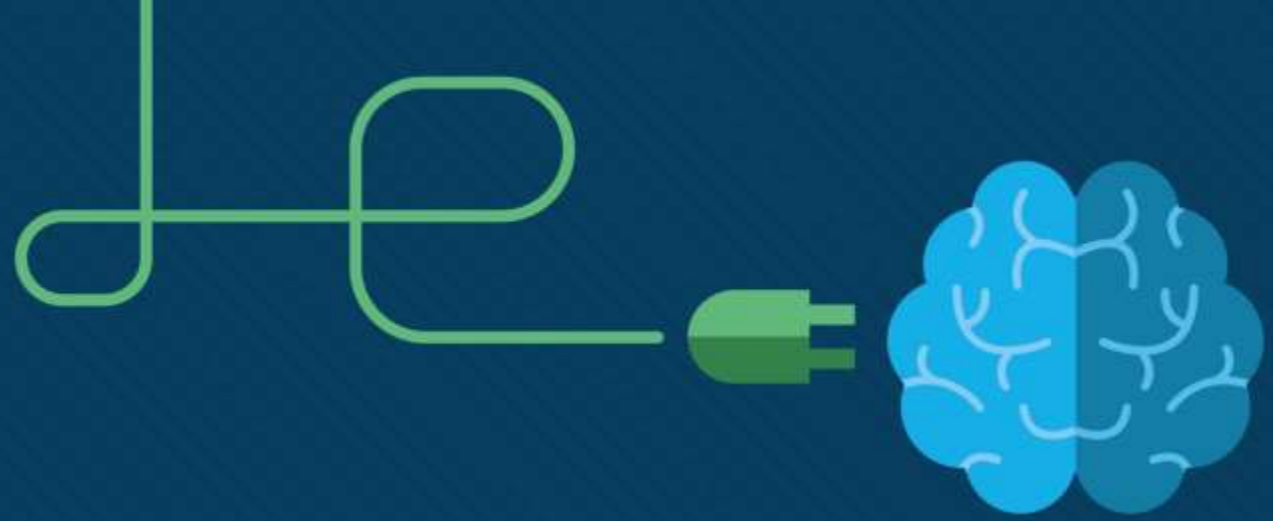
Python Institute

15:35 a 16:45h

Una introducción práctica al procesamiento de imágenes y la visión artificial con Python

Jordi Ariño | Software Developer Manager at PUE

PYTHON INSTITUTE
Open Education & Development Group



PCAP
Programming Essentials in Python

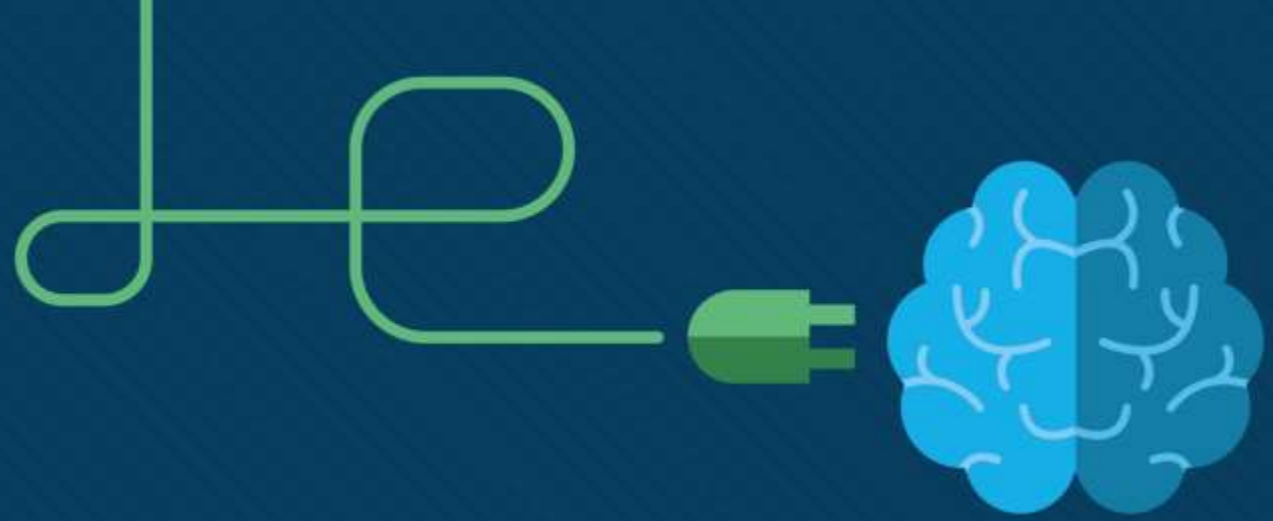
Una introducción práctica al procesamiento de imágenes y la visión artificial con Python

Jordi Ariño

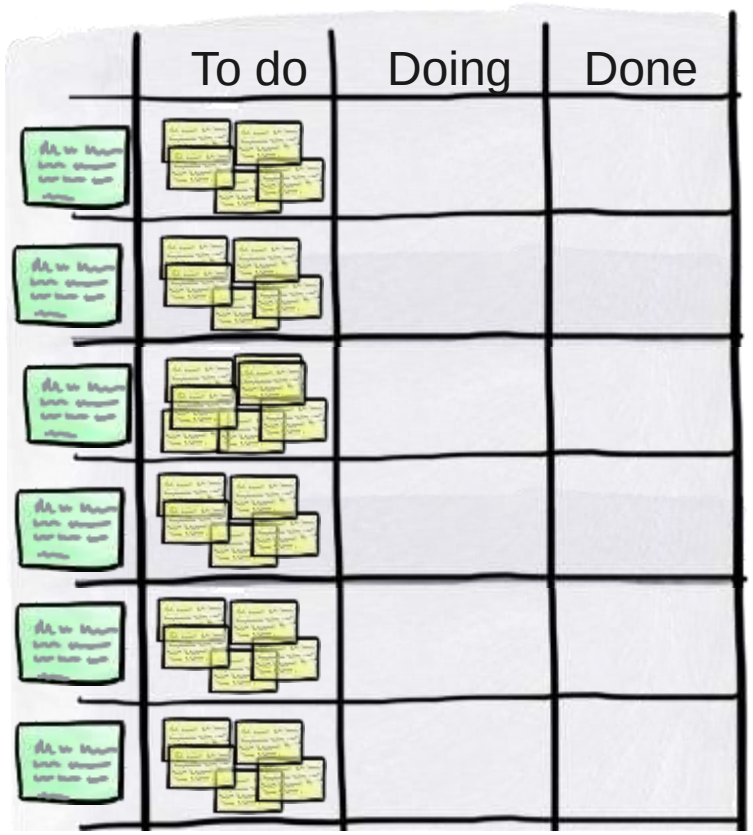
Software Developer Manager at PUE
Agile Software Developer

jordi.arino@pue.es
@jordiAS2K

#PUEAcademyDay19



Introducción



Agenda

1.- Introducción

2.- Python en la actualidad

3.- CNA: Portfolio educativo

4.- PCAP: Programming Essentials in Python

5.- PCAP: Demo del curso

6.- Hands-On Labs

Introducción

Este **workshop** se enmarca dentro de la iniciativa de colaboración entre **Python Institute** y el programa **Cisco Networking Academy**.

Esta colaboración nace con el objetivo de ofrecer una serie de **recursos docentes**, a disposición de los centros Cisco Academy y sus instructores, para la formación de sus alumnos en **Python**.

Entre estos recursos docentes ofrecidos destaca el curso **PCAP**: **Programming Essentials in Python** disponible dentro de la plataforma *NetAcad*.

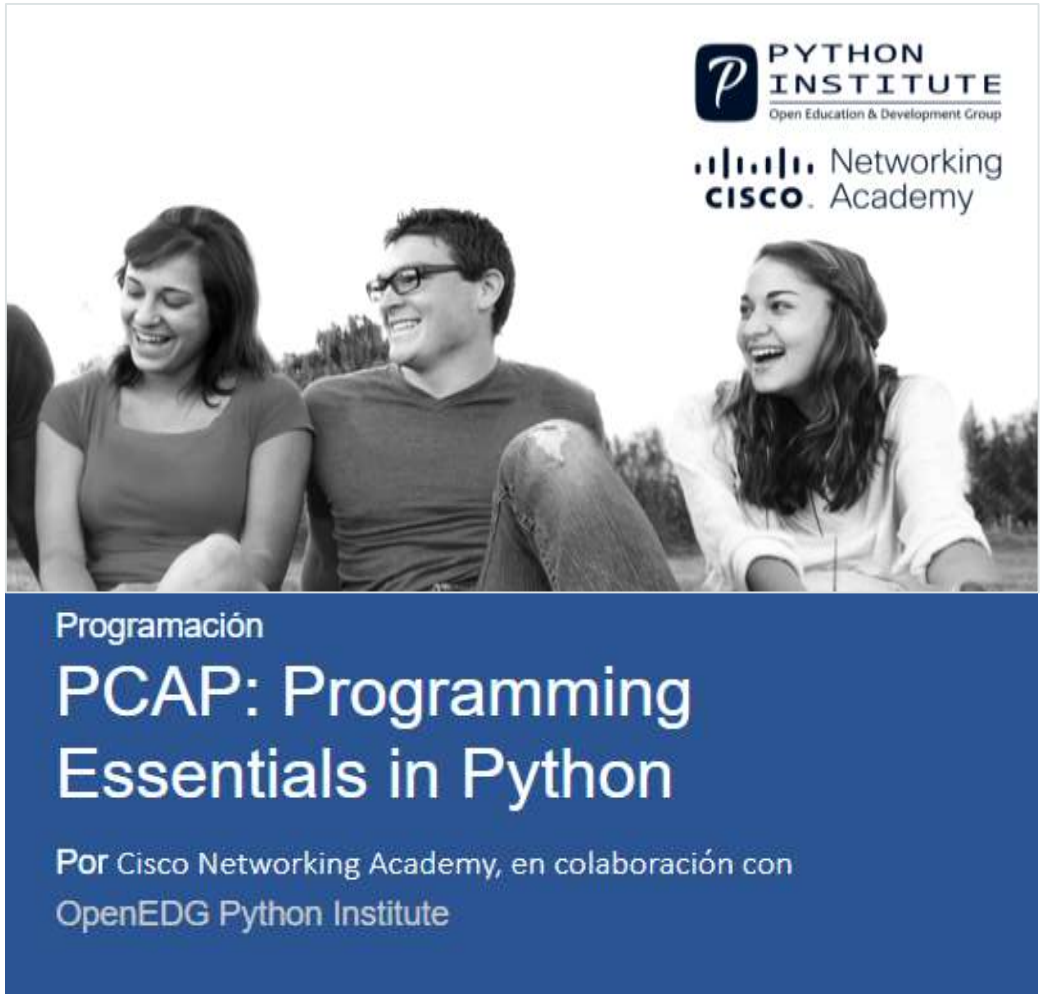
Introducción

Objetivo: Analizar el contenido y formato del curso **PCAP** y ver cómo permite formar a los alumnos en las habilidades necesarias para iniciarse en la programación con Python.

Este curso permite obtener un conjunto de conocimientos fundamentales y habilidades de programación necesarias en los campos de uso de Python: IA, Data Science, Machine Learning, visión por computador, IoT, estadística, …

Como **caso práctico** realizaremos una sencilla introducción de cómo utilizar Python en el campo del procesamiento de imágenes y vídeos dentro de la **visión por computador** o **visión artificial**.

Introducción



Python Institute

Python Institute es un proyecto independiente sin ánimo de lucro desarrollado por el Open Education and Development Group (**OpenEDG**).

Nace con el principal objetivo de promover el lenguaje de programación Python, y ayudar en la formación de una nueva generación de programadores.

Ha desarrollado, en colaboración con Pearson VUE, un **programa de certificación** independiente y *vendor-neutral* en el lenguaje de programación Python.

Iniciativa de colaboración entre **Python Institute** y el programa **Cisco Networking Academy**.

PYTHON INSTITUTE
Open Education & Development Group

Terms and Conditions Policies FAQ Support

ABOUT STUDY RESOURCES CERTIFICATION COMMUNITY CONTACT

LEARN, CERTIFY, PROGRAM YOUR CAREER

Introducción

Programación

PCAP: Programming Essentials in Python

Por Cisco Networking Academy, en colaboración con OpenEDG Python Institute

Certificación PCAP

Nombre: Certified Associate in Python Programming

Código: PCAP-31-02

Versión: Python 3.x

Nivel: Associate

Pre-requisitos: Ninguno

Duración: 65 min

Preguntas: 40 preguntas

Formato: Tipo test (*single/multiple choice*)

Passing Score: 70% (28/40)

Idioma: Inglés

Introducción

Programación

PCAP: Programming Essentials in Python

Por Cisco Networking Academy, en colaboración con OpenEDG Python Institute

Certificación PCAP

Exam block #1
Control and Evaluations (25%)

Exam block #2
Data Aggregates (25%)

Exam block #3
Functions and Modules (25%)

Exam block #4
Classes, Objects and Exceptions (25%)

Introducción

Programación

PCAP: Programming Essentials in Python

Por Cisco Networking Academy, en colaboración con OpenEDG Python Institute

What snippet would you insert in the line indicated below:

```
n = 0

while n < 4:

    n += 1

    # insert your code here
```

To print the following string to the monitor after the loop finishes its execution:

```
1 2 3 4
```

A. `print(n)`

B. `print(n, sep=" ")`

C. `print(n, end=" ")`

D. `print(n, " ")`

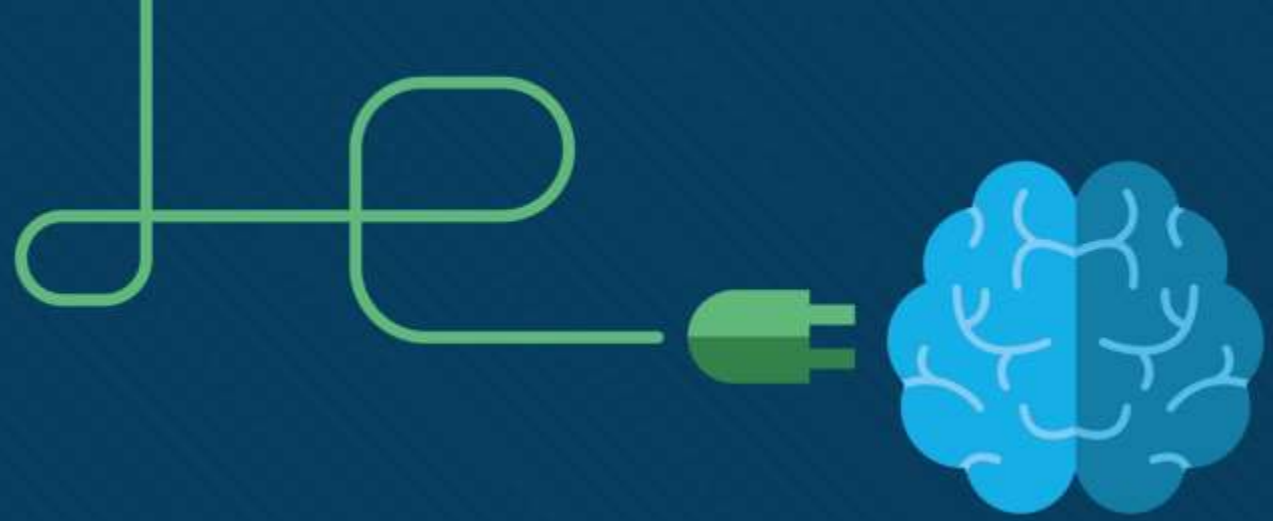
Introducción

Con la inscripción a este workshop habréis recibido acceso, como alumno, al curso **PCAP: Programming Essentials in Python** dentro de la plataforma NetAcad de Cisco.

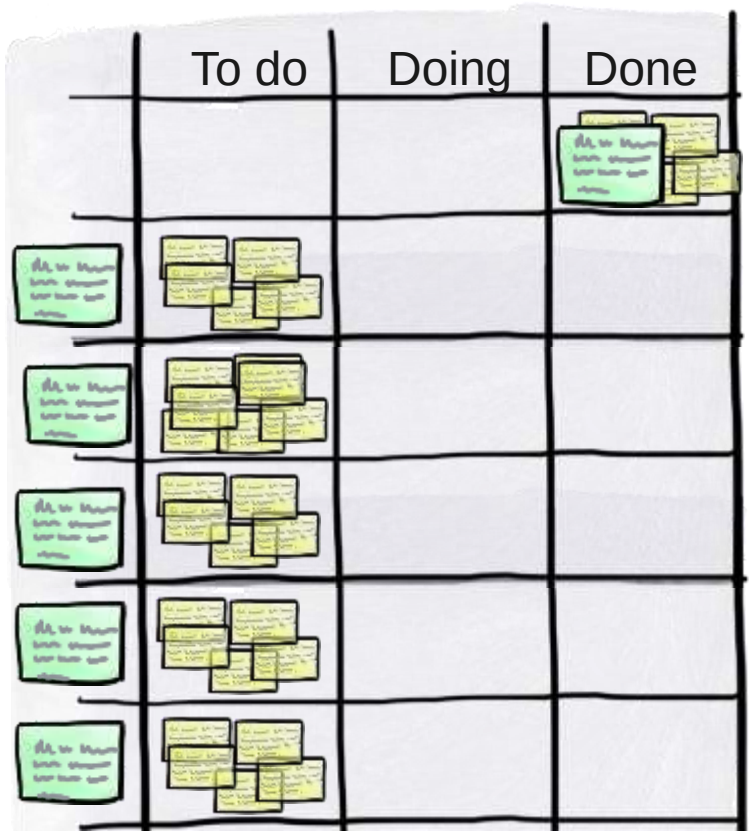
Objetivo: Poder evaluar el curso y los recursos e-learning que Cisco pone a disposición de centros y profesores para la formación de sus alumnos en programación en Python de forma práctica, oficial y reconocida.

Destacar, como valor añadido, que este curso ayuda a los estudiantes en su preparación para la certificación oficial:

- **PCAP – Python Certified Associate Programmer**



Python en la actualidad



Agenda

1.- Introducción

2.- Python en la actualidad

3.- CNA: Portfolio educativo

4.- PCAP: Programming Essentials in Python

5.- PCAP: Demo del curso

6.- Hands-On Labs

Python en la actualidad

A día de hoy, **Python es uno de los lenguajes de programación más populares y extendidos**, adoptado en la mayoría de sectores e industrias.

Ha sido nombrado el **lenguaje del año 2018** por el índice **TIOBE**.

El índice **PYPL** (PopularitY of Programming Language), considera a Python el **lenguaje más popular** y el que ha experimentado **un mayor crecimiento** en estos últimos 5 años (17.1%).

La tendencia del mercado laboral muestra que la demanda de profesionales de Python crece exponencialmente cada año.

Python en la actualidad

TIOBE Programming Community Index

Python en la actualidad

TIOBE Programming Community Index

Apr 2019	Apr 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.035%	-0.74%
2	2		C	14.076%	+0.49%
3	3		C++	8.838%	+1.62%
4	4		Python	8.166%	+2.36%
5	6	↑	Visual Basic .NET	5.795%	+0.85%
6	5	↓	C#	3.515%	-1.75%
7	8	↑	JavaScript	2.507%	-0.99%
8	9	↑	SQL	2.272%	-0.38%
9	7	↓	PHP	2.239%	-1.98%
10	14	⇈	Assembly language	1.710%	+0.05%

Python en la actualidad

TIOBE Programming Community Index

Year	Winner
2018	Python
2017	C
2016	Go
2015	Java
2014	JavaScript
2013	Transact-SQL
2012	Objective-C
2011	Objective-C

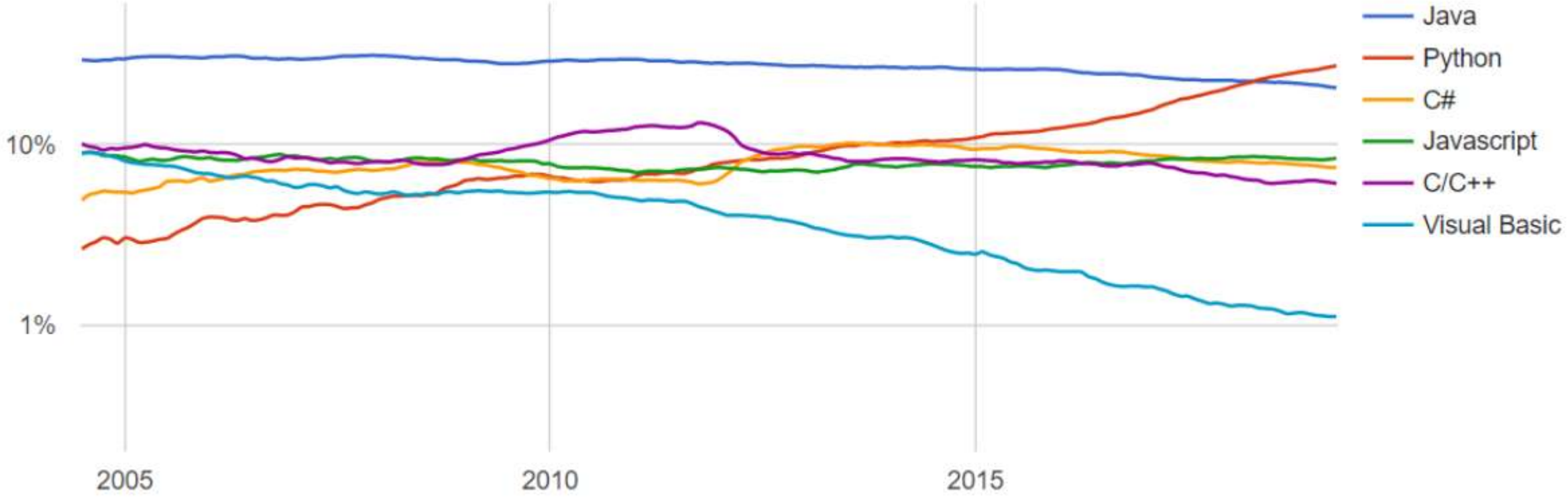
Python en la actualidad

TIOBE Programming Community Index

Programming Language	2019	2014	2009	2004	1999	1994	1989
Java	1	2	1	1	13	-	-
C	2	1	2	2	1	1	1
C++	3	4	3	3	2	2	2
Python	4	7	5	9	26	21	-
Visual Basic .NET	5	10	-	-	-	-	-
C#	6	5	6	7	23	-	-
JavaScript	7	8	8	8	18	-	-
PHP	8	6	4	5	-	-	-
SQL	9	-	-	6	-	-	-

Python en la actualidad

PYPL PopularitY of Programming Language



Python en la actualidad

PYPL PopularitY of Programming Language

Worldwide, Apr 2019 compared to a year ago:

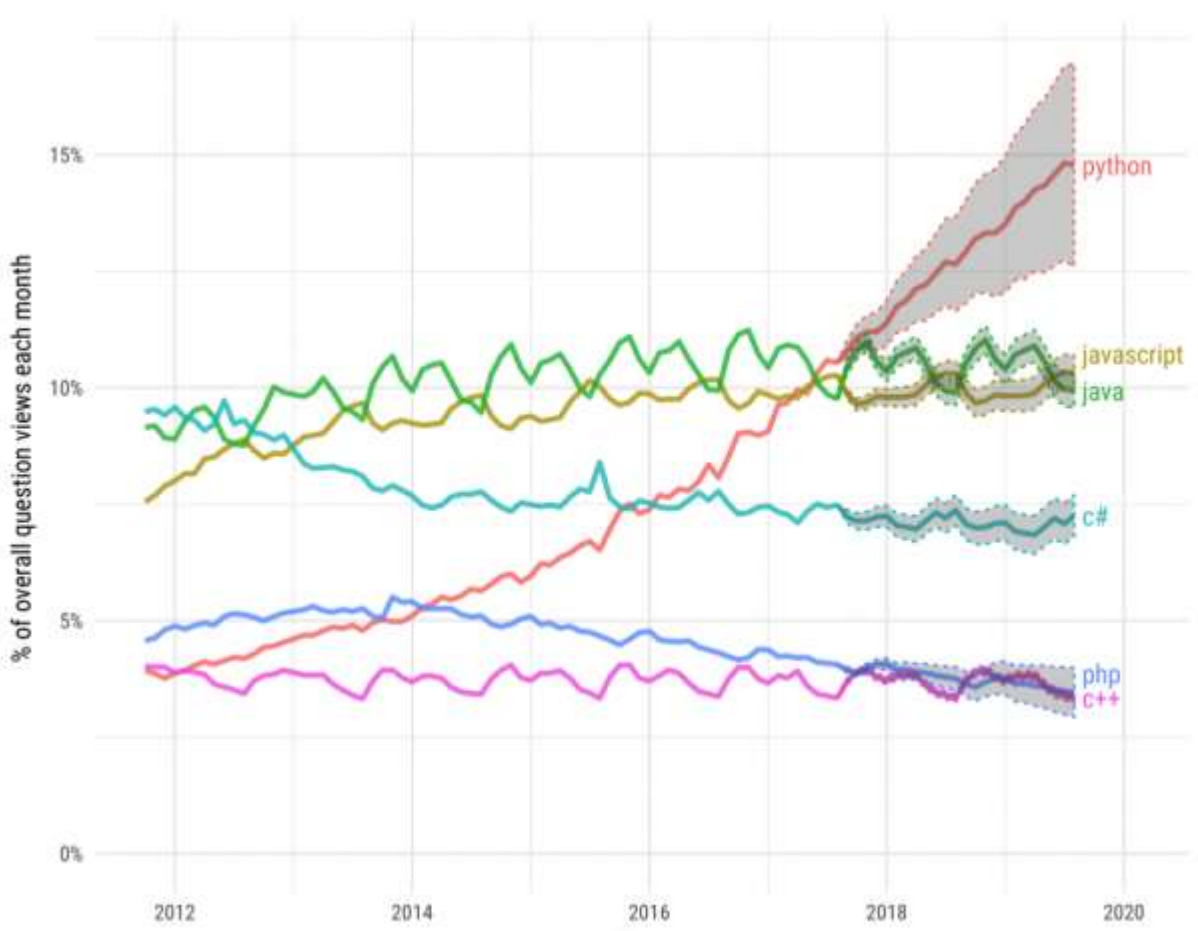
Rank	Change	Language	Share	Trend
1	↑	Python	27.35 %	+5.2 %
2	↓	Java	20.64 %	-1.8 %
3		Javascript	8.4 %	-0.2 %
4	↑	C#	7.45 %	-0.5 %
5	↓	PHP	7.18 %	-1.1 %
6		C/C++	6.08 %	-0.3 %
7		R	4.18 %	-0.1 %
8		Objective-C	2.86 %	-0.8 %
9		Swift	2.47 %	-0.4 %
10		Matlab	2.06 %	-0.3 %

Worldwide, Apr 2019 compared to a year ago:

Rank	Change	Language	Share	Trend
11	↑	TypeScript	1.64 %	+0.2 %
12	↓	Ruby	1.39 %	-0.3 %
13		VBA	1.39 %	+0.0 %
14	↑↑	Kotlin	1.18 %	+0.3 %
15		Scala	1.16 %	-0.0 %
16	↑	Go	1.14 %	+0.3 %
17	↓↓↓	Visual Basic	1.11 %	-0.2 %
18		Perl	0.58 %	-0.2 %
19		Rust	0.54 %	+0.2 %
20		Lua	0.37 %	+0.0 %

Python en la actualidad

Stack Overflow Trends



Python en la actualidad

¿Qué hace que Python sea tan especial?

- Es **fácil de aprender**: el tiempo necesario para aprender Python es más corto que en muchos otros lenguajes.
- Es **fácil de enseñar**: la carga de trabajo de enseñanza es más pequeña que la que necesitan otros lenguajes.
- Es **fácil de entender**; a menudo, también es más fácil entender el código de otra persona más rápido si está escrito en Python.
- Es **fácil de obtener, instalar y desplegar**: Python es gratuito, abierto y multiplataforma.
- Ofrece un gran ecosistema de **paquetes**, **librerías y frameworks**

Find, install and publish Python packages with the Python Package Index

Search projects

Or browse projects

177,638 projects 1,300,352 releases 1,863,365 files 324,283 users

python™ Package Index

The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages.

Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

Trending projects

Trending projects as downloaded by the community

- guillotina-oauth 2.0.0
  guillotina oauth support

New releases

Hot off the press: the newest project releases

- sheet-disk 0.1.1
  Use Google Sheets as a storage device!

The Python Package Index (PyPI)

PyPI has over more than 160,000 packages. Here are some of its functionalities


Graphical User Interfaces


Scientific Computing


System Administration


Image Processing


Web Frameworks


Test Frameworks

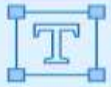

Documentation


Databases


Multimedia

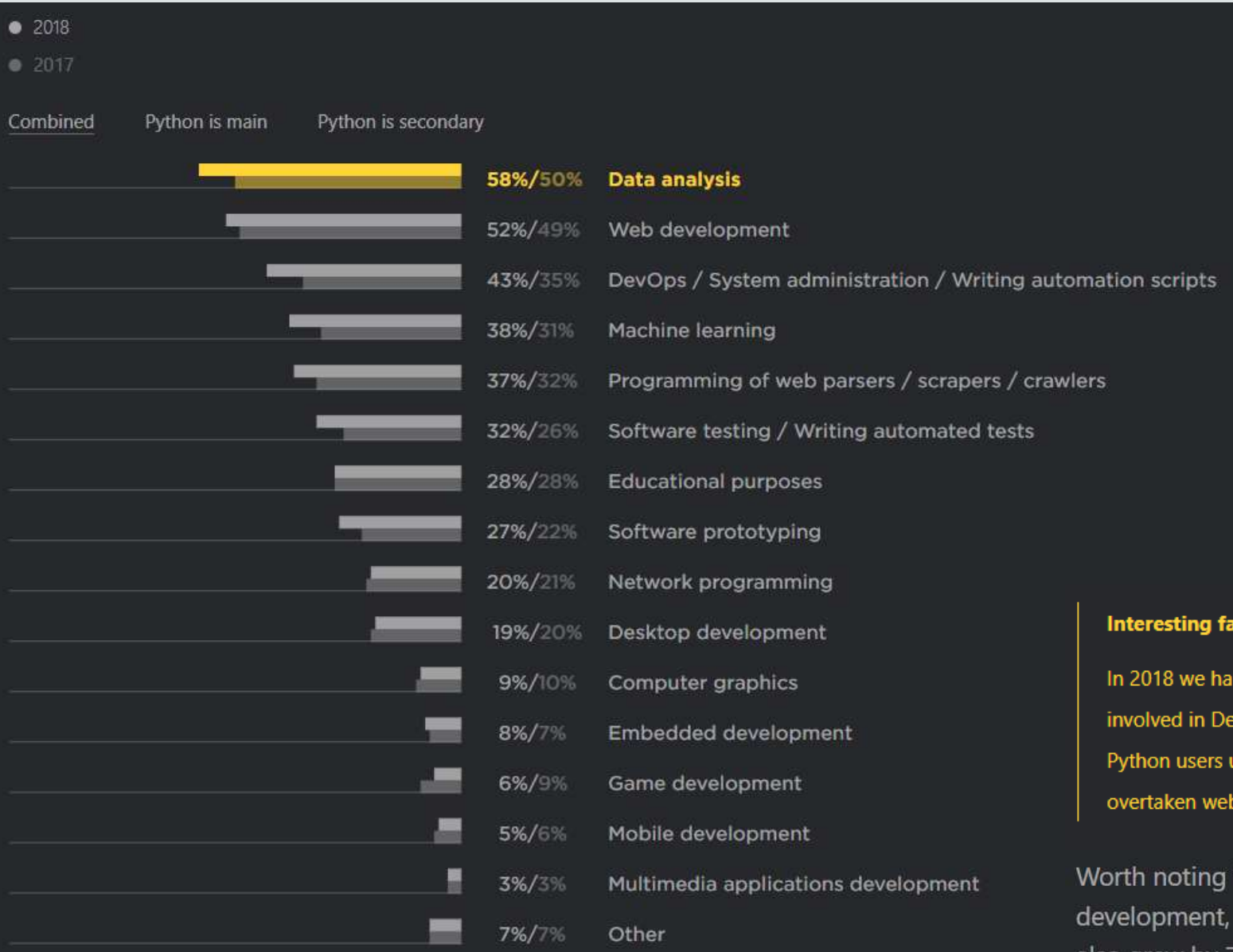

Networking


Automation


Text Processing

SOURCE: https://en.wikipedia.org/wiki/Python_(programming_language)

Python Developers Survey 2018

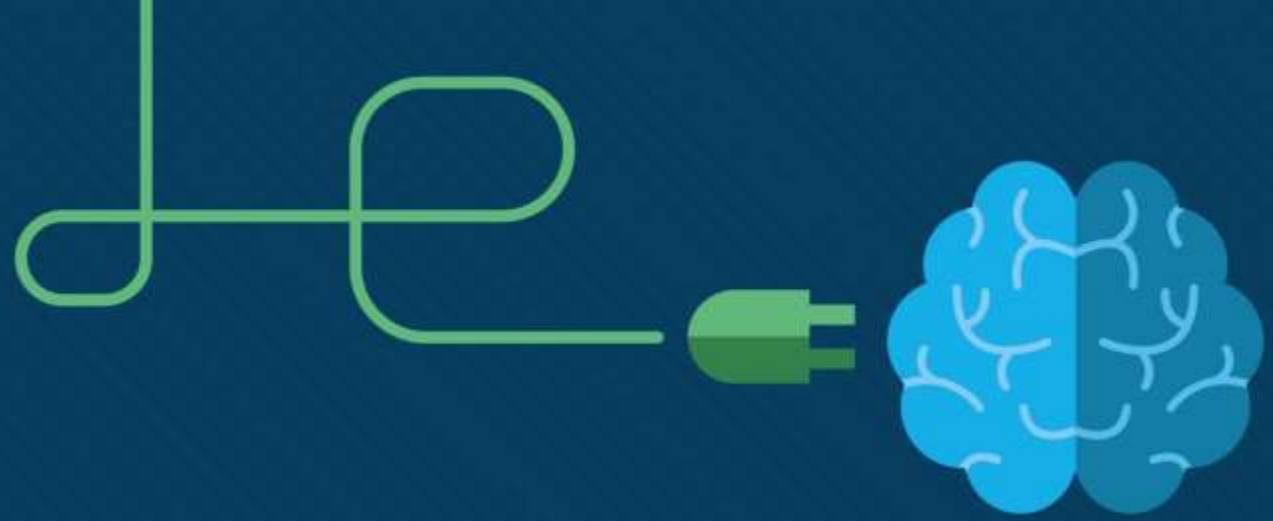


Interesting fact

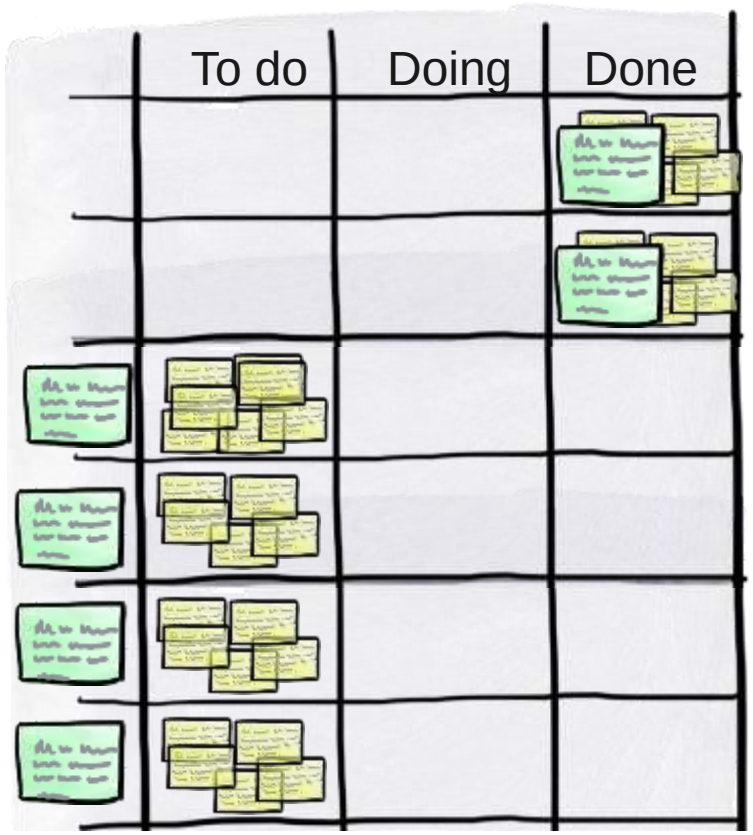
In 2018 we had significantly more respondents specifying they're involved in DevOps (an increase of 8% compared to 2017). In terms of Python users using Python as their secondary language, DevOps has overtaken web development.

Worth noting is that Data analysis has become more popular than Web development, growing from 50% in 2017 to 58% in 2018. Machine learning also grew by 7 percentage points. These types of development are experiencing faster growth than Web development, which has only increased by 2 percentage points when compared to previous year.

16 Famous Companies that uses PYTHON



CNA: Porfolio educativo



Agenda

1.- Introducción

2.- Python en la actualidad

3.- CNA: Portfolio educativo

4.- PCAP: Programming Essentials in Python

5.- PCAP: Demo del curso

6.- Hands-On Labs

El portafolio educativo de Networking Academy

Colaborar para generar impacto

Introduction to Packet Tracer | **Packet Tracer** | **Hackatones** | **Prototyping Lab** | **Prácticas laborales**

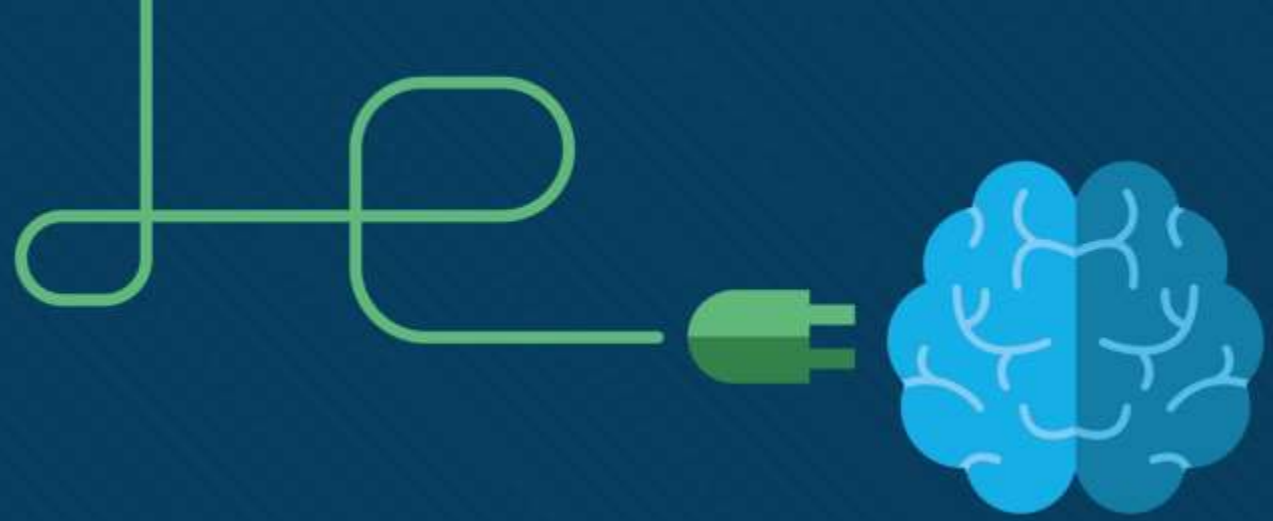
	Exploratorio	Básico	Profesional
Redes		**Networking Essentials** **Mobility Fundamentals** **Talleres de tecnología emergente:** Programabilidad de redes con Cisco APIC-EM*	**CCNA R&S:** Introduction to Networks, R&S Essentials, Scaling Networks, Connecting Networks **CCNP R&S:** Switch, Route, TShoot
Seguridad	**Introduction to Cybersecurity**	**Cybersecurity Essentials**	**CCNA Cyber Ops** **CCNA Security**
IoT y análisis	**Introduction to IoT**	**IoT Fundamentals:** Connecting Things, Big Data & Analytics, IoT Security* Hackathon Playbook	
SO y TI	**NDG Linux Unhatched**	**NDG Linux Essentials** **IT Essentials**	**NDG Linux I** **NDG Linux II**
Programación		**CLA: Programming Essentials in C** **CPA: Programming Essentials in C++** **PCAP: Programming Essentials in Python** **Talleres de tecnología emergente:** Experimentando con API REST utilizando Cisco Spark*	**CLP: Advanced Programming in C*** **CPP: Advanced Programming in C++**
Laboral	**Be Your Own Boss**	**Entrepreneurship**	
Instrucción digital	**Get Connected**		

El portafolio educativo de Networking Academy

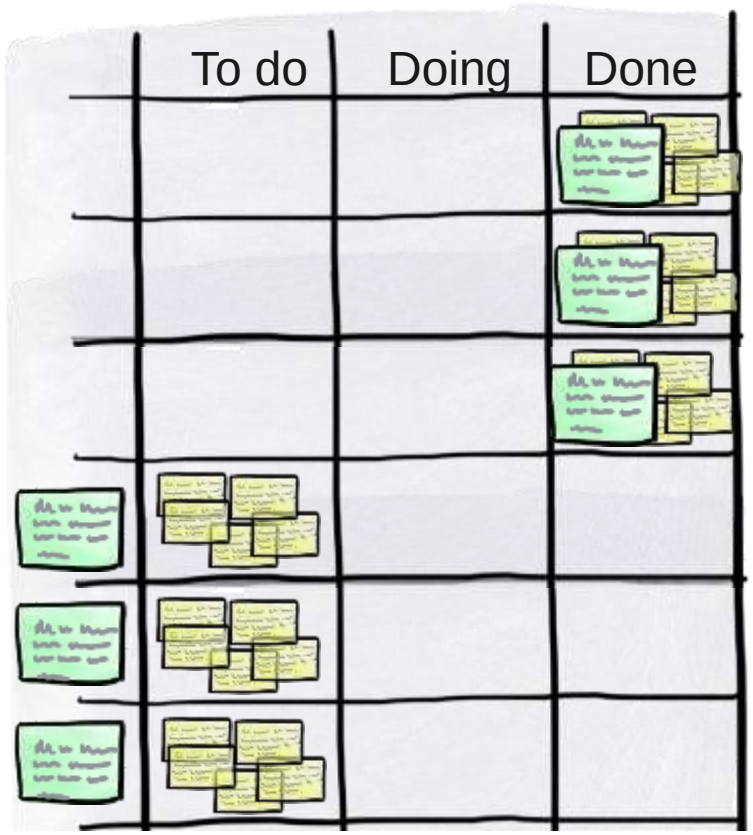
Colaborar para generar impacto

Introduction to Packet Tracer | **Packet Tracer** | **Hackatones** | **Prototyping Lab** | **Prácticas laborales**

	Exploratorio	Básico	Profesional
Redes		**Networking Essentials** **Mobility Fundamentals** **Talleres de tecnología emergente:** Programabilidad de redes con Cisco APIC-EM*	**CCNA R&S:** Introduction to Networks, R&S Essentials, Scaling Networks, Connecting Networks **CCNP R&S:** Switch, Route, TShoot
Seguridad	**Introduction to Cybersecurity**	**Cybersecurity Essentials**	**CCNA Cyber Ops** **CCNA Security**
IoT y análisis	**Introduction to IoT**	**IoT Fundamentals:** Connecting Things, Big Data & Analytics, IoT Security* Hackathon Playbook	
SO y TI	**NDG Linux Unhatched**	**NDG Linux Essentials** **IT Essentials**	**NDG Linux I** **NDG Linux II**
Programación		**CLA: Programming Essentials in C** **CPA: Programming Essentials in C++** **PCAP: Programming Essentials in Python** **Talleres de tecnología emergente:** Experimentando con API REST utilizando Cisco Spark*	**CLP: Advanced Programming in C*** **CPP: Advanced Programming in C++**
Laboral	**Be Your Own Boss**	**Entrepreneurship**	
Instrucción digital	**Get Connected**		



PCAP: Programming Essentials in Python



Agenda

1.- Introducción

2.- Python en la actualidad

3.- CNA: Portfolio educativo

4.- PCAP: Programming Essentials in Python

5.- PCAP: Demo del curso

6.- Hands-On Labs

PCAP: Programming Essentials in Python

Descripción general del curso

Diseñado para ser un curso fácil de entender y sencillo para principiantes, centrado en diversos tipos de recopilación de datos, herramientas de manipulación y operaciones de lógica y bit, y en la creación de API REST básicas.

Beneficios

Con PCAP: Programming Essentials in Python, aprenderá a diseñar, escribir, depurar y ejecutar programas codificados en el lenguaje Python. No se requiere ningún conocimiento previo de programación. El curso comienza con una orientación básica paso a paso hasta que se vuelva experto en resolver problemas más complejos.

Componentes educativos

- 5 módulos de contenido instructivo interactivo
- Más de 30 prácticas de laboratorio incluidas
- Herramienta en línea integrada para realizar prácticas de laboratorio
- Quizzes y Tests de evaluación por módulo y sección
- Test de evaluación final

Coordinado con la certificación

Características

Público objetivo: estudiantes de secundaria y terciarios

Requisitos previos: ninguno

Requiere capacitación a cargo de instructor: no

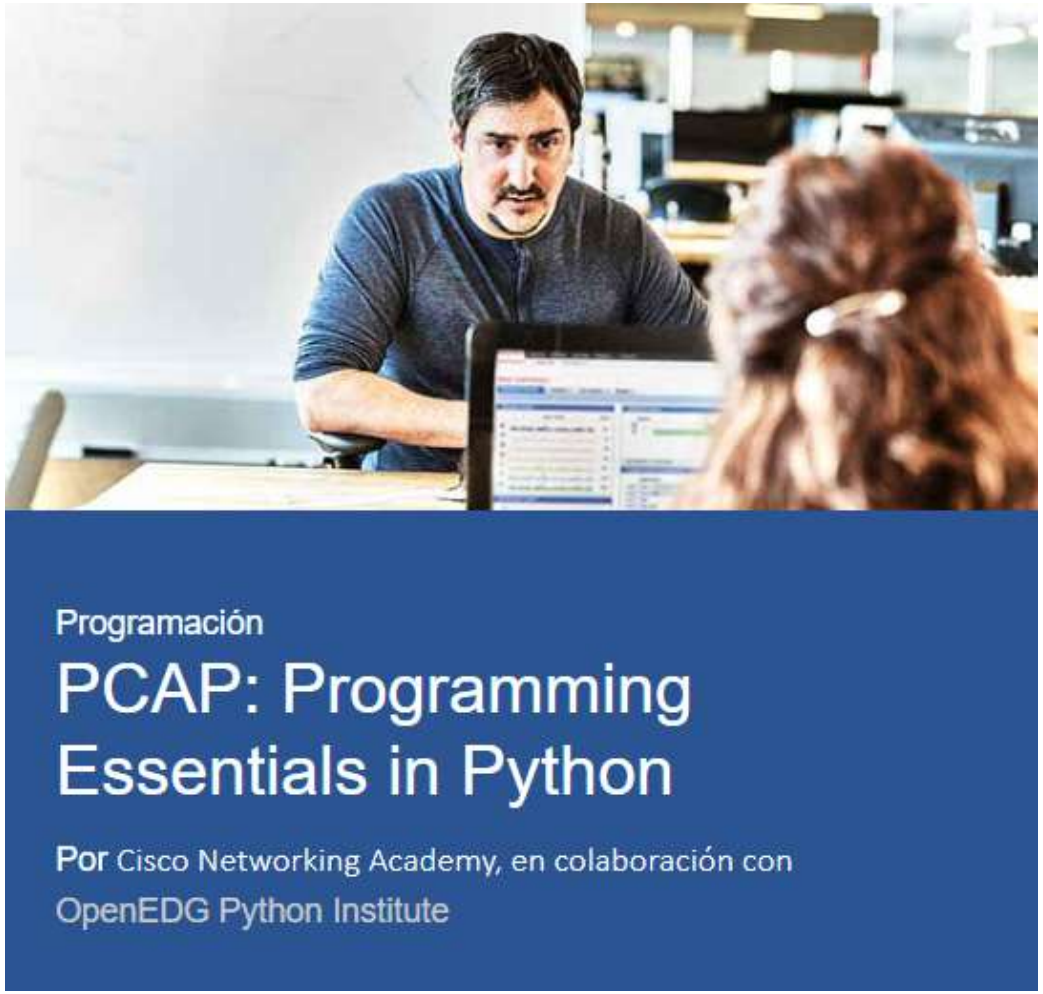
Idiomas: inglés

Presentación del curso: con instructor

Tiempo estimado para completar el curso: 60 a 70 horas

Próximo curso recomendado: IoT Fundamentals, Networking Essentials, NDG Linux Essentials

Programming Essentials in Python



Course Syllabus

El curso está formado por **5 módulos** con contenido interactivo, distribuidos en dos partes o secciones (*Basics* e *Intermediate*).

Introduction
Module 0

Part I: Basics
Module 1: Basics I
Module 2: Basics II
Module 3: Basics II

Part II: Intermediate
Module 4: Intermediate I
Module 5: Intermediate II

Programming Essentials in Python

Programación

PCAP: Programming Essentials in Python

Por Cisco Networking Academy, en colaboración con OpenEDG Python Institute

Edube Sandbox

Herramienta en línea integrada en el propio navegador web que ofrece un interprete de Python.

https://edube.org/sandbox?language=python

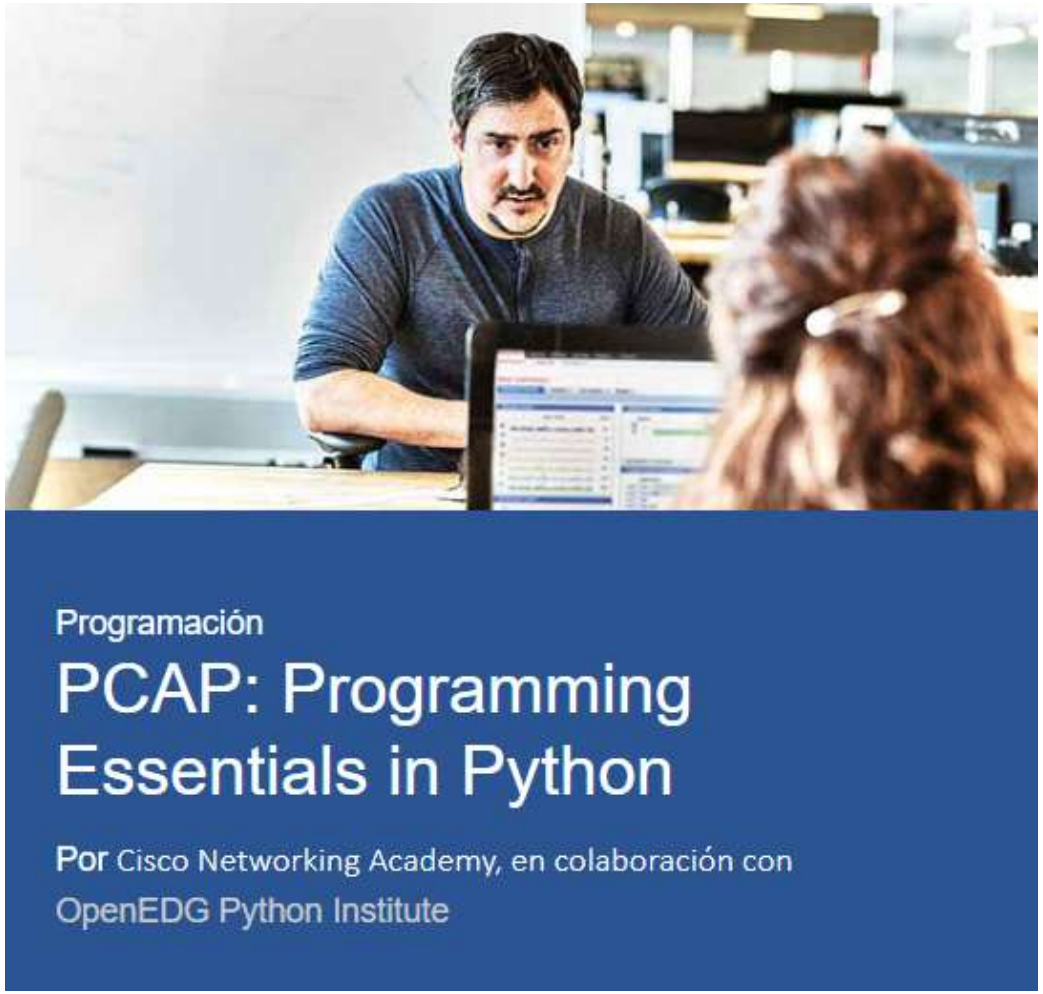
Sandbox

```python
import random as rnd

def max(op1, op2):
    return op1 if (op1 >= op2) else op2

def min(op1, op2):
    return op1 if (op1 <= op2) else op2


a = rnd.randint(1, 100)
b = rnd.randint(1, 100)
print("MAX({}, {}) = {}".format(a, b, max(a,b)))
print("MIN({}, {}) = {}".format(a, b, min(a,b)))
```

Console >_ Clear

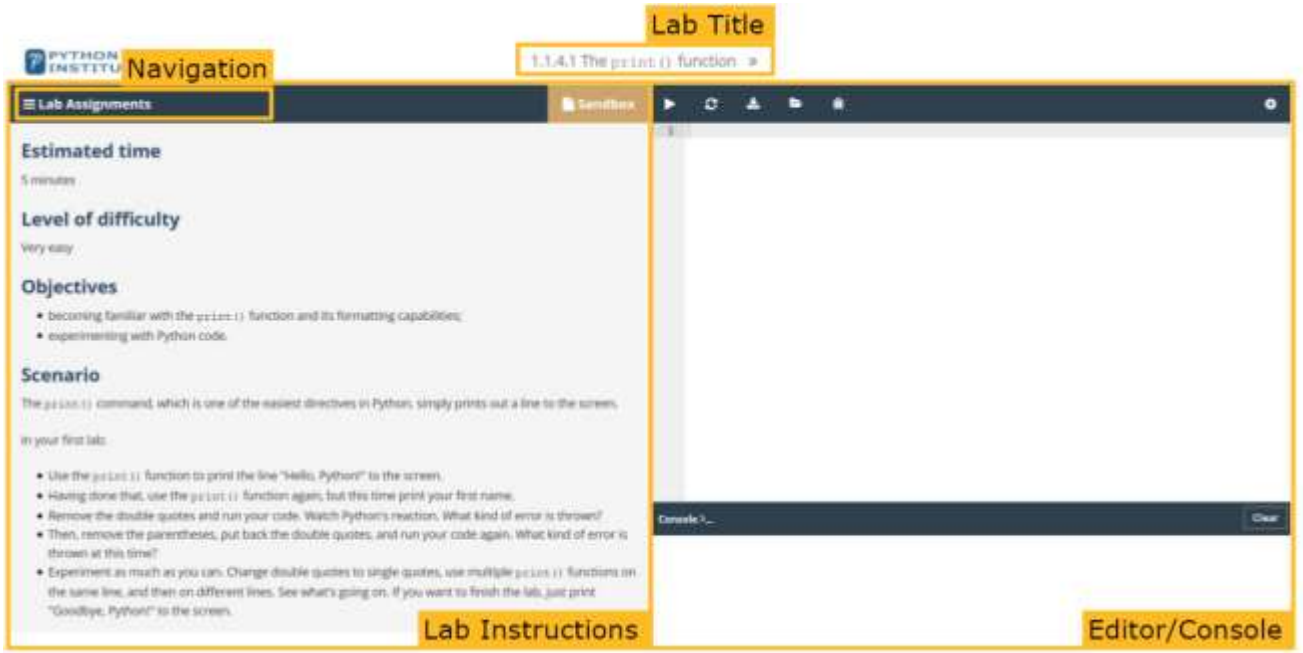
```
MAX(9, 43) = 43
MIN(9, 43) = 9
```

Programming Essentials in Python



Edube Labs

El curso contiene más de **30 prácticas o laboratorios** con soluciones incluidas.



Lab Assignments | Sandbox

Estimated time

30-45 minutes

Level of difficulty

Hard

Objectives

- improving the student's skills in operating with strings and lists;
- converting strings into lists.

Scenario

As you probably know, Sudoku is a number-placing puzzle played on a 9x9 board. The player has to fill the board in a very specific way:

- each row of the board must contain all digits from 0 to 9 (the order doesn't matter)
- each column of the board must contain all digits from 0 to 9 (again, the order doesn't matter)
- each of the nine 3x3 "tiles" (we will name them "sub-squares") of the table must contain all digits from 0 to 9.

If you need more details, you can find them here.

Your task is to write a program which:

- reads 9 rows of the Sudoku, each containing 9 digits (check carefully if the entered data are valid)
- outputs "Yes" if the Sudoku is valid, and "No" otherwise.

Test your code using the data we've provided.

Example input
295743861 431865927 876192543 387459216 612387495 549216738 763524189 928671354 154938672

Example output
Yes

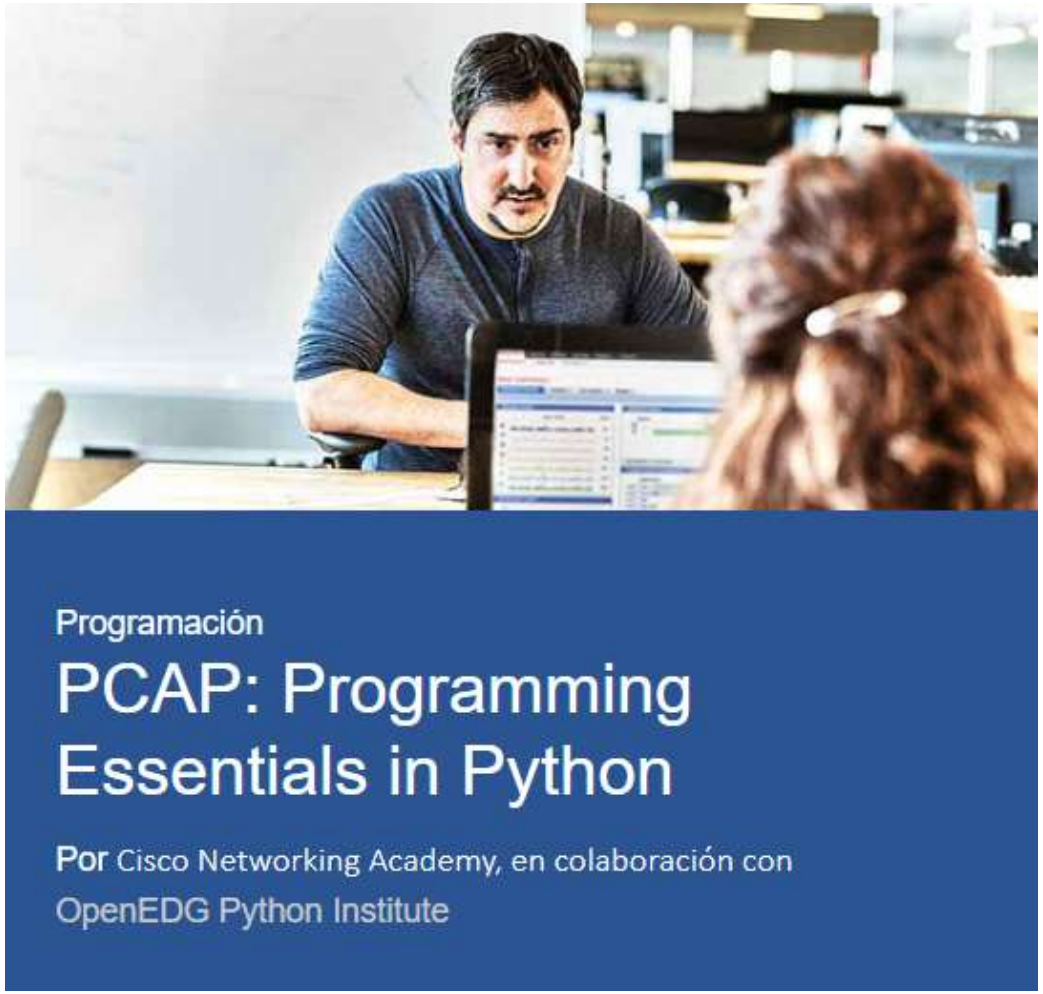
Example input
195743862 431865927 876192543 387459216 612387495 549216738 763524189 928671354 254938671

Example output
No

1

Console >_

Programming Essentials in Python

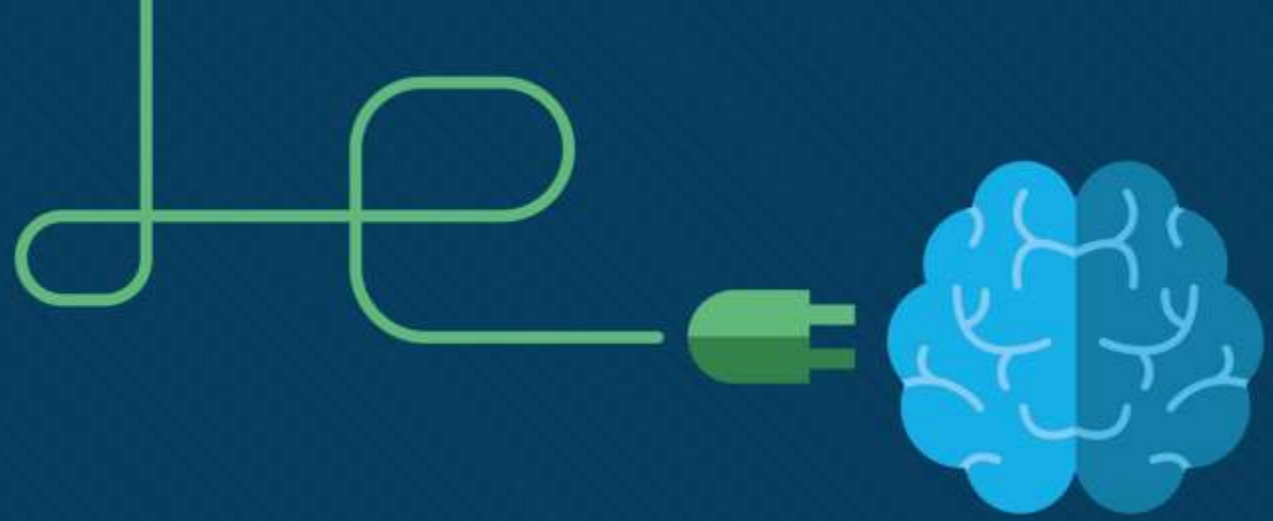


Modelo de evaluación

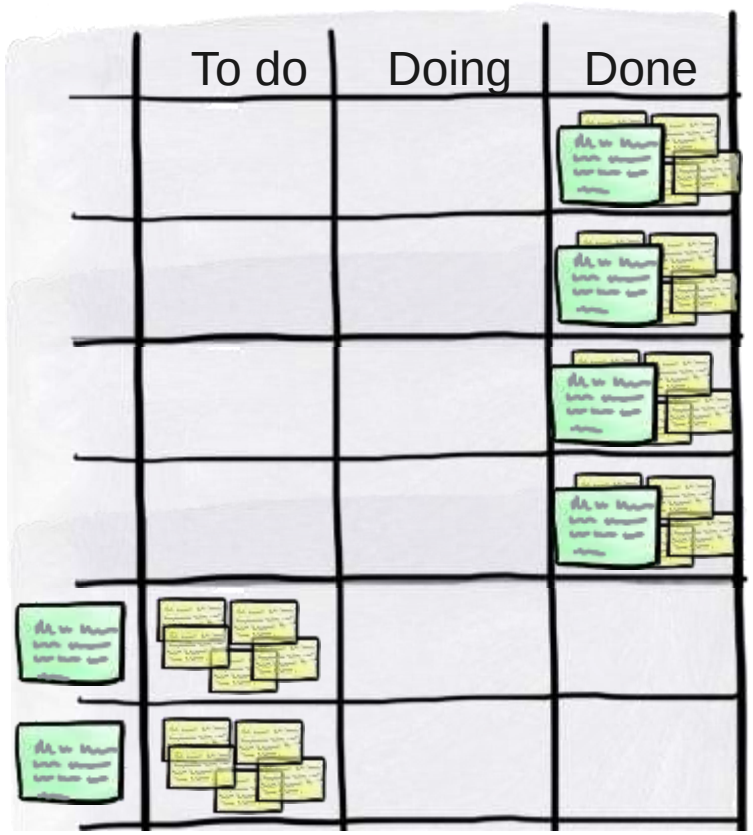
El curso contiene una serie de **quizzes** de preparación y **tests** de evaluación para cada módulo y sección.

Además existe un **test de evaluación final**.

Module 1 Test 20 ptos. | 20 preguntas	Module 5 Test 30 ptos. | 30 preguntas
Module 2 Test 20 ptos. | 20 preguntas	Summary Test 1 30 ptos. | 30 preguntas
Module 3 Test 20 ptos. | 20 preguntas	Summary Test 2 30 ptos. | 30 preguntas
Module 4 Test 20 ptos. | 20 preguntas	Final Test 50 ptos. | 50 preguntas



PCAP: Demo del curso



Agenda

1.- Introducción

2.- Python en la actualidad

3.- CNA: Portfolio educativo

4.- PCAP: Programming Essentials in Python

5.- PCAP: Demo del curso

6.- Hands-On Labs

Una oportunidad de por vida

Durante 20 años, Cisco Networking Academy les ha cambiado la vida a 9.2 millones de estudiantes en 180 países al ofrecerles educación, capacitación técnica y orientación en desarrollo profesional.

Meet Soso

Una oportunidad increíble lo está esperando. La tecnología está cambiando el mundo porque conecta miles de millones de dispositivos y mejora nuestra forma de vivir, trabajar, jugar y tratar a nuestro planeta. Ninguna industria es inmune. ¿Está listo para cambiar su vida y, posiblemente, hacer del mundo un lugar mejor?

Networking

Internet of Things

Programming

Security

OS & IT

Packet Tracer

Mi NetAcad
Cuenta
Tablero
Cursos
Calendario
Bandeja de entrada
Ayuda

Página de inicio
Módulos
Evaluaciones
Foros
Personas

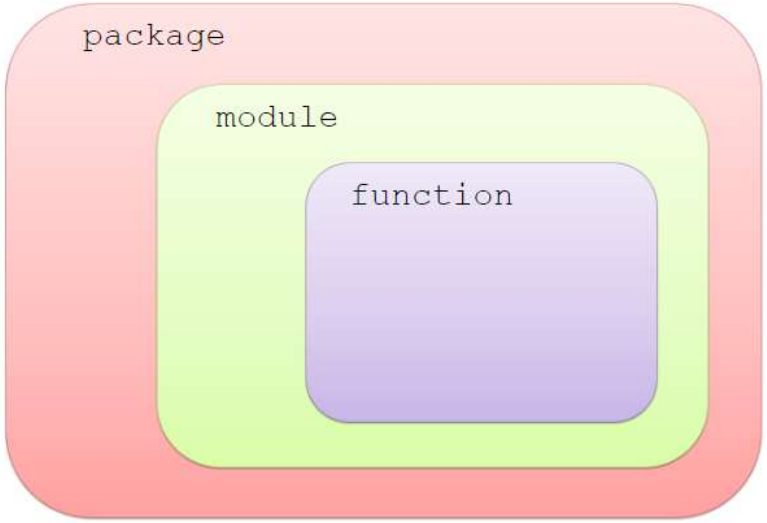
▸ Welcome to PCAP | Programming Essentials in Python		Complete todos los elementos ✓
▸ Edube | Sandbox and Lab Tool	Prerrequisitos: Welcome to PCAP | Programming Essentials in Python	
▸ Part 1: BASICS	Prerrequisitos: Welcome to PCAP | Programming Essentials in Python	
▸ Part 1 Summary Test	Prerrequisitos: Welcome to PCAP | Programming Essentials in Python	Complete todos los elementos
▸ Part 2: INTERMEDIATE	Prerrequisitos: Welcome to PCAP | Programming Essentials in Python, Part 1 Summary Test	🔒
▸ Part 2 Summary Test	Prerrequisitos: Welcome to PCAP | Programming Essentials in Python, Part 1 Summary Test	Complete todos los elementos 🔒
▸ Final Test	Prerrequisitos: Welcome to PCAP | Programming Essentials in Python, Part 2 Summary Test	Complete todos los elementos 🔒
▸ Course Completion Voucher	Prerrequisitos: Welcome to PCAP | Programming Essentials in Python, Final Test	🔒

4.3.1 Modules and packages

Writing your own modules doesn't differ much from writing ordinary scripts. There are some specific aspects you must be aware of, but it definitely isn't rocket science. You'll see this soon enough.

Let's summarize some important issues:

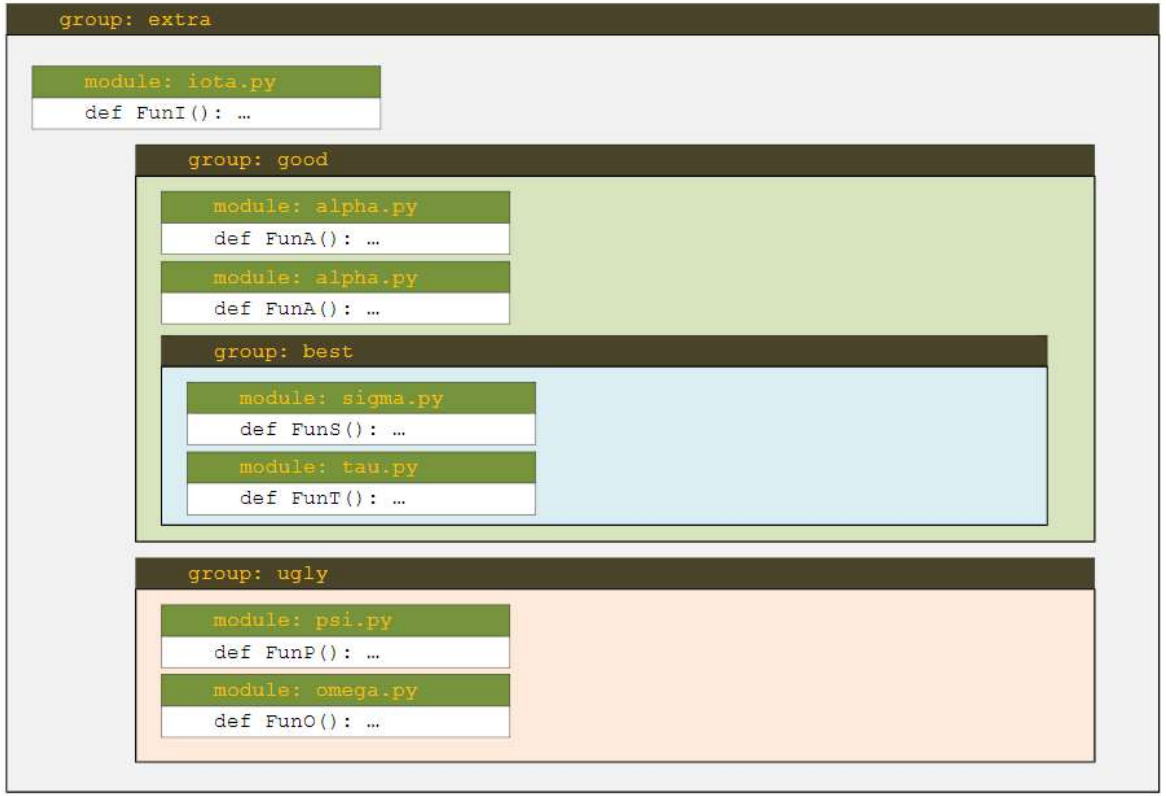
- a module is a **kind of container** filled with functions – you can pack as many functions as you want into one module and distribute it across the world;
- of course, it's generally a good idea **not to mix** functions with different application areas within one module (just like in a library – nobody expects scientific works to be put among comic books), so group your functions carefully and name the module containing them in a clear and intuitive way (e.g., don't give the name `arcade_games` to a module containing functions intended to partition and format hard disks)
- making many modules may cause a little mess – sooner or later you'll want to **group** your modules exactly in the same way as you've previously grouped functions – is there a more general container than a module?
- yes, there is – it's a **package**; in the world of modules, a package plays a similar role to a folder/directory in the world of files.



4.3.15 Your first package

Suddenly, somebody notices that these modules form their own hierarchy, so putting them all in a flat structure won't be a good idea.

After some discussion, the team comes to the conclusion that the modules have to be grouped. All participants agree that the following tree structure perfectly reflects the mutual relationships between the modules →



Let's review this from the bottom up:

- the `ugly` group contains two modules: `psi` and `omega`;
- the `best` group contains two modules: `sigma` and `tau`;
- the `good` group contains two modules (`alpha` and `beta`) and one subgroup (`best`)
- the `extra` group contains two subgroups (`good` and `bad`) and one module (`iota`)

Does it look bad? Not at all – analyze the structure carefully. It resembles something, doesn't it?

It looks like a directory structure.

5.5.6 Inheritance – why and how?

Read the code carefully →

There are two nested loops. Their purpose is to check all possible ordered pairs of classes, and to print the results of the check to determine whether the pair matches the subclass-superclass relationship.

The program produces the following output:

```
True  False False True  True  False True  True
True
```

Let's make the result more readable:

↓ is a subclass of →	Vehicle	LandVehicle	TrackedVehicle
Vehicle	**True**	**False**	**False**
LandVehicle	**True**	**True**	**False**
TrackedVehicle	**True**	**True**	**True**

There is one important observation to make: **each class is considered to be a subclass of itself**.

```
class Vehicle:
      pass

class LandVehicle(Vehicle):
      pass

class TrackedVehicle(LandVehicle):
      pass

for cl1 in [Vehicle, LandVehicle, TrackedVehicle]:
      for cl2 in [Vehicle, LandVehicle, TrackedVehicle]:
            print(issubclass(cl1,cl2),end='\t')
      print()
```

Download 5.5.6.py

1.1.23 The print() function

```
My_name_is*Monty*Python*
```

This is how it looks now →

Now that you understand the `print()` function, you're ready to consider how to store and process data in Python.

Without `print()`, you wouldn't be able to see any results.

Practice Labs

- Lab PCA_1_1_23_1_A *The `print()` function:*
  Launch in Edube | Launch Sample Solution
- Lab PCA_1_1_23_2_A *Formatting the output:*
  Launch in Edube | Launch Sample Solution

Estimated time

5-10 minutes

Level of difficulty

Easy

Objectives

- experimenting with existing Python code;
- discovering and fixing basic syntax errors;
- becoming familiar with the `print()` function and its formatting capabilities.

Scenario

We strongly encourage you to play with the code we've written for you, and make some (maybe even destructive) amendments. Feel free to modify any part of the code, but there is one condition - learn from your mistakes and draw your own conclusions.

Try to:

- minimize the number of `print()` function invocations by inserting the `\n` sequence into the strings
- make the arrow twice as large (but keep the proportions)
- duplicate the arrow, placing both arrows side by side; note: a string may be multiplied by using the following trick:
  `"string" * 2`
  will produce `"stringstring"` (we'll tell you more about it soon)
- remove any of the quotes, and look carefully at Python's response; pay attention to where

Sandbox

```
1 print("    *")
2 print("   * *")
3 print("  *   *")
4 print(" *     *")
5 print("***   ***")
6 print("  *   *")
7 print("  *   *")
8 print("  *****")
```

Console >_

Sandbox

SAMPLE SOLUTION: This is only an example of how you can do the lab. Sometimes, there may be multiple ways to do an exercise, and the solution here may not necessarily be the only, or even the best, way.

Estimated time

5-10 minutes

Level of difficulty

Easy

Objectives

- experimenting with existing Python code;
- discovering and fixing basic syntax errors;
- becoming familiar with the `print()` function and its formatting capabilities.

Scenario

We strongly encourage you to play with the code we've written for you, and make some (maybe even destructive) amendments. Feel free to modify any part of the code, but there is one condition - learn from your mistakes and draw your own conclusions.

Try to:

- minimize the number of `print()` function invocations by inserting the `\n` sequence into the strings
- make the arrow twice as large (but keep the proportions)

```
# Sample Solution

##################
print("original version:")
##################
print("    *")
print("   * *")
print("  *   *")
print(" *     *")
print("***   ***")
print("  *   *")
print("  *   *")
print("  *****")
##################
print("with fewer 'print()' invocations:")
##################
print("    *\n   * *\n  *   *\n *     *\n***   ***")
print("  *   *\n  *   *\n  *****")
##################
print("higher:")
##################
print("        *")
print("       * *")
print("      *   *")
print("     *     *")
print("    *       *")
print("   *         *")
print("  *           *")
print(" *             *")
print("******     ******")
print("     *     *")
print("     *     *")
```

Console >_

Lab 3.3.12.2 Writing and using your own functions - basics:

Open in the online simulator

Download a PDF

Lab 3.3.12.3 Writing and using your own functions - basics:

Open in the online simulator

Download a PDF

Lab 3.3.12.4 Prime numbers - how do we find them?

Open in the online simulator

Download a PDF

```
def strangelist(n):
   list = [ ]
   for i in range(0,n):
      list.insert(0,i)
   return list

print(strangelist(5))
```

Estimated time

10-20 minutes

Level of difficulty

Medium

Objectives

- improving the student's skills in operating with strings;
- using built-in Python string methods.

Scenario

You already know how the `split()` works. Now we want you to prove it.

Your task is to write your own function, which behaves almost exactly like the original `split()` method, i.e:

- it should accept exactly one argument – a string;
- it should return a list of words created from the string, divided in the places where the string contains white spaces;
- if the string is empty, the function should return an empty list;
- its name should be `mysplit()`

Use the template presented below. Test your code carefully.

Sandbox

```
def mysplit(strng):
#
# put your code here
#

print(mysplit("To be or not to be, that is the question"))
print(mysplit("To be or not to be,that is the question"))
print(mysplit("   "))
print(mysplit(" abc "))
print(mysplit(""))
```

Console >_

decimal digit using a subset of seven segments. If you still don't know what it is, refer to the following Wikipedia article.

Your task is to write a program which is able to simulate the work of a seven-display device, although you're going to use single LEDs instead of segments. Each digit is constructed from 13 LEDs (some lit, some dark, of course) - that's how we imagine it:

Note: the number 8 shows all the LED lights on.

Your code has to *display* any non-negative integer number entered by the user.

Hint: using a list containing patterns of all ten digits may be very helpful.

Example input
123

Example output
###

Example input
9081726354

Example output
#
#

Sandbox

```
1 # ### ### # # ### ### ### ### ### ###
2   #   #   # # # #   #     # # # # # # #
3   # ### ### ### ### ###   # ### ### # #
4   # #     #   #   # # #   # # #   # # #
5   # ### ###   # ### ###   # ### ### ###
```

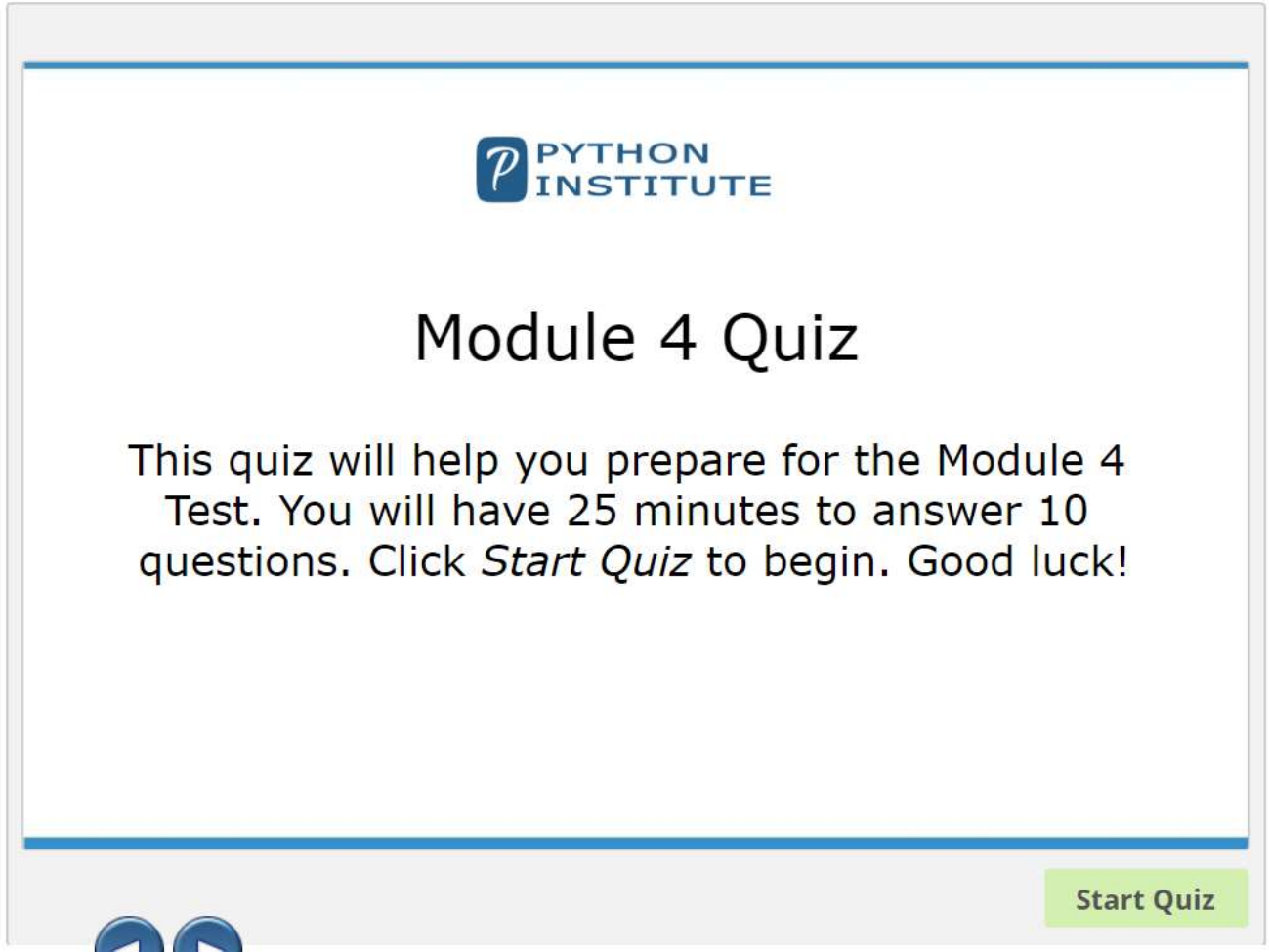
Console >_

Module 4 Quiz

Welcome to the Module 4 Quiz!

This quiz will help you prepare for the Module 4 Test. You will have 25 minutes to answer 10 questions, and you will be able to see the correct answers after you submit the quiz. You are free to take the quiz as many times as you like.

Please allow a few seconds for the quiz to load.



Module 4 Quiz

Welcome to the Module 4 Quiz!

This quiz will help you prepare for the Module 4 Test. You will have 25 minutes to answer 10 questions, and you will be able to see the correct answers after you submit the quiz. You are free to take the quiz as many times as you like.

Please allow a few seconds for the quiz to load.

Question 2 of 10 | Point Value: 1 | Total Points: 0 out of 10 | 24:41

If you want to import *pi* from *math*, you'd use:

- ○ import pi from math
- ○ from pi import math
- ◉ from math import pi

Submit

Module 4 Quiz

Welcome to the Module 4 Quiz!

This quiz will help you prepare for the Module 4 Test. You will have 25 minutes to answer 10 questions, and you will be able to see the correct answers after you submit the quiz. You are free to take the quiz as many times as you like.

Please allow a few seconds for the quiz to load.

Question 2 of 10 | Point Value: 1 | Total Points: 1 out of 10 | 23:47

If you want to import *pi* from *math*, you'd use:

- ○ import pi from math
- ○ from pi import math
- ◉ from math import pi

Correct

That's right! You answered correctly.

Continue

Buscar evaluación

Evaluaciones para tareas

- **Final Test**
  50 ptos. | 50 preguntas
- **Module 1 Test**
  20 ptos. | 20 preguntas
- **Module 2 Test**
  20 ptos. | 20 preguntas
- **Module 3 Test**
  20 ptos. | 20 preguntas
- **Module 4 Test**
  20 ptos. | 20 preguntas
- **Module 5 Test**
  30 ptos. | 30 preguntas
- **Summary Test 1**
  30 ptos. | 30 preguntas
- **Summary Test 2**
  30 ptos. | 30 preguntas

Mi NetAcad
Cuenta
Tablero
Cursos
Calendario
Bandeja de entrada
Ayuda

Página de inicio
Módulos
Evaluaciones
Foros
Personas

Module 3 Test

Fecha límite No hay fecha límite **Puntos** 20 **Preguntas** 20 **Tiempo límite** 30 minutos
Intentos permitidos 30

Instrucciones

Number of questions: **20**

Time limit: **30 minutes**

Allowed attempts: **3**

See Correct Answers: **Yes** (after the last attempt)

Points to score: **20** (each question is worth 1 point)

Passing score: **No**

Realizar la evaluación

◂ Anterior

Siguiente ▸

See Correct Answers: **Yes** (after the last attempt)

Points to score: **20** (each question is worth 1 point)

Passing score: **No**

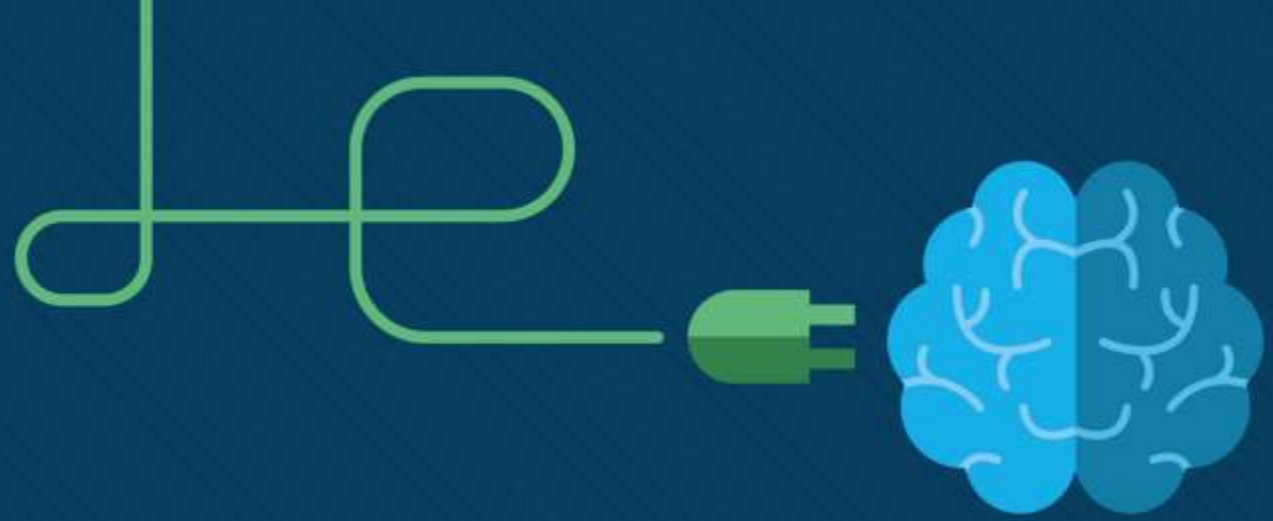
Pregunta 2 — 1 ptos.

A function defined in the following way:

```
def function(x=0):
    return x
```

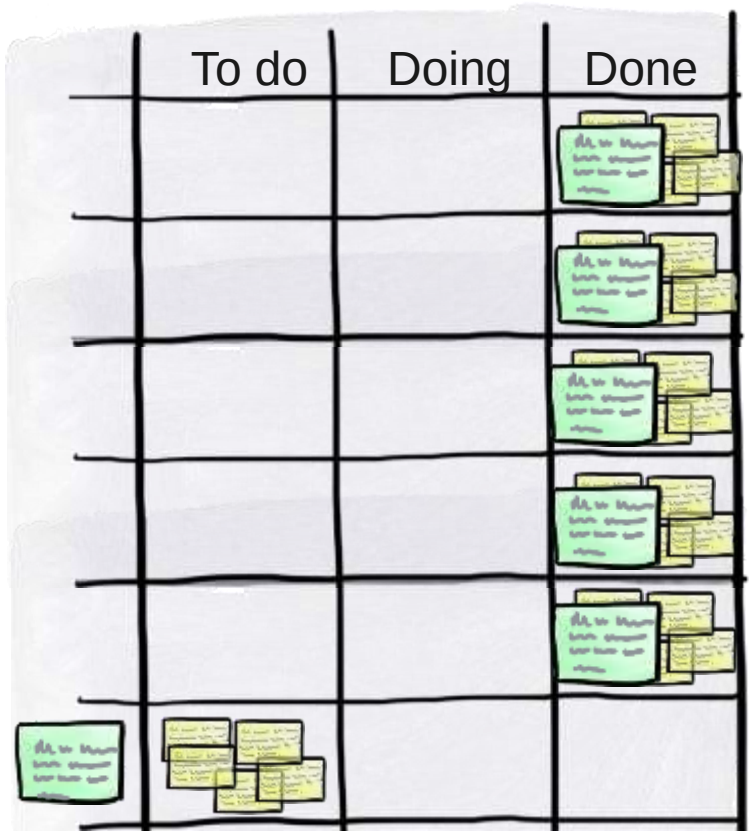
- may be invoked without any argument, or with just one
- may be invoked with any number of arguments (including zero)
- must be invoked with exactly one argument
- must be invoked without arguments

◂ Anterior Siguiente ▸



Hands-On Labs

Una introducción práctica al procesamiento de imágenes y la visión artificial con Python



Agenda

1.- Introducción

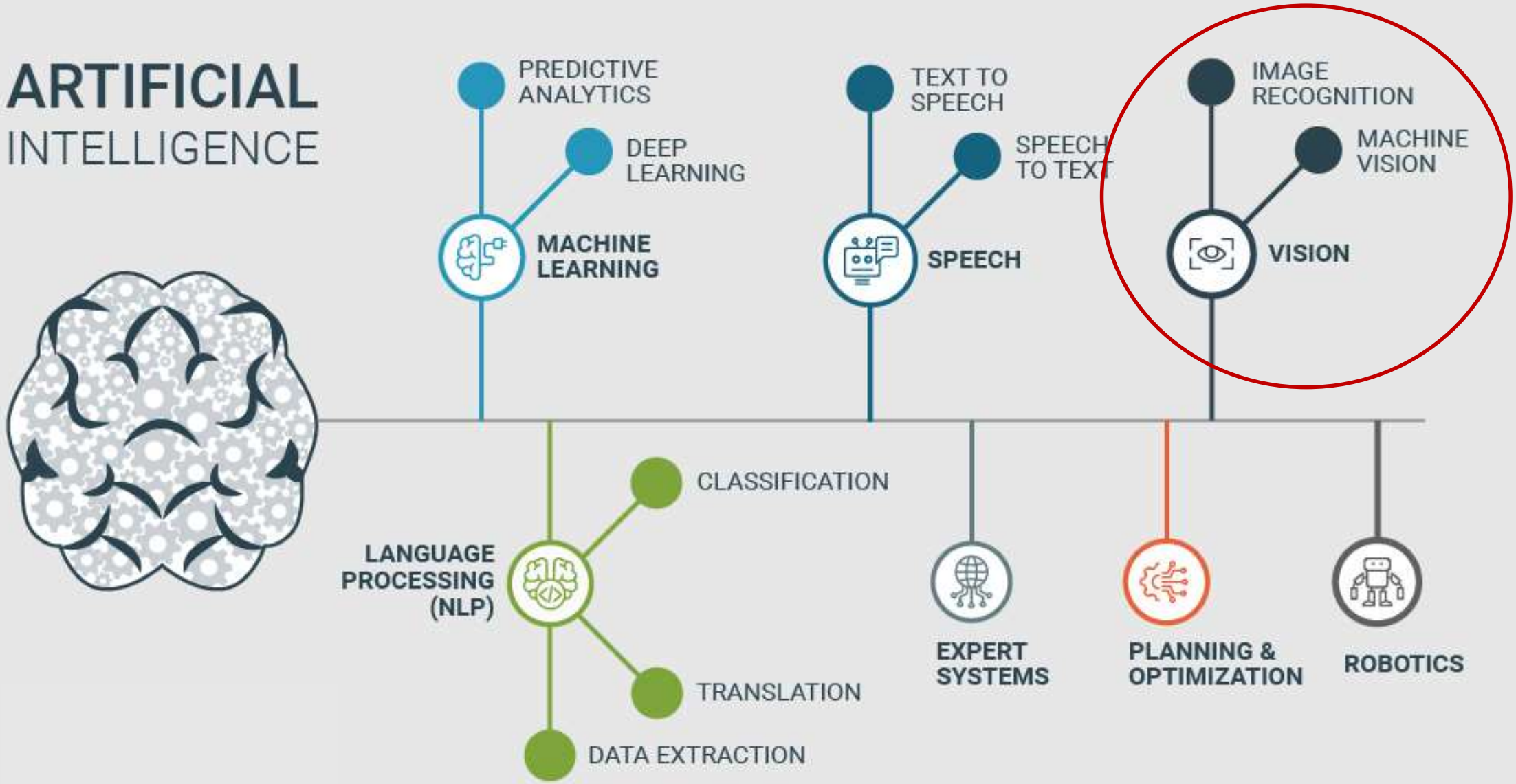
2.- Python en la actualidad

3.- CNA: Portfolio educativo

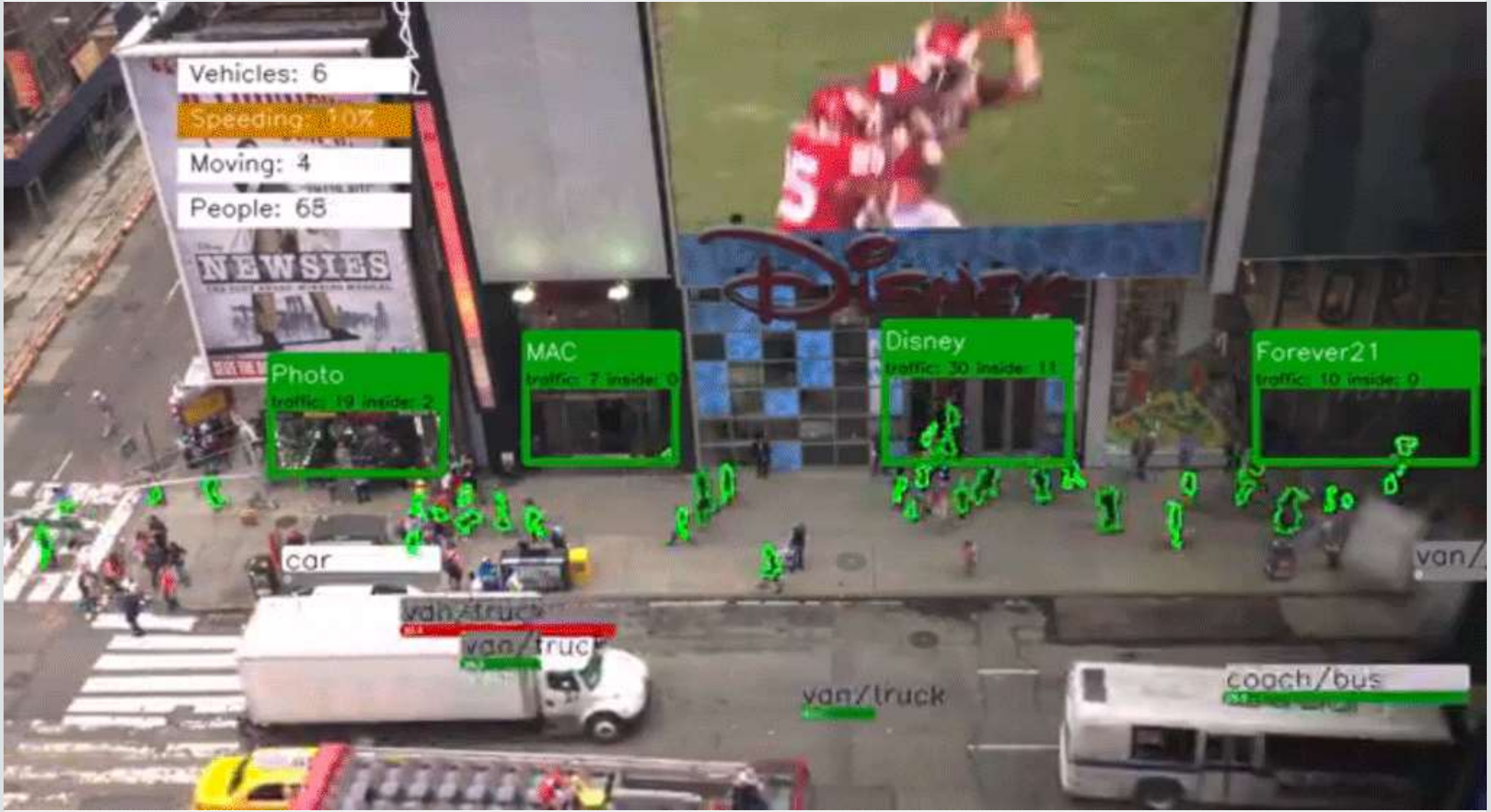
4.- PCAP: Programming Essentials in Python

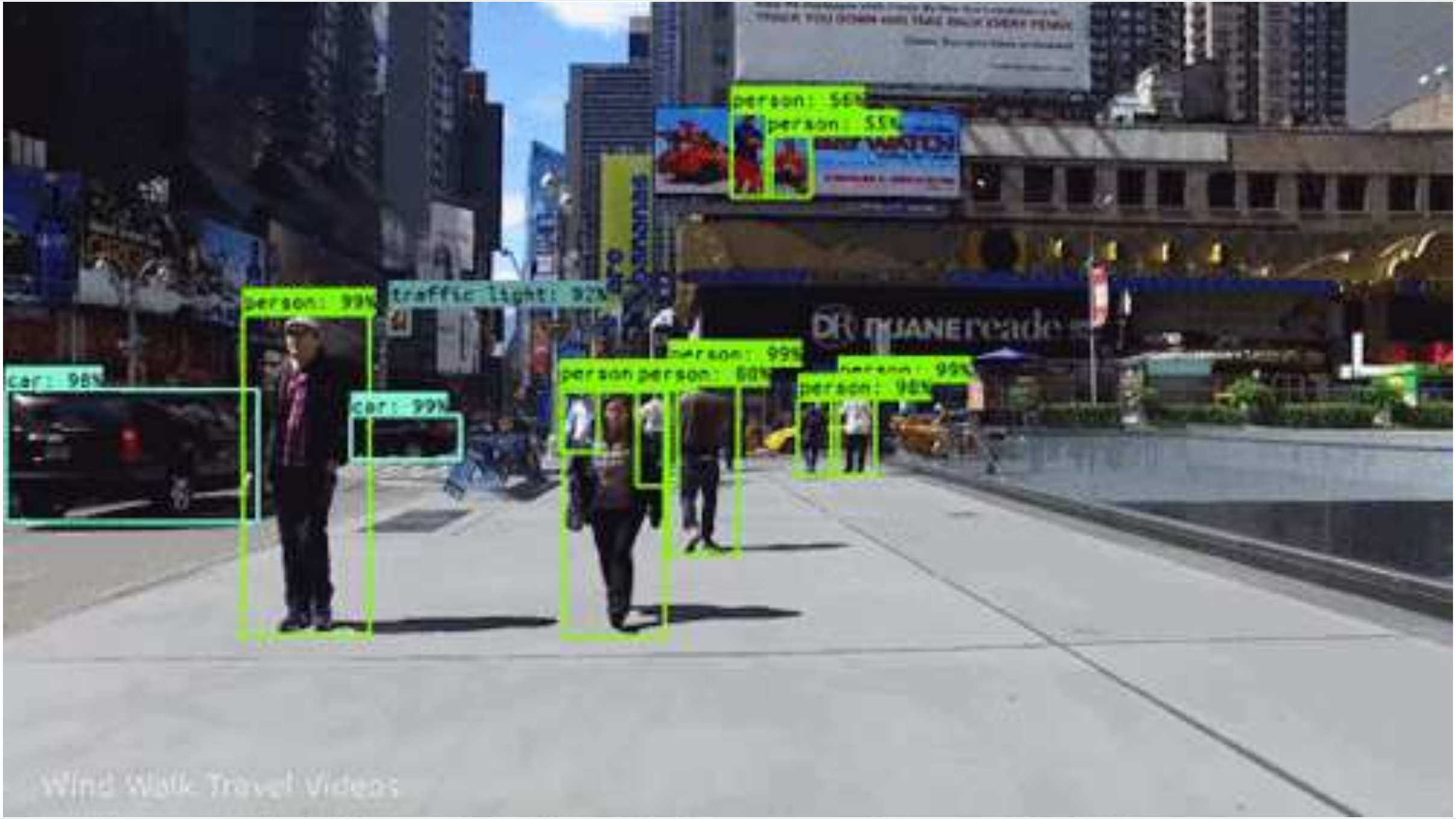
5.- PCAP: Demo del curso

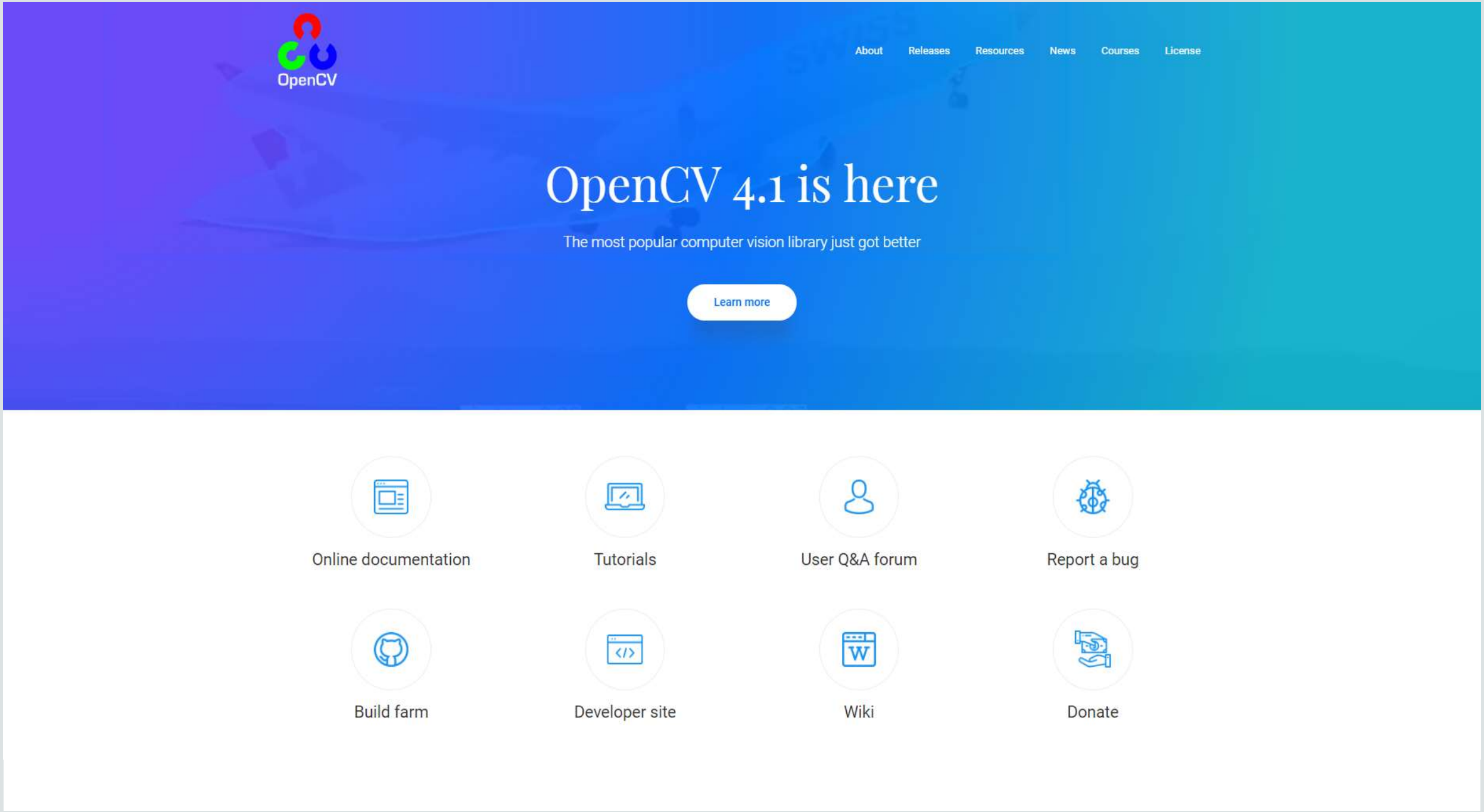
6.- Hands-On Labs



Conseguir que las máquinas imiten el sistema de visión humano
Resolver el problema de interpretar de forma automática el contenido de las imágenes y vídeos


Vehicles: 6
Speeding: 17%
Moving: 4
People: 65
NEWSIES
Photo
traffic: 19 inside: 2
MAC
traffic: 7 inside: 0
Disney
traffic: 30 inside: 11
Forever21
traffic: 10 inside: 0
car
van/truck
van/truck
van/truck
coach/bus


person: 56%
person: 55%
person: 99%
traffic light: 92%
car: 98%
car: 99%
person person: 88%
person: 99%
person: 98%
person: 99%
DR JANEreade


OpenCV
About
Releases
Resources
News
Courses
License
OpenCV 4.1 is here
The most popular computer vision library just got better
Learn more
Online documentation
Tutorials
User Q&A forum
Report a bug
Build farm
Developer site
Wiki
Donate

Hands-On Labs

OpenCV

python™

```
> pip install opencv-contrib-python

Collecting opencv-contrib-python
  Using cached
https://files.pythonhosted.org/packages/aa/4c/a3125e25da4b121b4077
c575eee8a691ffabb5d97afc5ae832fd5a7015e3/opencv_contrib_python-
4.1.0.25-cp37-cp37m-win32.whl
Collecting numpy>=1.14.5 (from opencv-contrib-python)
  Using cached
https://files.pythonhosted.org/packages/ab/75/9ac63977cbca68e17406
a53a8c573a925a16771800be47a73f18c838f3fb/numpy-1.16.3-cp37-cp37m-
win32.whl
Installing collected packages: numpy, opencv-contrib-python
Successfully installed numpy-1.16.3 opencv-contrib-python-4.1.0.25

> pip list
Package                Version
---------------------- --------
numpy                  1.16.3
opencv-contrib-python  4.1.0.25
pip                    19.1
setuptools             40.8.0
```

https://pypi.org/project/opencv-contrib-python/

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

load_image.py

#PUEAcademyDay19

```
# Import the necessary packages
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True,
                help = "path to where the image file resides")
ap.add_argument("-f", "--full", required = False, action='store_true’,
                help = "use full screen mode")
args = vars(ap.parse_args())

# Configure windows
windowName = "Image Visualizer"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)

# Loading the image using OpenCV
# The cv2.imread function returns a NumPy array representing the image
image = cv2.imread(args["image"])

# Show image info
print(f"Width: {image.shape[1]} pixels")
print(f"Height: {image.shape[0]} pixels")
print(f"Channels: {image.shape[2]}")
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

load_image.py


```
# Display the image to our screen
cv2.imshow(windowName, image)

# Note: OpenCV stores RGB channels in reverse order.
# While we normally think in terms of Red, Green and Blue;
# OpenCV stores them in the following order: Blue, Green and Red
# Get the pixel located at (0,0) represented as a tuple (b,g,r)
copy = image.copy()
(b, g, r) = copy[0, 0]
print(f"Pixel at (0,0) - RGB: ({r}, {g}, {b})")

# Modify the pixel located at (0,0)
copy[0, 0] = (0,0,255)
(b, g, r) = copy[0, 0]
print(f"Pixel at (0,0) - RGB: ({r}, {g}, {b})")

# Use NumPy array slicing capabilities to access larger rectangular
# portions of image
copy [0:100, 0:100] = (0,0,0)

# Write our image to disk
cv2.imwrite("images/demo.jpg", copy)

# Wait for a key press to finish program
cv2.waitKey(0)

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

load_video.py

#PUEAcademyDay19

```
# Import the necessary packages
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", required = True,
                help = "path to where the video resides")
ap.add_argument("-f", "--full", required = False, action='store_true’,
                help = "use full screen mode")
args = vars(ap.parse_args())

# Configure windows
windowName = "Video Visualizer"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)

# Loading the video using OpenCV
video = cv2.VideoCapture(args["video"])
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

load_video.py


```
# Display the video to our screen
# Start to lopping over all frames in the video.
# At most basic level, a video is simply a sequence of images together
while video.isOpened():
      (grabbed, frame) = video.read()

      # Display the output
      cv2.imshow(windowName, frame)

      # Wait for a key press to finish program
      key = cv2.waitKey(1)
      if (key in [ord("q"), 27] or
      cv2.getWindowProperty(windowName,cv2.WND_PROP_VISIBLE) < 1) :
            break

# The reference to the video is released
video.release()

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```


01. Usain Bolt
MisterFilOfficial
100m Men
WR 9.69 CR 9.80
6.1

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

load_cam.py

#PUEAcademyDay19

```
# Import the necessary packages
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--mirror", required = False,
                action='store_true', help = "enable mirror mode")
ap.add_argument("-f", "--full", required = False, action='store_true',
                help = "use full screen mode")
args = vars(ap.parse_args())

# Configure windows
windowName = "Webcam Visualizer"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)

# Loading the webcam image using OpenCV
cam = cv2.VideoCapture(0)
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

load_cam.py

#PUEAcademyDay19

```
# Display the webcam to our screen
# Start to lopping over all frames in the webcam.
# At most basic level, a video is simply a sequence of images together
while True:
	(grabbed, frame) = cam.read()

	# Mirror effect: Flip on Y axis
	if (args["mirror"]):
		frame = cv2.flip(frame,1)

	# Display the output
	cv2.imshow(windowName, frame)

	# Wait for a key press to finish program
	key = cv2.waitKey(1)
	if (key in [ord("q"), 27] or
	cv2.getWindowProperty(windowName,cv2.WND_PROP_VISIBLE) < 1) :
		break

# The reference to the video is released
cam.release()

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

cam_capture.py

#PUEAcademyDay19

```
# Import the necessary packages
import time
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--mirror", required = False,
                action='store_true', help = "enable mirror mode")
ap.add_argument("-f", "--full", required = False, action='store_true',
                help = "use full screen mode")
args = vars(ap.parse_args())

# Configure windows
windowName = "Webcam Visualizer"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)

# Loading the webcam image using OpenCV
cam = cv2.VideoCapture(0)
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

cam_capture.py

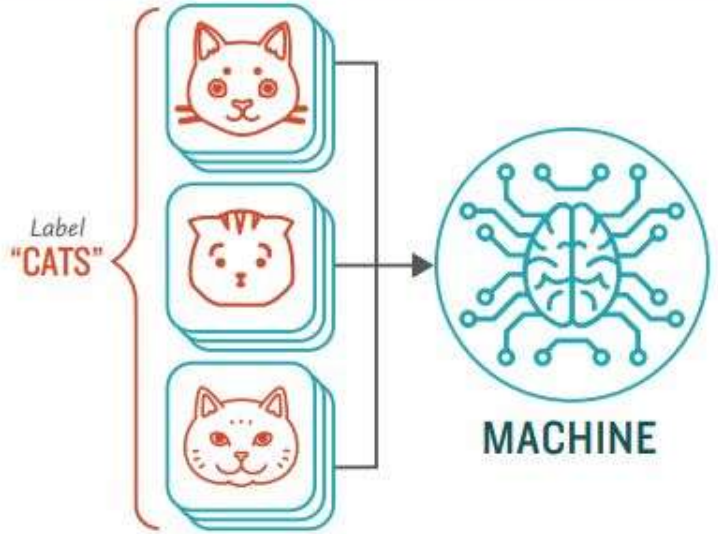
#PUEAcademyDay19

```
# Display the webcam to our screen
# Start to lopping over all frames in the webcam.
# At most basic level, a video is simply a sequence of images together
while True:
	(grabbed, frame) = cam.read()

	# Mirror effect: Flip on Y axis
	if (args["mirror"]):
		frame = cv2.flip(frame,1)

	# Display the output
	cv2.imshow(windowName, frame)

	# Wait for a key press to capture a frame or finish the program
	key = cv2.waitKey(1)
	if (key == ord("p")) :
		filename = time.strftime("%Y-%m-%d-%H-%M-%S")
		cv2.imwrite(f"captures/{filename}.png", frame)
	elif (key in [ord("q"), 27] or
	cv2.getWindowProperty(windowName,cv2.WND_PROP_VISIBLE) < 1) :
		break

# The reference to the video is released
cam.release()

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```

How **Supervised** Machine Learning Works

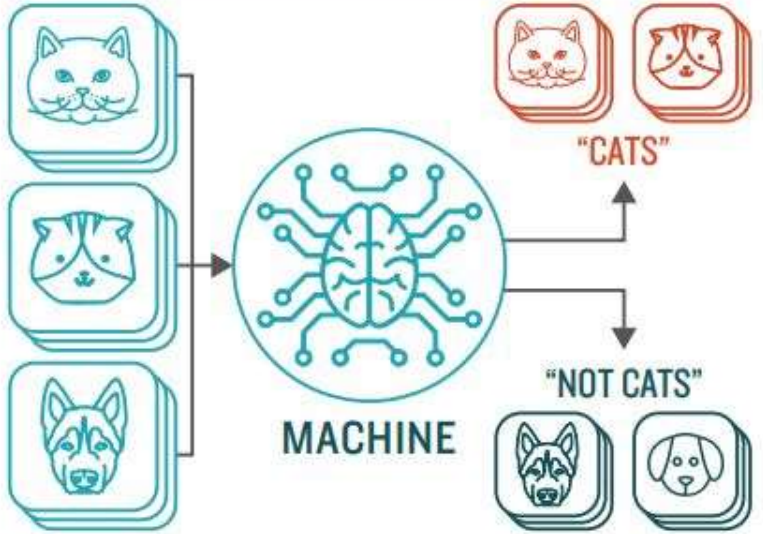
STEP 1

Provide the machine learning algorithm categorized or "labeled" input and output data from to learn

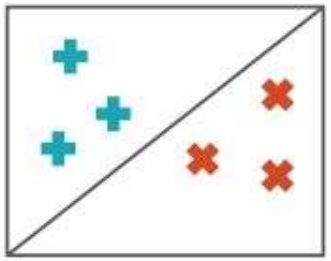


STEP 2

Feed the machine new, unlabeled information to see if it tags new data appropriately. If not, continue refining the algorithm

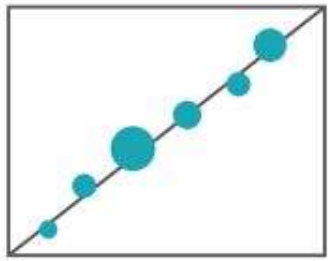


TYPES OF PROBLEMS TO WHICH IT'S SUITED



CLASSIFICATION

Sorting items into categories



REGRESSION

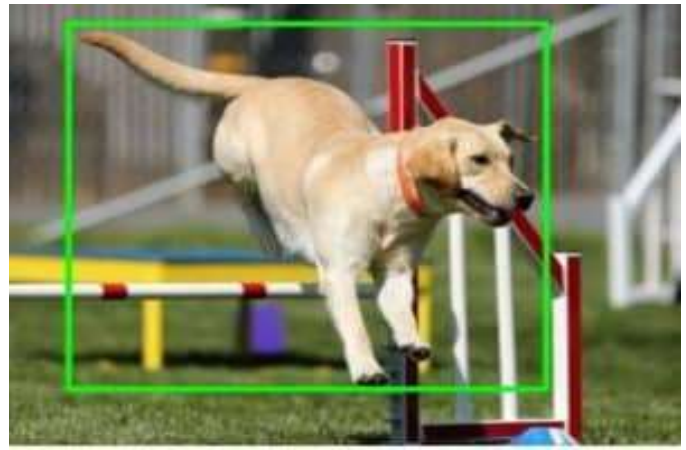
Identifying real values (dollars, weight, etc.)

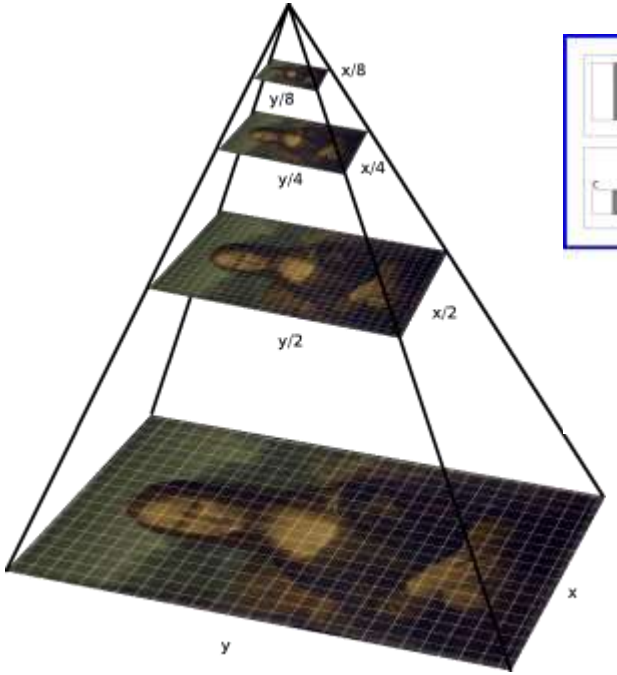
Hands-On Labs

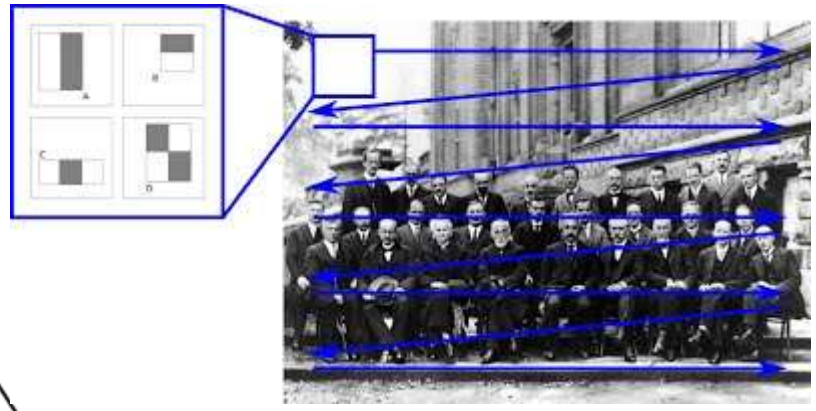
OpenCV

python™

Detección de objetos







Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

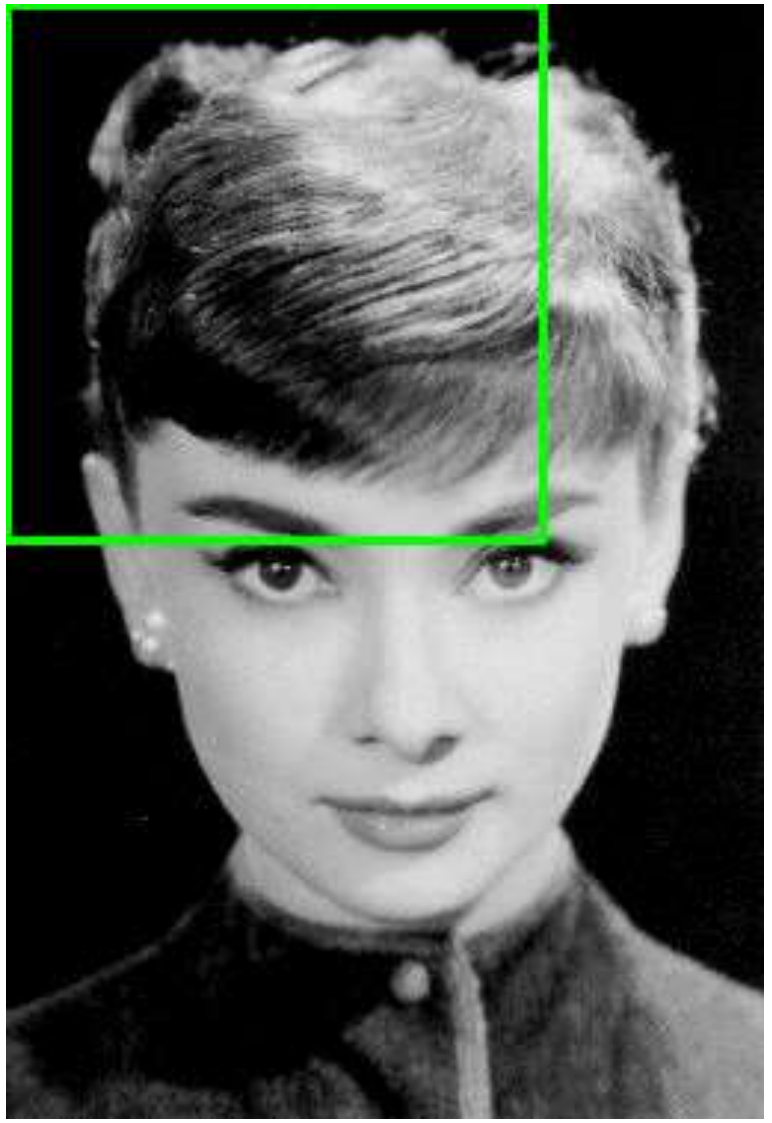
Sliding Window + Image Pyramid

Hands-On Labs

OpenCV

python™

Sliding Window + Image Pyramid



Hands-On Labs

OpenCV

python™

Sliding Window + Image Pyramid

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

img_faces_detection.py


```
# Import the necessary packages
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-fc", "--fclassifier", required = False,
                default="classifiers/haarcascade_frontalface.xml",
                help = "path to where the face cascade resides")
ap.add_argument("-i", "--image", required = True,
                help = "path to where the image file resides")
ap.add_argument("-f", "--full", required = False, action='store_true',
                help = "use full screen mode")
args = vars(ap.parse_args())

# Configure windows
windowName = "Faces Detector"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)

# Loading the image using OpenCV
image = cv2.imread(args["image"])

# Convert image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Hands-On Labs

OpenCV

python™

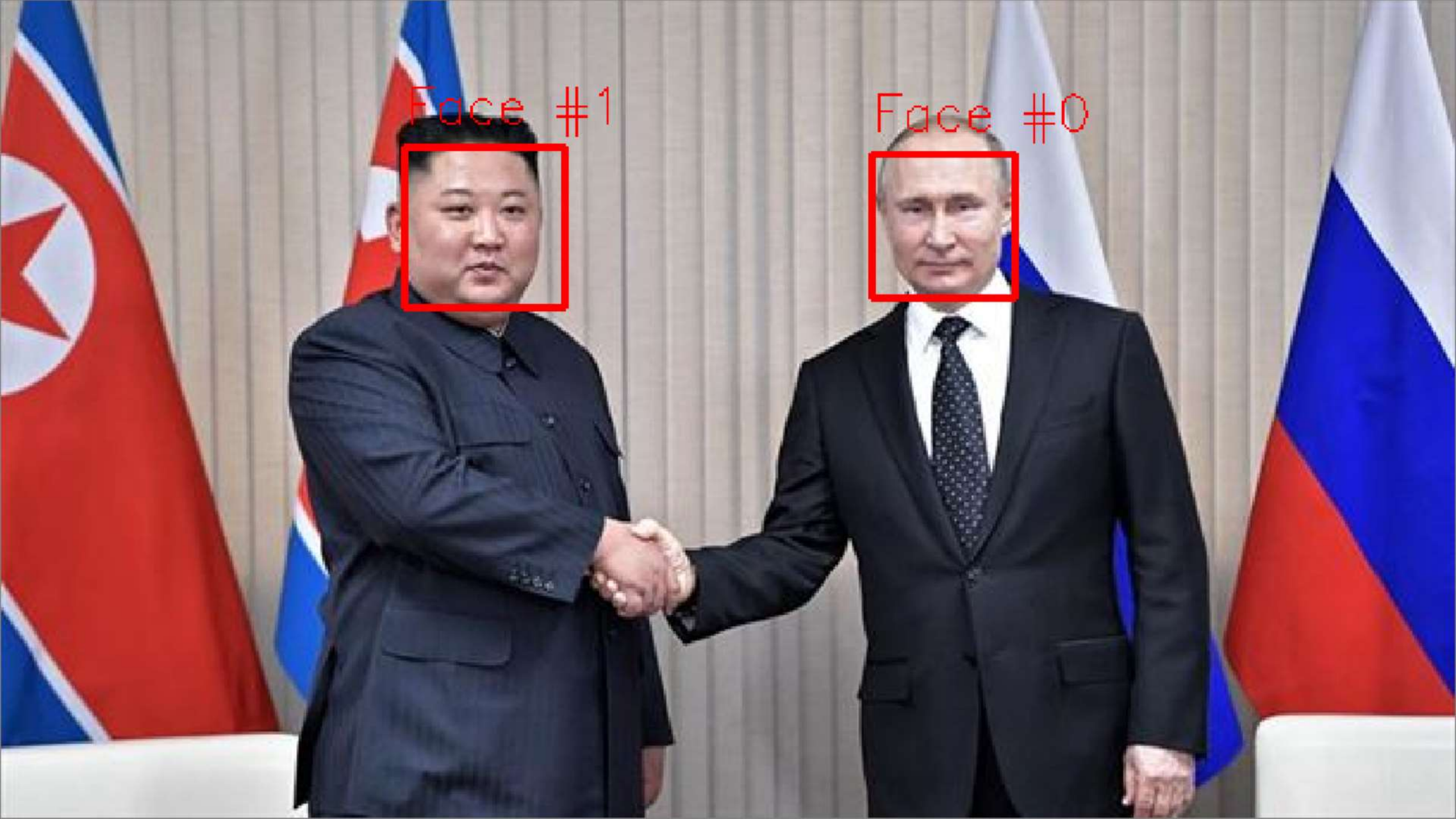
PUE ACADEMY Day

img_faces_detection.py

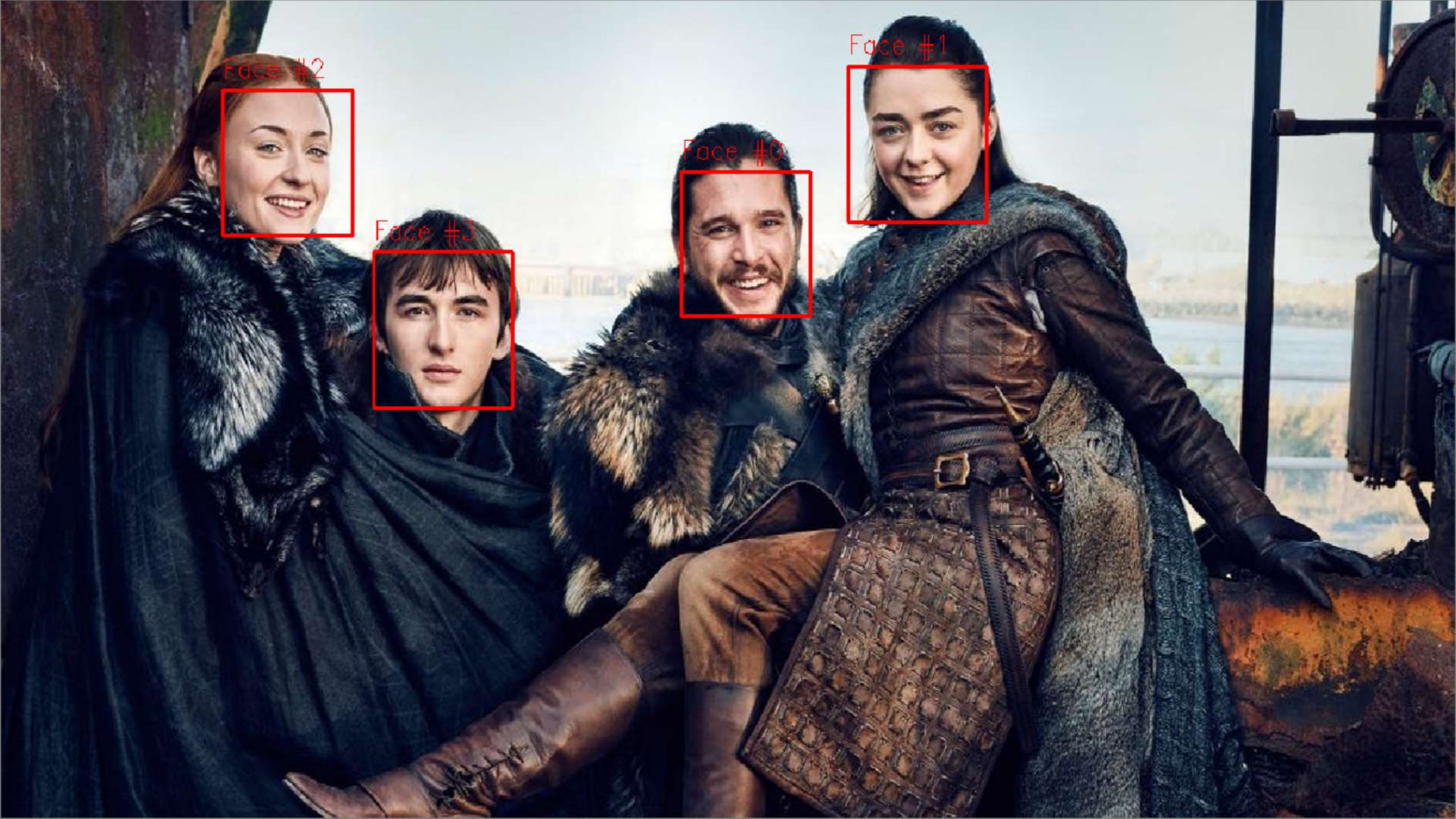
#PUEAcademyDay19

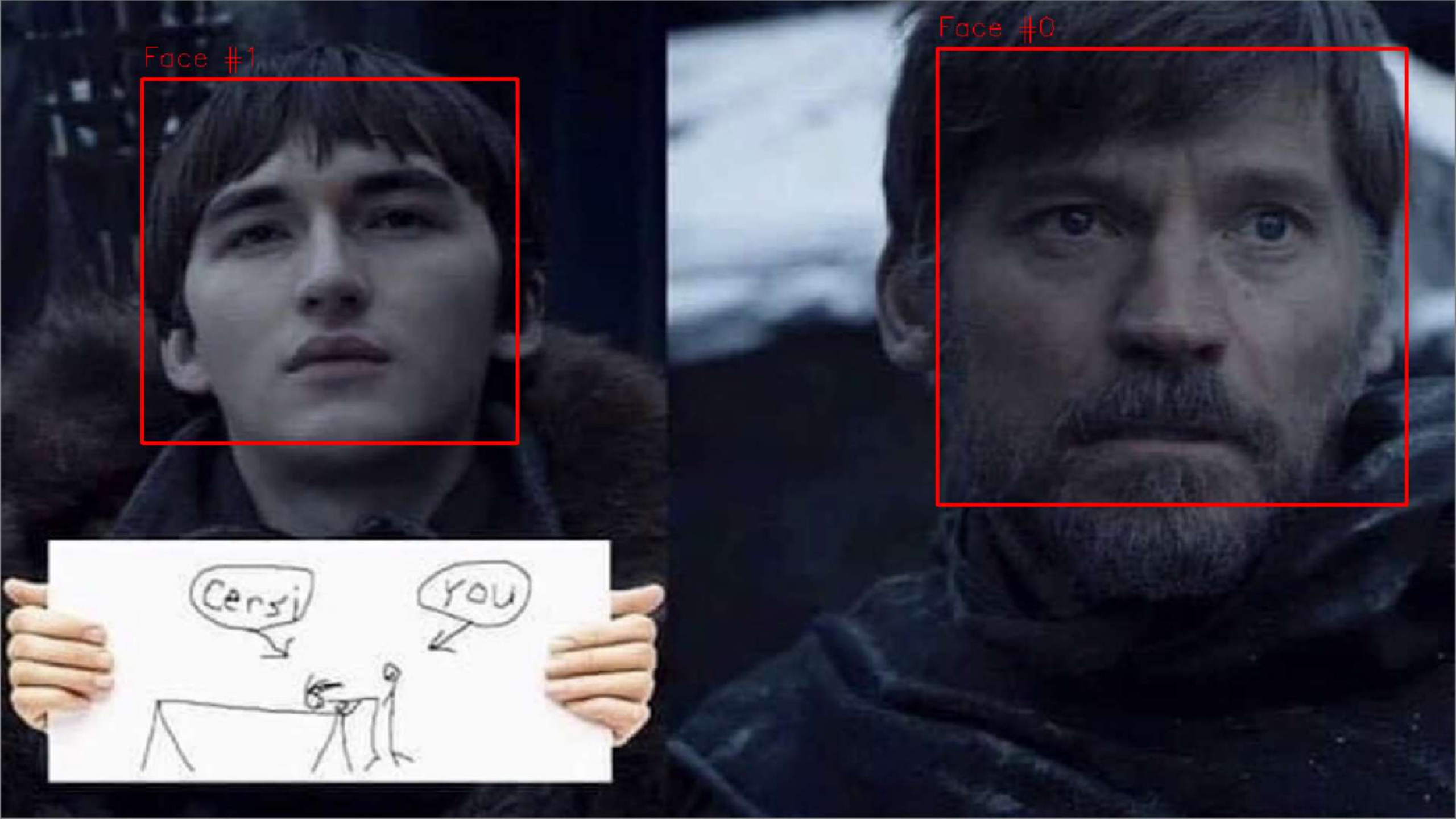
```
# In order to build face recognition system, we use the built-in
# Haar cascade classifiers in OpenCV.
# https://github.com/opencv/opencvdata/haarcascades
# This folder contains trained classifiers for detecting objects of a
# particular type,  e.g. faces (frontal, profile), pedestrians etc
facesClassifier = cv2.CascadeClassifier(args["fclassifier"])

# “sliding window” approach
# These classifiers work by scanning an image from left to right, and
# top to bottom, at varying scale sizes.
faces = facesClassifier.detectMultiScale(gray, scaleFactor = 1.3,
                                         minNeighbors = 5, minSize = (30,30))

print(f"I found {len(faces)} face(s)")

# loop over the faces and draw a rectangle surrounding each
for (i, (x, y, w, h)) in enumerate(faces):
    cv2.rectangle(image, pt1 = (x, y), pt2 = (x + w, y + h), …)
    cv2.putText(image, text = f"Face #{i}", org = (x, y - 10), …)

# Display the image to our screen
cv2.imshow(windowName, image)

# Wait for a key press to finish program
cv2.waitKey(0)

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```


Face #3
Face #4
Face #1
Face #5
Face #0
Face #2


Face #1
Face #0


Face #1
Face #0


Face #2
Face #3
Face #0
Face #1


Face #1
Face #0
Cersi
You


Face #8
Face #10
Face #11
Face #4
Face #6
Face #5
Face #0
Face #7
Face #9
Face #3
Face #1
Face #2

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

img_cats_detection.py


```
# Import the necessary packages
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-cc", "--cclassifier", required = False,
                default="classifiers\\haarcascade_frontalcatface.xml",
                help = "path to where the cat cascade resides")
ap.add_argument("-i", "--image", required = True,
                help = "path to where the image file resides")
ap.add_argument("-f", "--full", required = False, action='store_true',
                help = "use full screen mode")
args = vars(ap.parse_args())

# Configure windows
windowName = "Cats Detector"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)

# Loading the image using OpenCV
image = cv2.imread(args["image"])

# Convert image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Hands-On Labs

OpenCV

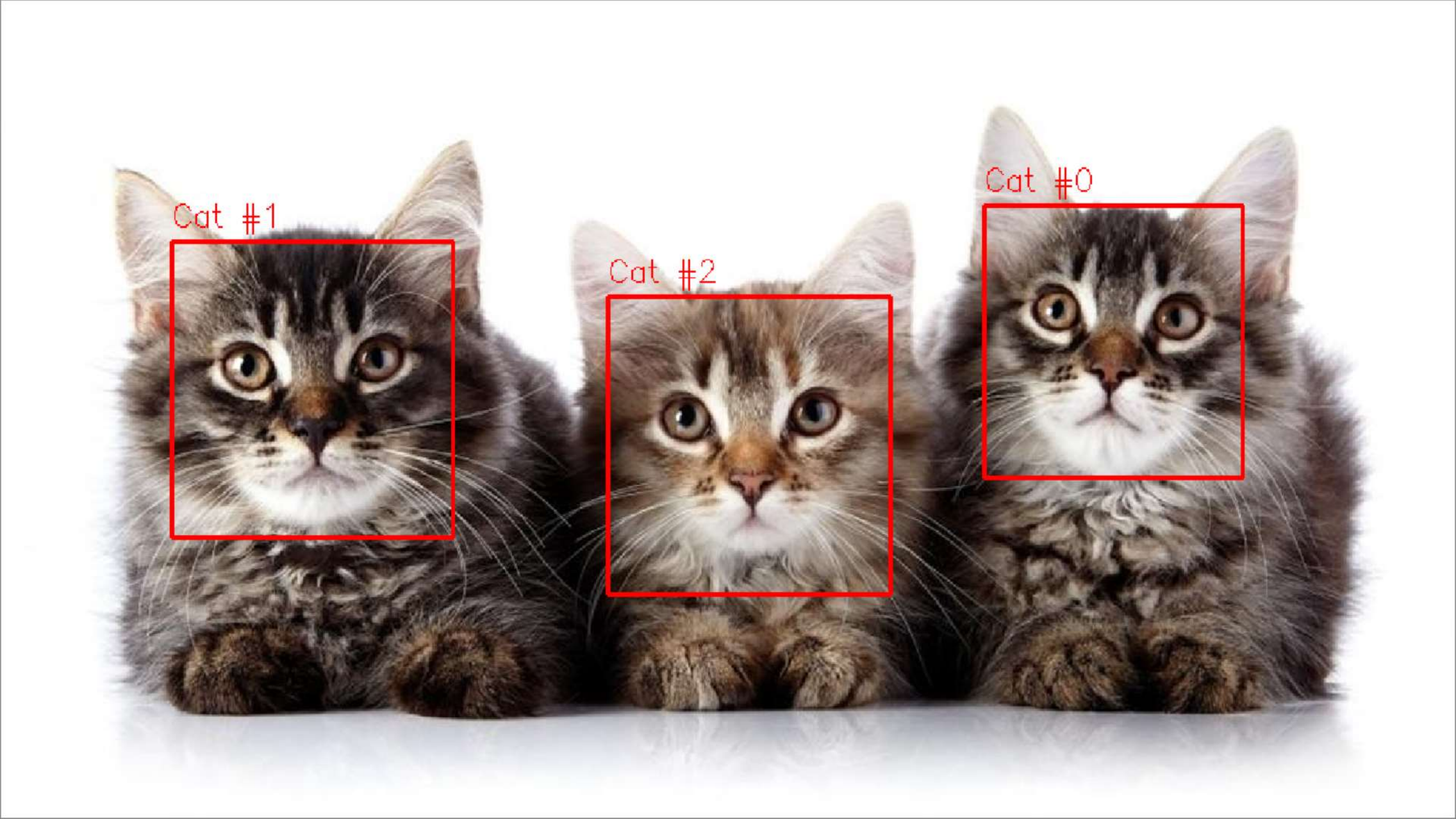
python™

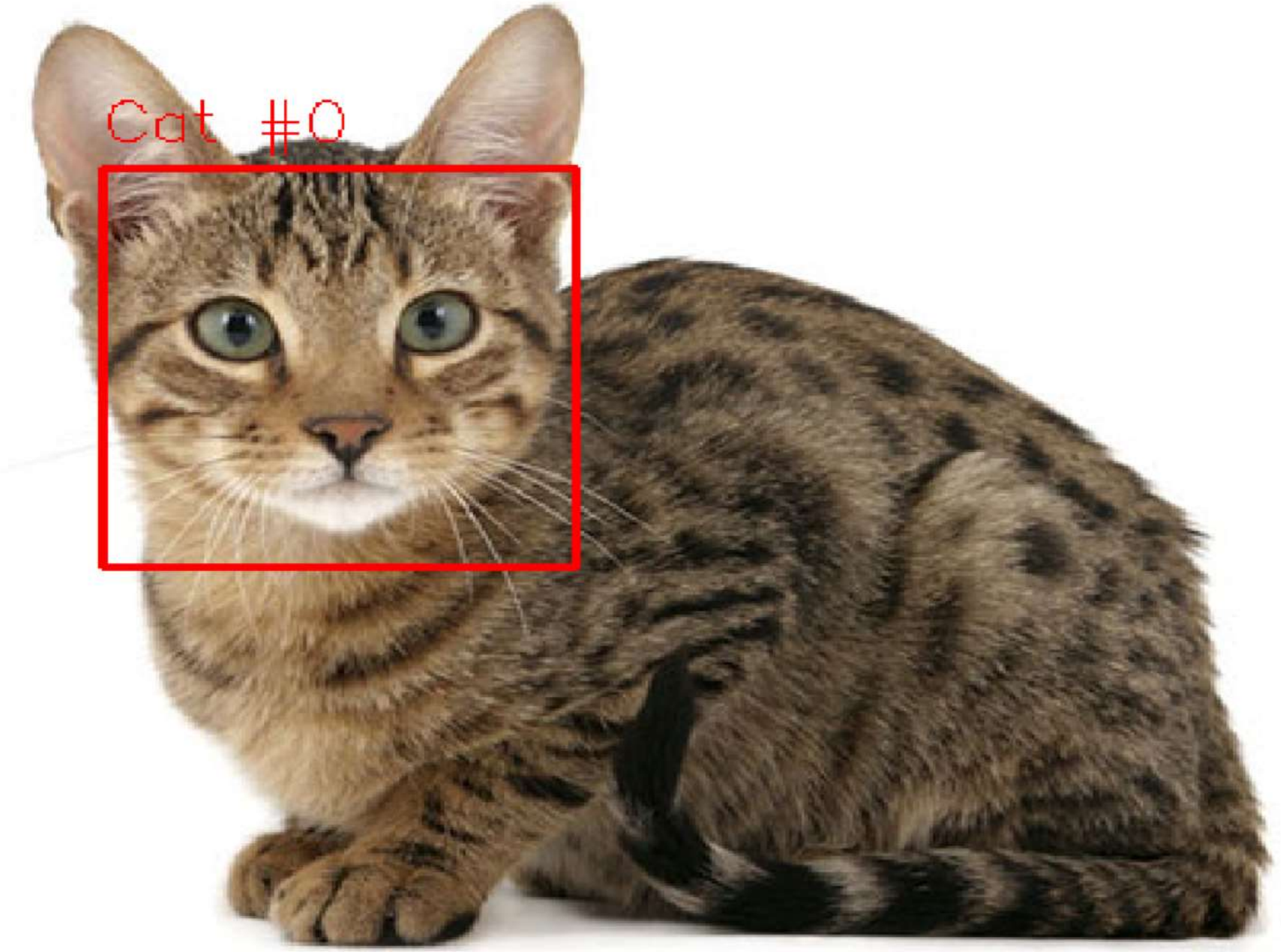
PUE ACADEMY Day

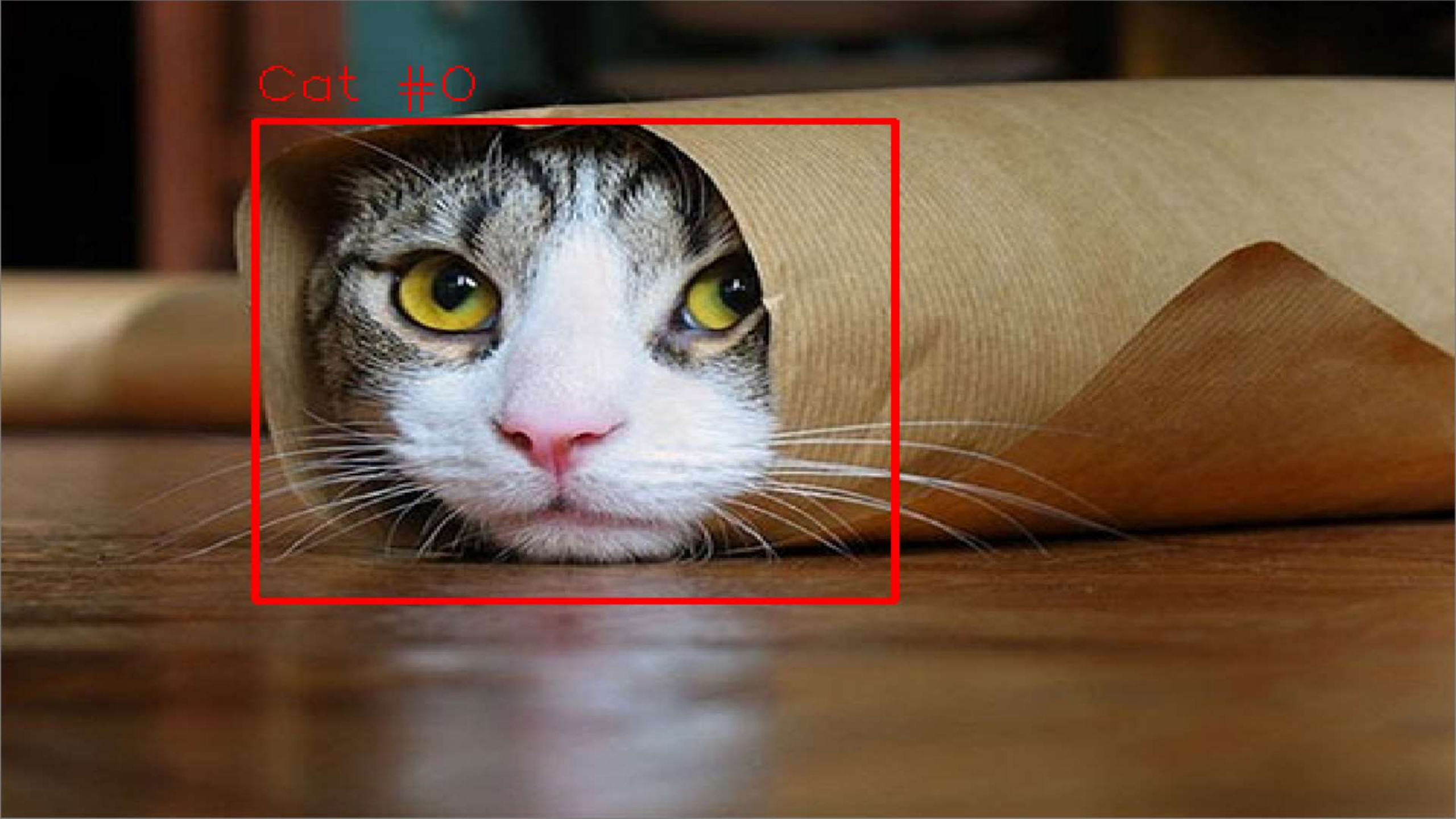
img_cats_detection.py

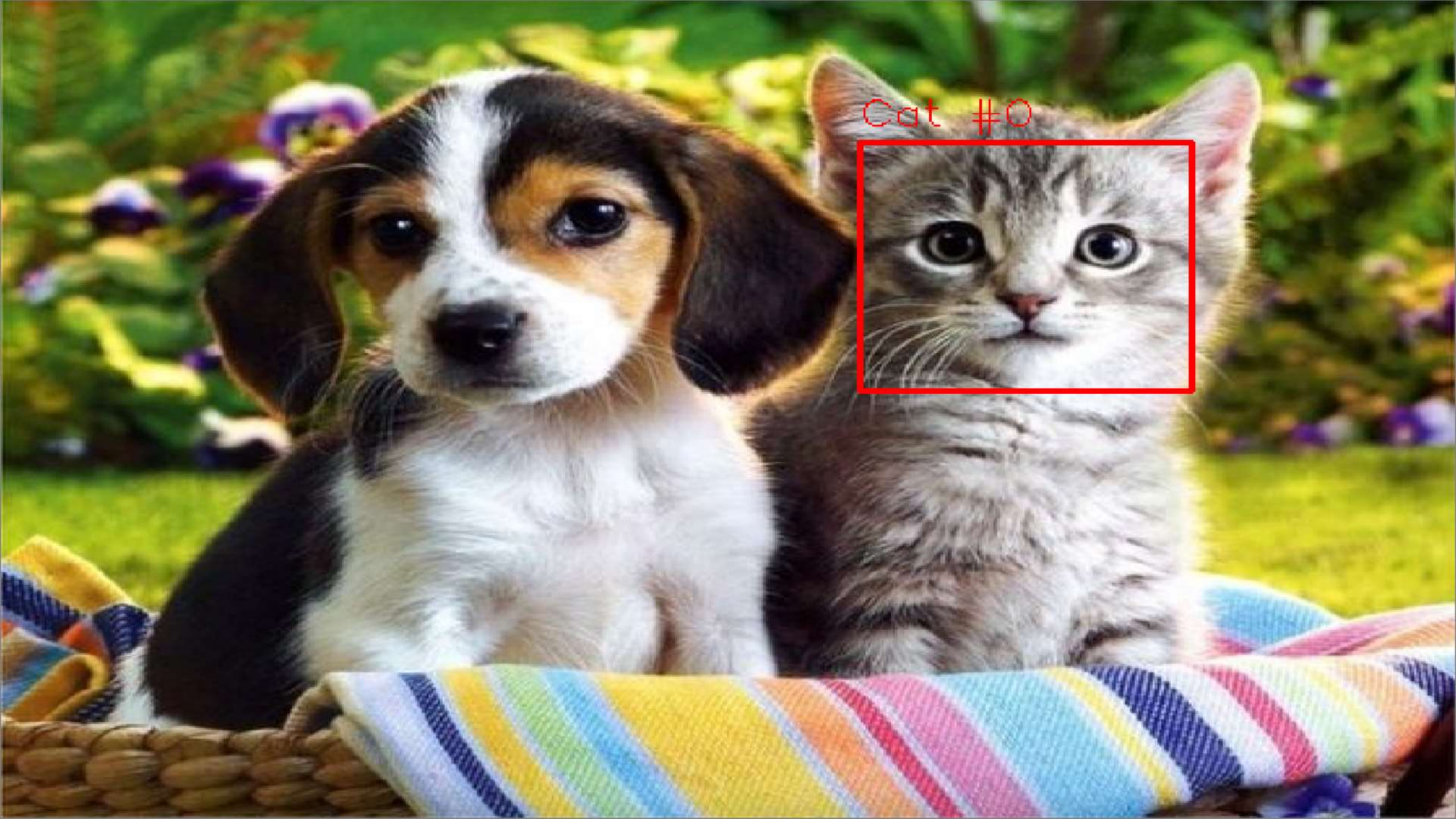
#PUEAcademyDay19

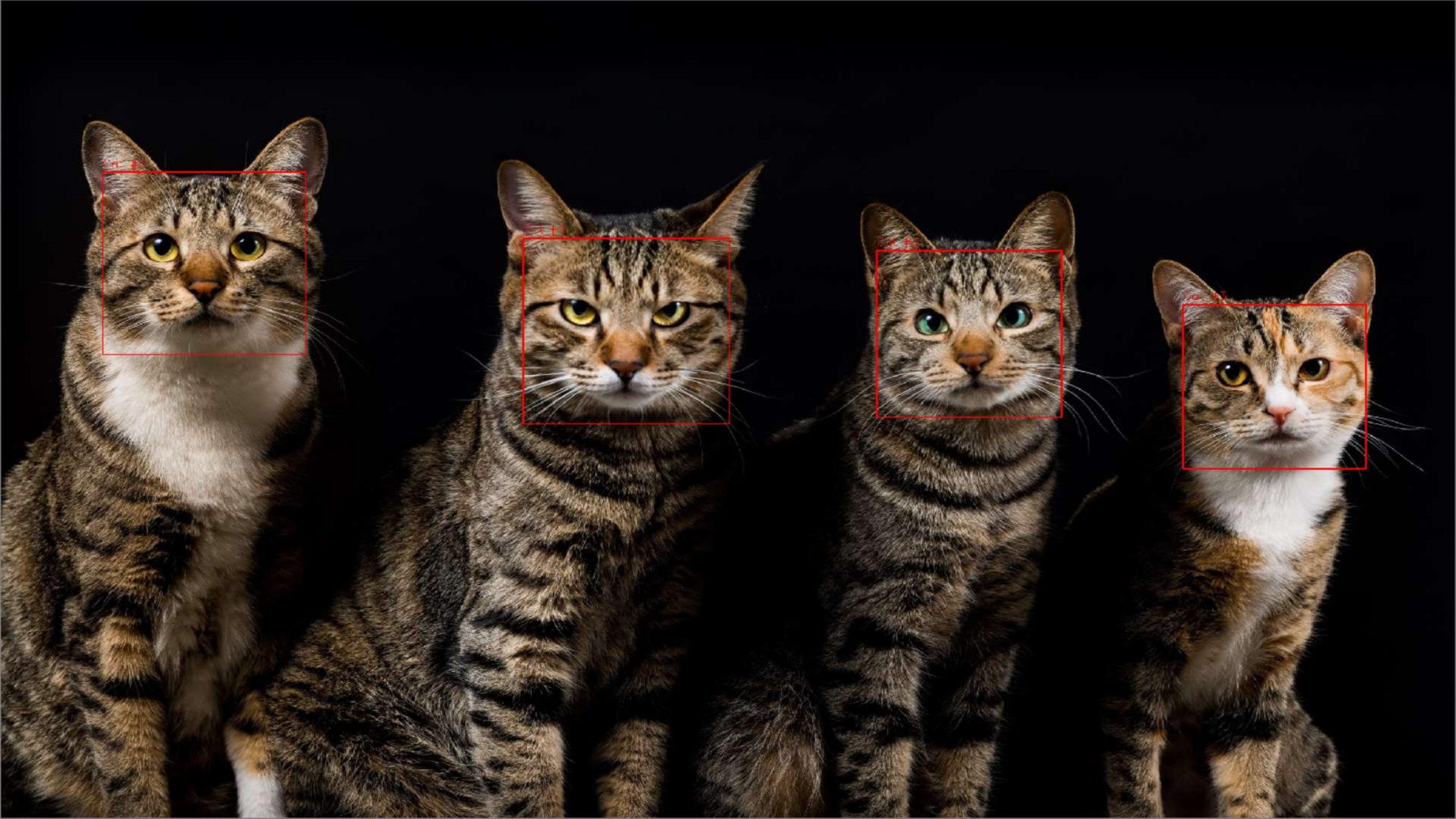
```
# In order to build cats recognition system, we use the built-in
# Haar cascade classifiers in OpenCV.
# https://github.com/opencv/opencvdata/haarcascades
# This folder contains trained classifiers for detecting objects of a
# particular type,  e.g. faces (frontal, profile), pedestrians etc
catsClassifier = cv2.CascadeClassifier(args["cclassifier"])

# “sliding window” approach
# These classifiers work by scanning an image from left to right, and
# top to bottom, at varying scale sizes.
cats = catsClassifier.detectMultiScale(gray, scaleFactor = 1.05,
                                       minNeighbors = 4, minSize = (30,30))
print(f"I found {len(cats)} cats(s)")

# loop over the cat faces and draw a rectangle surrounding each
for (i, (x, y, w, h)) in enumerate(cats):
    cv2.rectangle(image, pt1 = (x, y), pt2 = (x + w, y + h), …)
    cv2.putText(image, text = f"Face #{i}", org = (x, y - 10), …)

# Display the image to our screen
cv2.imshow(windowName, image)

# Wait for a key press to finish program
cv2.waitKey(0)

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```

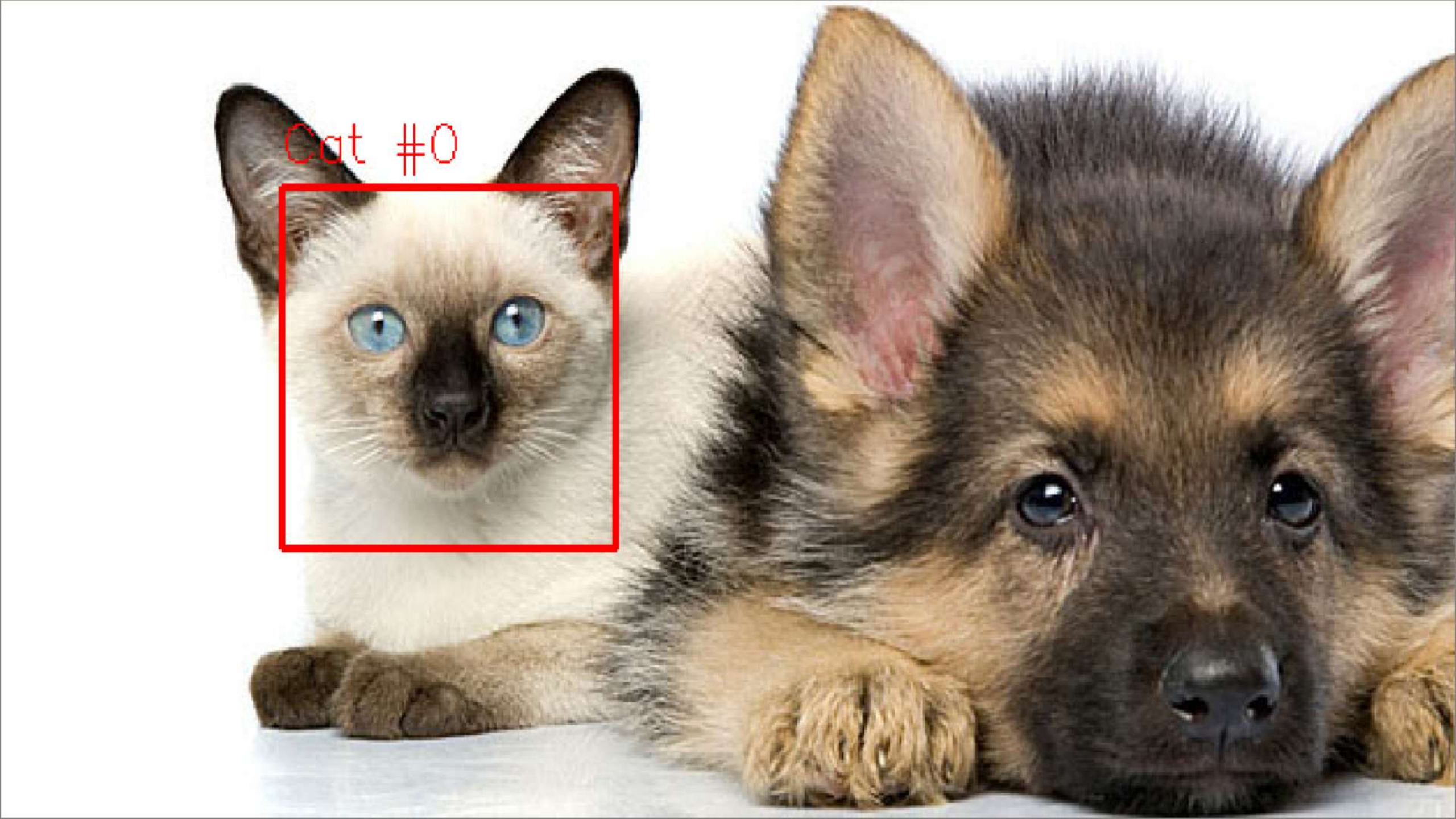

Cat #1
Cat #2
Cat #0


Cat #0


Cat #0


Cat #0




Cat #0

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

video_faces_detection.py


```
# Import the necessary packages
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-fc", "--fclassifier", required = False,
            default="classifiers/haarcascade_frontalface_default.xml",
            help = "path to where the face cascade resides")
ap.add_argument("-v", "--video", required = False,
                help = "path to where the video resides")
ap.add_argument("-f", "--full", required = False, action='store_true’,
                help = "use full screen mode")
args = vars(ap.parse_args())

# Configure windows
windowName = "Video: Cam Detector“
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)

# Loading the video using OpenCV
video = cv2.VideoCapture(args["video"])

# Face recognition system: built-in Haar  cascade
facesClassifier = cv2.CascadeClassifier(args["fclassifier"])
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

video_faces_detection.py

#PUEAcademyDay19

```
# Start to lopping over all frames in the video.
# At most basic level, a video is simply a sequence of images
while video.isOpened():
	# Read next frame
	(grabbed, frame) = video.read()

	# Convert frame to grayscale
	gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

	# Detecting the actual faces in the frame
	faces = facesClassifier.detectMultiScale(gray, scaleFactor =
				1.1, minNeighbors = 5, minSize = (30,30))

	# loop over the faces and draw a rectangle surrounding each
	for (x, y, w, h) in faces:
		cv2.rectangle(frame, pt1 = (x, y), pt2 = (x + w, y + h),
				color = (0, 0, 255), thickness = 2)

	# Display the output
	cv2.imshow(windowName, frame)

	# Wait for a key press to finish program

# The reference to the video is released
video.release()

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```


Himno España Debut Lopetegui partido Belgica-España 1-9-2016
Pausa (k)
0:23 / 1:12
Desliza la página para ver más detalles


DIRECTO Plató Debate Atresmedia
EL DEBATE DECISIVO: ÁGIL Y MÁS CONFRONTACIÓN

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

cam_faces_detection.py

#PUEAcademyDay19

```
# Import the necessary packages
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-fc", "--fclassifier", required = False,
                default="classifiers/haarcascade_frontalface.xml",
                help = "path to where the face cascade resides")
ap.add_argument("-m", "--mirror", required = False,
                action='store_true', help = "enable mirror mode")
ap.add_argument("-f", "--full", required = False, action='store_true',
                help = "use full screen mode")
args = vars(ap.parse_args())

# Configure windows
windowName = "Webcam: Face Detector"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)

# Loading the video (webcam) using OpenCV
cam = cv2.VideoCapture(0)

# Face recognition system: built-in Haar  cascade
facesClassifier = cv2.CascadeClassifier(args["fclassifier"])
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

cam_faces_detection.py

#PUEAcademyDay19

```
# Start to lopping over all frames in the webcam.
# At most basic level, a video is simply a sequence of images
while True:
      # Read next frame
      (grabbed, frame) = cam.read()

      # Mirror effect: Flip on Y axis
      if (args["mirror"]): frame = cv2.flip(frame,1)
      # Convert frame to grayscale
      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

      # Detecting the actual faces in the frame
      faces = facesClassifier.detectMultiScale(gray, scaleFactor =
                              1.1, minNeighbors = 5, minSize = (30,30))

      # loop over the faces and draw a rectangle surrounding each
      for (x, y, w, h) in faces:
            cv2.rectangle(frame, pt1 = (x, y), pt2 = (x + w, y + h),
                        color = (0, 0, 255), thickness = 2)

      # Display the output
      cv2.imshow(windowName, frame)

# The reference to the video is released
cam.release()

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

eyes_tracking.py


```
# Import the necessary packages
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-fc", "--fclassifier", required = False,
      default="classifiers/haarcascade_frontalface_default.xml",
      help = "path to where the face cascade resides (optional)")
ap.add_argument("-ec", "--eclassifier", required = False,
      default="classifiers/haarcascade_eye.xml",
      help = "path to where the eyes cascade resides (optional)")
ap.add_argument("-v", "--video", required = False,
      help = "path to where the video resides (optional)")
ap.add_argument("-m", "--mirror", required = False,
      action='store_true', help = "enable mirror mode (optional)")
ap.add_argument("-f", "--full", required = False, action='store_true',
                help = "use full screen mode (optional)")
args = vars(ap.parse_args())

# Configure windows
windowName = "Eyes Tracking"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

eyes_tracking.py


```
# If --video is not supplied we will read video from the webcam.
# Otherwise, open the video file pointed by the --video argument
if (args["video"]):
	video = cv2.VideoCapture(args["video"])
else:
	video = cv2.VideoCapture(0)

# In order to build face and eyes recognition system, we use the built-
in Haar cascade classifiers in OpenCV.
facesClassifier = cv2.CascadeClassifier(args["fclassifier"])
eyesClassifier = cv2.CascadeClassifier(args["eclassifier"])

# Display the video or webcam to our screen
while True:
	(grabbed, frame) = video.read()

	# Detect if video finished
	if (args["video"] and not grabbed): break

	# Mirror effect on webcam: Flip on Y axis
	if (not args["video"] and args["mirror"]):
		frame = cv2.flip(frame,1)

	# Convert frame to grayscale
	gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Hands-On Labs

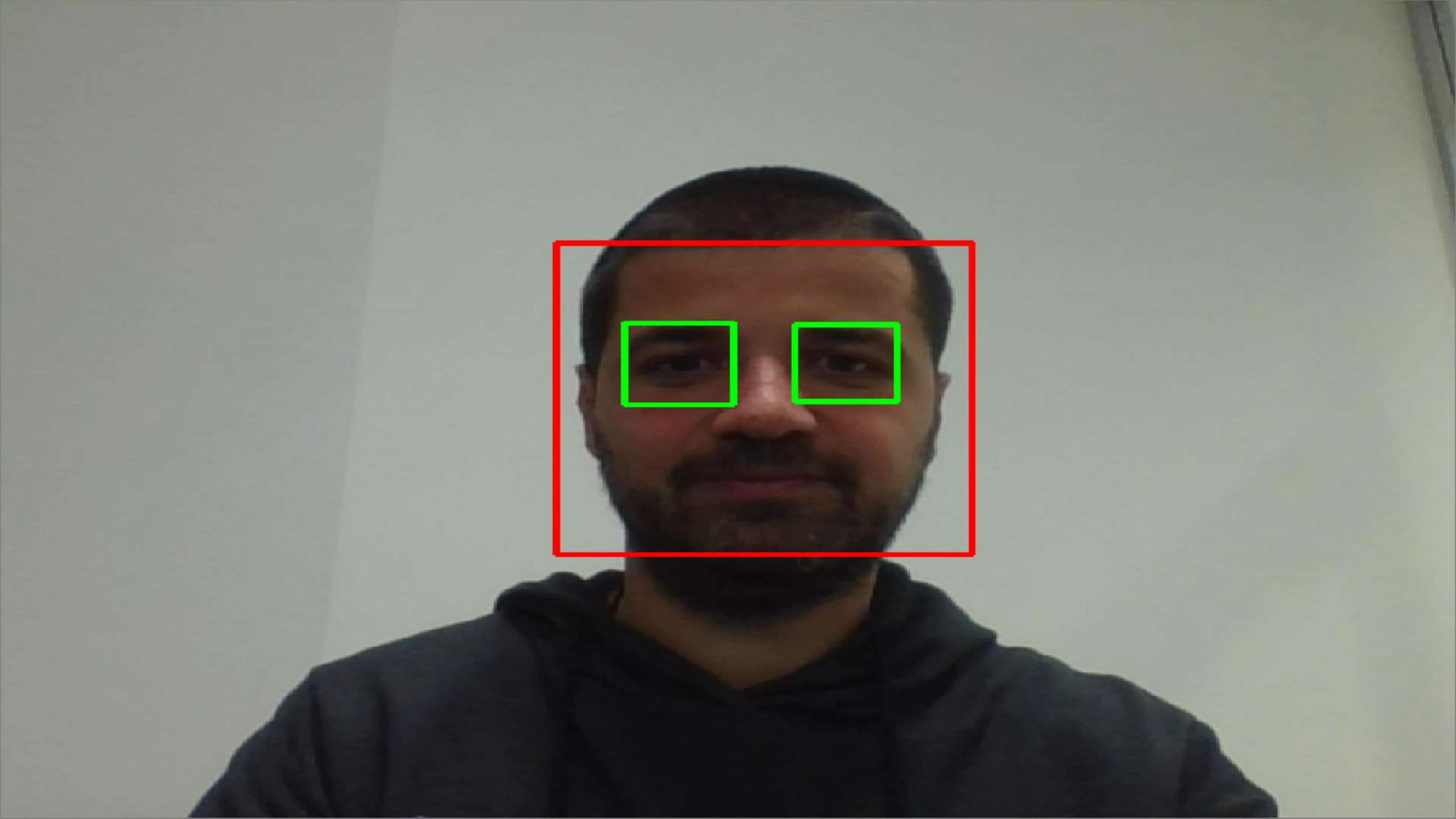
OpenCV

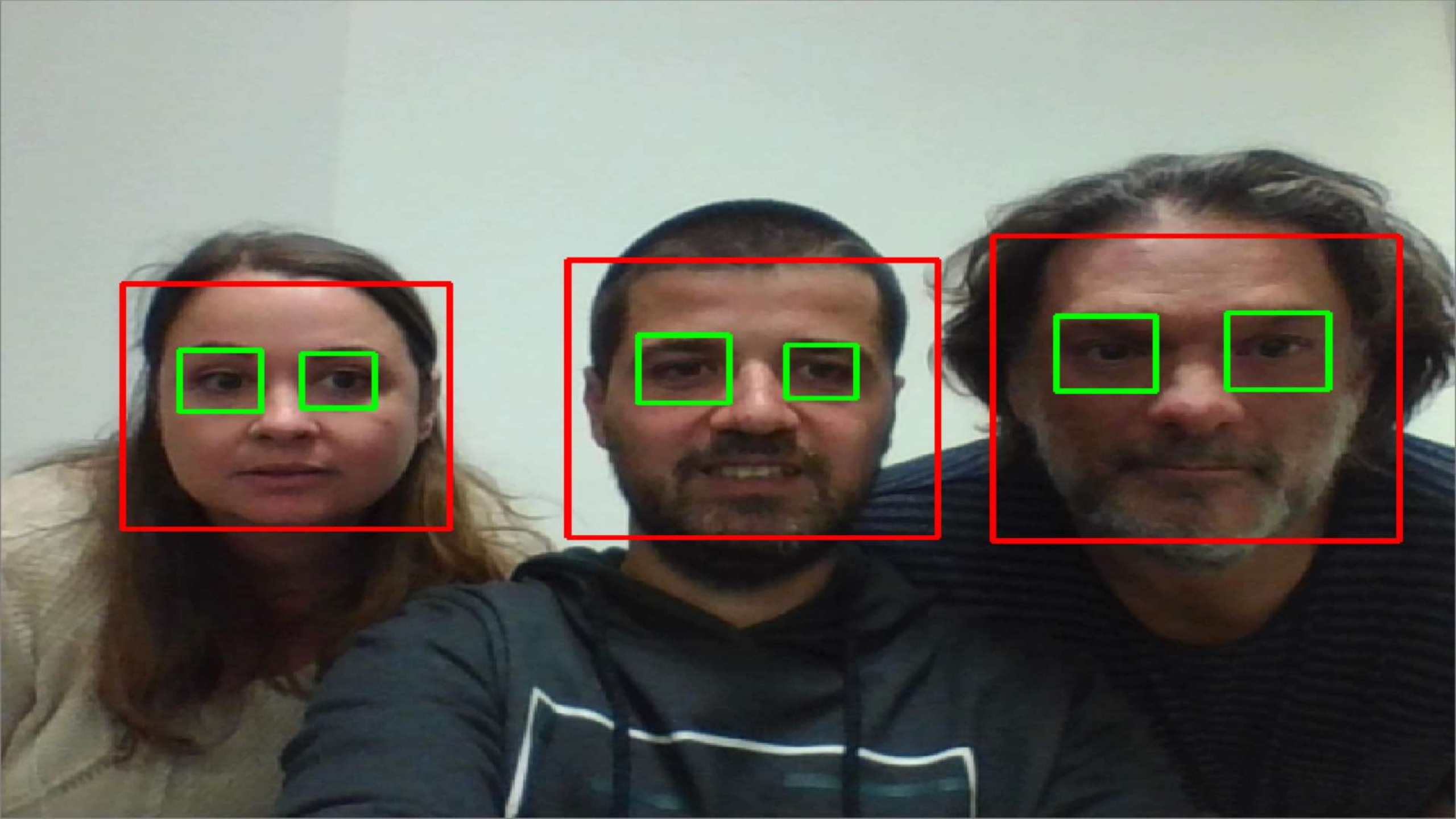
python™

PUE ACADEMY Day

eyes_tracking.py


```
        # Detecting the actual faces in the frame
        faces = facesClassifier.detectMultiScale(gray,
                scaleFactor = 1.1,  minNeighbors = 5, minSize = (30,30))

        # loop over the faces
        for (fx, fy, fw, fh) in faces:
                # Draw a rectangle surrounding each
                cv2.rectangle(frame, pt1 = (fx, fy), …)

                # Extract the face region of interest (ROI)
                faceROI = gray[fy:fy + fh, fx:fx + fw]

                # Detecting the actual eyes in the frame
                eyes = eyesClassifier.detectMultiScale(faceROI,
                  scaleFactor = 1.1, minNeighbors = 10, minSize = (20,20))

                # loop over the eyes and draw
                for (ex,ey,ew,eh) in eyes:
                        cv2.rectangle(frame, pt1 = (fx + ex, fy + ey), …)

        # Display the output
        cv2.imshow(windowName, frame)

# The reference to the video is released
video.release()

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```





Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

obj_tracking.py

#PUEAcademyDay19

```
# Import the necessary packages
import argparse
import numpy as np
import time
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", required = False,
     help = "path to where the video resides (optional)")
ap.add_argument("-t", "--threshold", required = False,
     action='store_true', help = "show threshold images (optional)")
ap.add_argument("-m", "--mirror", required = False,action='store_true’,
     help = "enable mirror mode (optional)")
ap.add_argument("-f", "--full", required = False, action='store_true’,
     help = "use full screen mode (optional)")
args = vars(ap.parse_args())

# Configure windows
windowName = "Object Tracking"
if (args["full"]):
    cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
                          cv2.WINDOW_FULLSCREEN)
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

obj_tracking.py


```
# The object that will be tracking is a "yellow" object.
colorLower = (20, 100, 100)
colorUpper = (30, 255, 255)

# If --video is not supplied we will read video from the webcam.
# Otherwise, open the video file pointed by the --video argument
if (args["video"]):
	video = cv2.VideoCapture(args["video"])
else:
	video = cv2.VideoCapture(0)

# Display the video or webcam to our screen
while True:
	(grabbed, frame) = video.read()

	# Detect if video finished
	if (args["video"] and not grabbed): break

	# Mirror effect on webcam: Flip on Y axis
	if (not args["video"] and args["mirror"]):
		frame = cv2.flip(frame,1)

	# Find shades of the yellow in the frame using the cv2.inRange
	# The result is a thresholded image with pixels falling within
	# the upper and lower set to white and pixels that do not fall
	# into this range se as black
	threshold = cv2.inRange(frame, colorLower, colorUpper)
	threshold = cv2.GaussianBlur(threshold, (3,3), 0)
```

Hands-On Labs

OpenCV

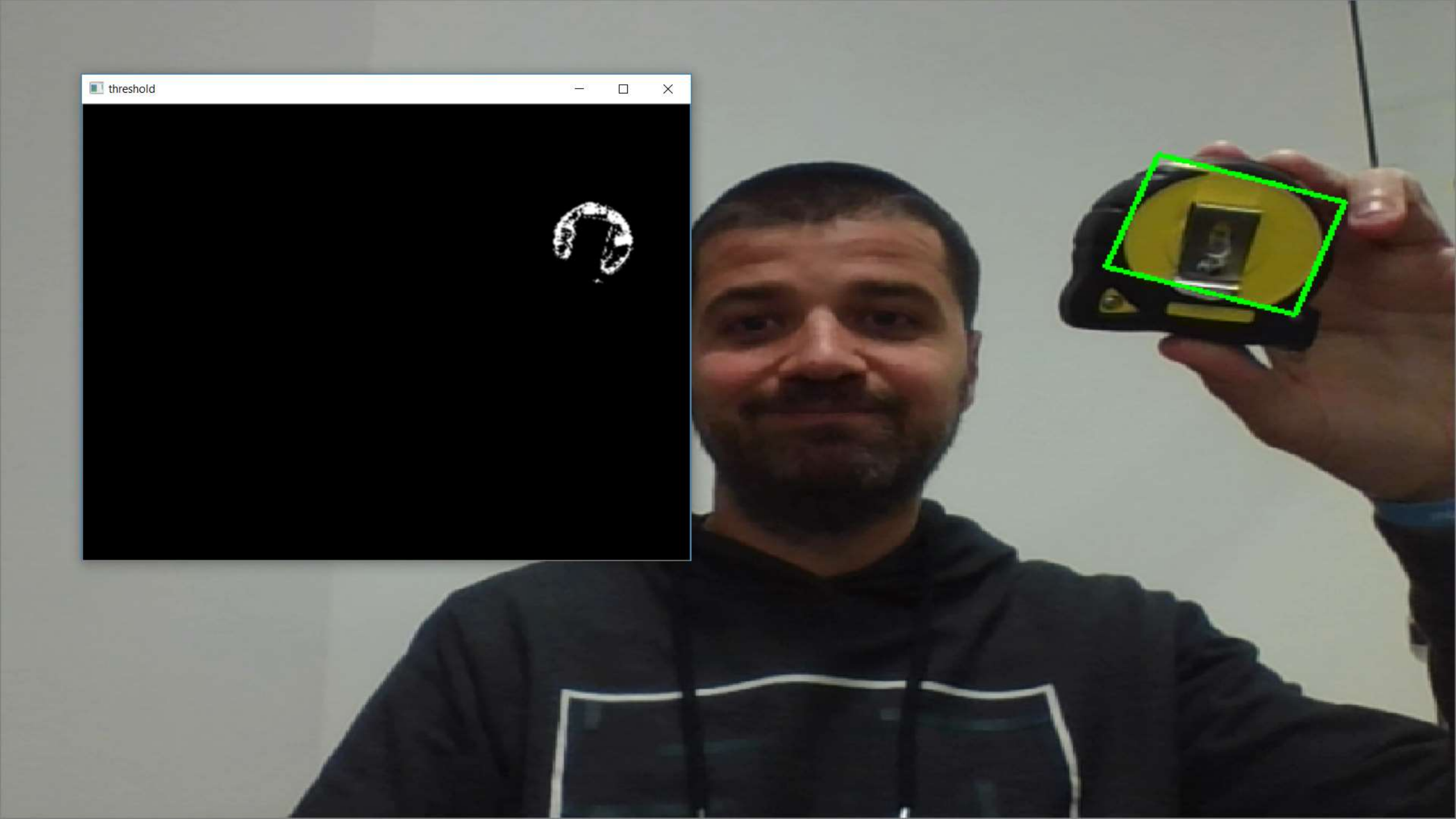
python™

PUE ACADEMY Day

obj_tracking.py

#PUEAcademyDay19

```
        # Display threshold images
        if (args["threshold"]): cv2.imshow("threshold", threshold)

        # Now that we have the thresholded image, we needs to find the
        # largest contour in the image
        (contours, h) = cv2.findContours(threshold.copy(),
             mode = cv2.RETR_EXTERNAL, method = cv2.CHAIN_APPROX_SIMPLE)
        if (len(contours) > 0) :
              contour = sorted(contours, key = cv2.contourArea,
                                  reverse = True)[0]

              # To draw the contours, cv2.drawContours function is used.
              rect = np.int32(cv2.boxPoints(cv2.minAreaRect(contour)))
              cv2.drawContours(frame, contours = [rect],
                    contourIdx = -1, color = (0, 255, 0), thickness = 2)

        cv2.imshow(windowName, frame)

        # Wait for a key press to finish program
        key = cv2.waitKey(1)
        if (key in [ord("q"), 27] or …

# The reference to the video is released
video.release()

# Any open window created by OpenCV are closed
cv2.destroyAllWindows()
```


threshold


"HANG ZEN"
QUIKSILVER

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

multi_object_tracking.py

#PUEAcademyDay19

```
# import the necessary packages
import argparse
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", required = False,
                help = "path to where the video resides (optional)")
ap.add_argument("-m", "--mirror", required = False,
        action='store_true', help = "enable mirror mode (optional)")
ap.add_argument("-f", "--full", required = False, action='store_true',
                 help = "use full screen mode (optional)")
ap.add_argument("-t", "--tracker", type=str, required = False,
       default="csrt",  help="OpenCV object tracker type (oprtional)")
args = vars(ap.parse_args())

# Configure windows
windowName = "Multi Tracking"
if (args["full"]):
       cv2.namedWindow(windowName, cv2.WND_PROP_FULLSCREEN)
       cv2.setWindowProperty(windowName, cv2.WND_PROP_FULLSCREEN,
       cv2.WINDOW_FULLSCREEN)
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

multi_object_tracking.py


```
# Initialize a dictionary that maps strings to their corresponding
# OpenCV object tracker implementations
# I recommend the following three algorithms:
# KCF: Fast and accurate
# CSRT: More accurate than KCF but slower
# MOSSE: Extremely fast but not as accurate as either KCF or CSRT
OPENCV_OBJECT_TRACKERS = {
    "csrt": cv2.TrackerCSRT_create,
    "kcf": cv2.TrackerKCF_create,
    "boosting": cv2.TrackerBoosting_create,
    "mil": cv2.TrackerMIL_create,
    "tld": cv2.TrackerTLD_create,
    "medianflow": cv2.TrackerMedianFlow_create,
    "mosse": cv2.TrackerMOSSE_create
}


# Initialize OpenCV's special multi-object tracker
# The class allows us to:
# Add new object trackers to the MultiTracker
# Update all object trackers inside the MultiTracker
trackers = cv2.MultiTracker_create()

# If --video is not supplied we will read video from the webcam.
if (args["video"]):
    video = cv2.VideoCapture(args["video"])
else:
    video = cv2.VideoCapture(0)
```

Hands-On Labs

OpenCV

python™

PUE ACADEMY Day

multi_object_tracking.py


```
# loop over frames from the video stream
while True:
    # grab the current frame, then handle if we are using a
    # VideoStream or VideoCapture object
    (grabbed, frame) = video.read()

    # Detect if video finished
    if (args["video"] and not grabbed): break

      # Mirror effect on webcam: Flip on Y axis
    if (not args["video"] and args["mirror"]):
       frame = cv2.flip(frame,1)

    # grab the updated bounding box coordinates (if any) for each
    # object that is being tracked
    # The update method will locate the object's new position and
    # return a success boolean
    # and the bounding box of the object.
    (success, boxes) = trackers.update(frame)

    # loop over the bounding boxes and draw then on the frame
    for box in boxes:
        (x, y, w, h) = [int(v) for v in box]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # show the output frame
    cv2.imshow(windowName, frame)
```

Hands-On Labs

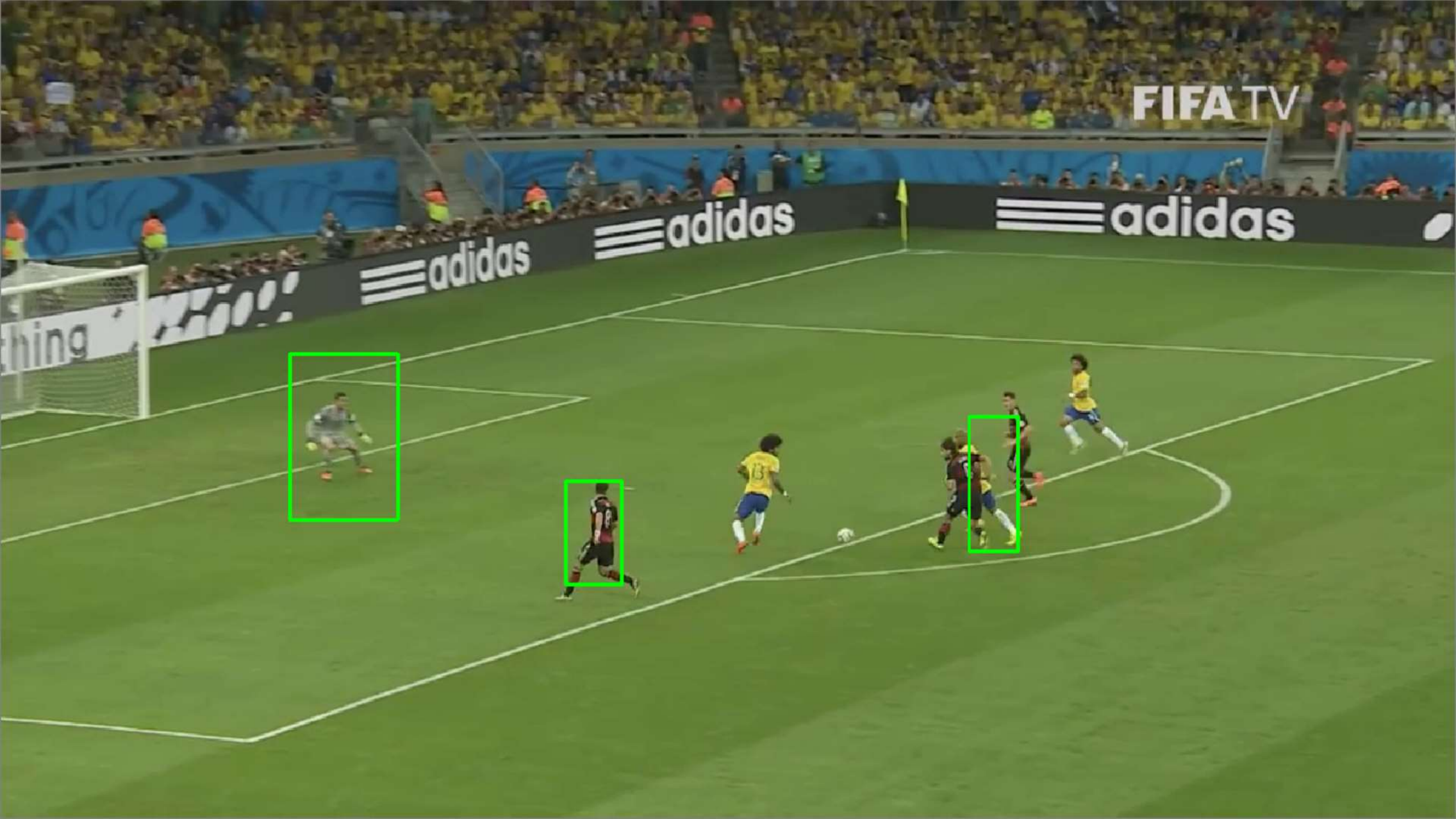
OpenCV

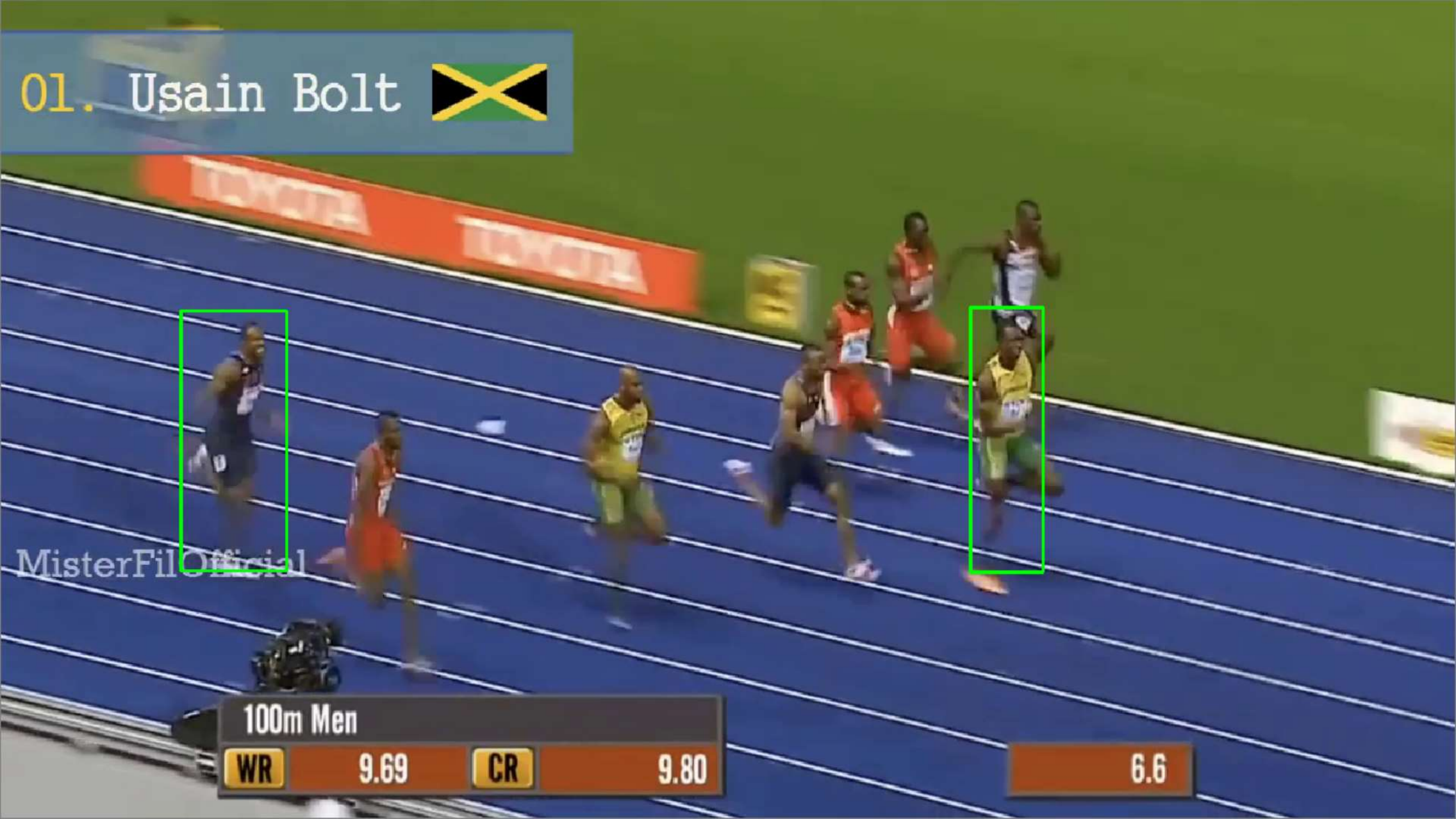
python™

PUE ACADEMY Day

multi_object_tracking.py


```
    key = cv2.waitKey(1)

    # if the 's' key is selected, we are going to "select" a
    # bounding box to track
    if key == ord("s"):

        # select the bounding box of the object we want to track (make
        # sure you press ENTER or SPACE after selecting the ROI)
        box = cv2.selectROI(windowName, frame, fromCenter=False,
                            showCrosshair=True)

        # create a new object tracker for the bounding box and add it
        # to our multi-object tracker
        tracker = OPENCV_OBJECT_TRACKERS[args["tracker"]]()
        trackers.add(tracker, frame, box)

    # if the `q` key was pressed, break from the loop
    elif (key in [ord("q"), 27] or …):
        break

video.release()

# close all windows
cv2.destroyAllWindows()
```


FIFA TV
adidas
adidas
adidas


01. Usain Bolt
MisterFilOfficial
100m Men
WR
9.69
CR
9.80
6.6

Hands-On Labs

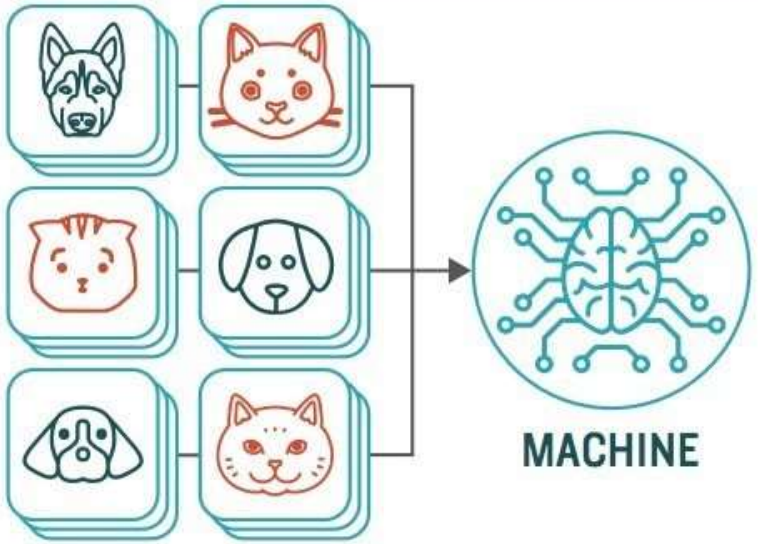
OpenCV

python™

How **Unsupervised** Machine Learning Works

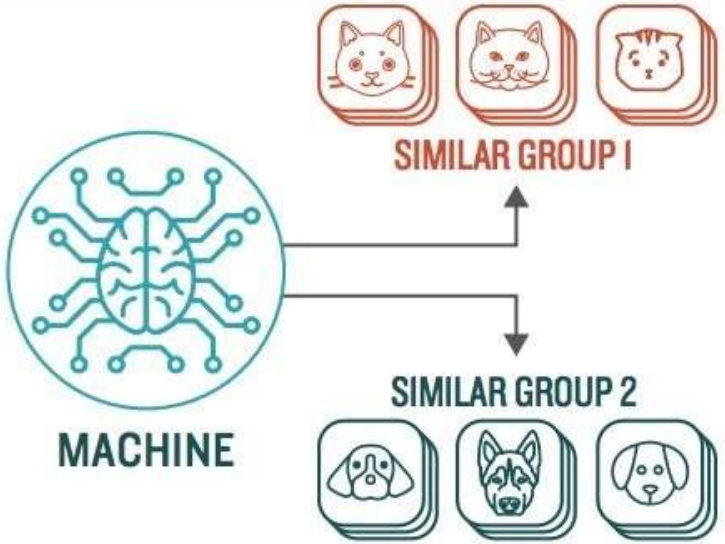
STEP 1

Provide the machine learning algorithm uncategorized, unlabeled input data to see what patterns it finds

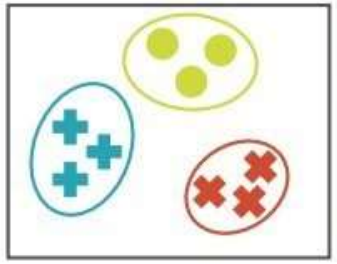


STEP 2

Observe and learn from the patterns the machine identifies



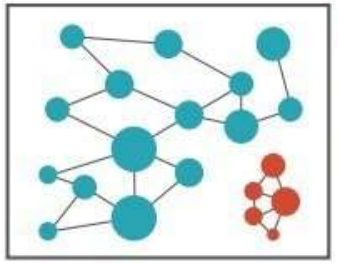
TYPES OF PROBLEMS TO WHICH IT'S SUITED



CLUSTERING

Identifying similarities in groups

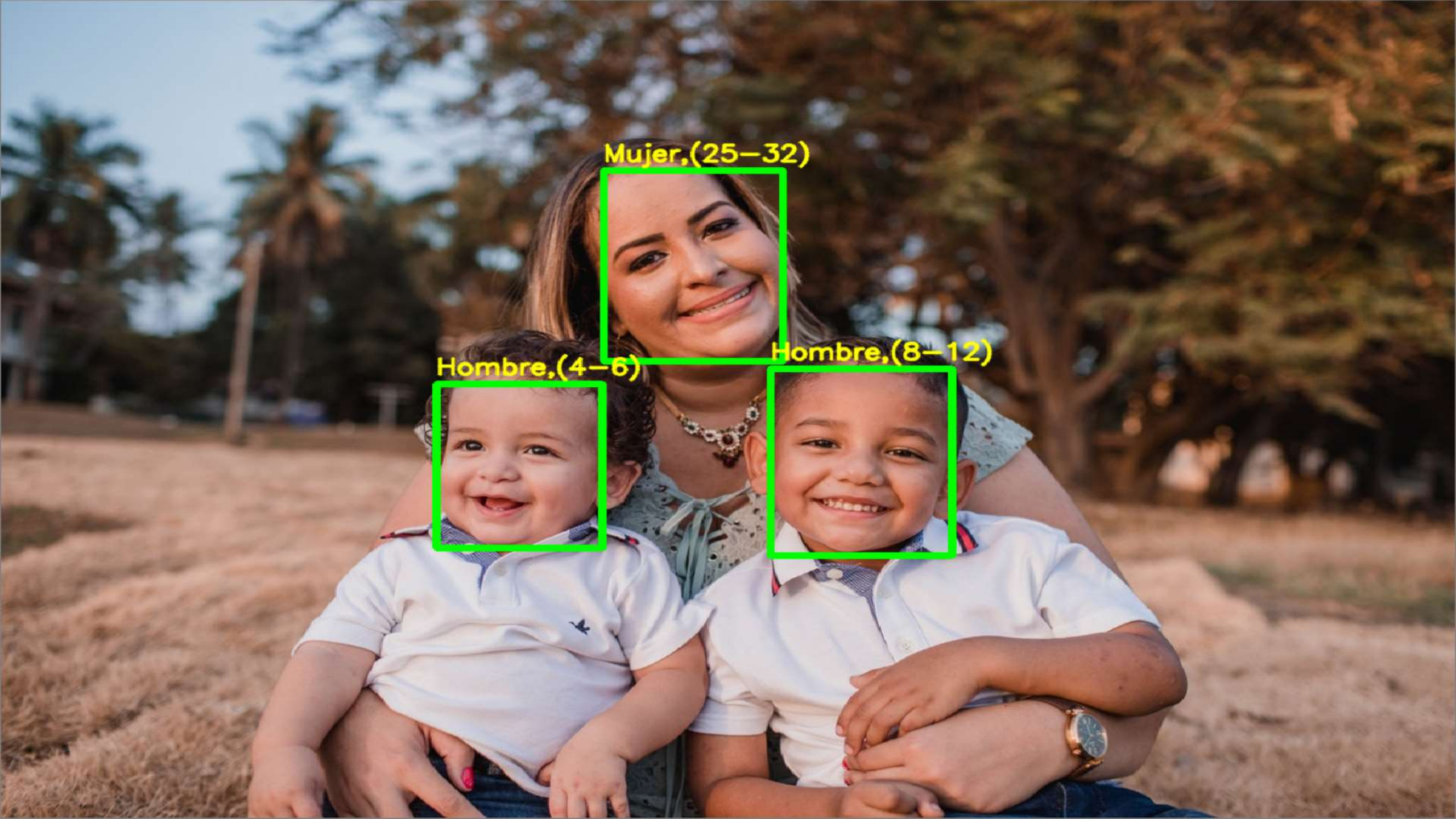
For Example: Are there patterns in the data to indicate certain patients will respond better to this treatment than others?



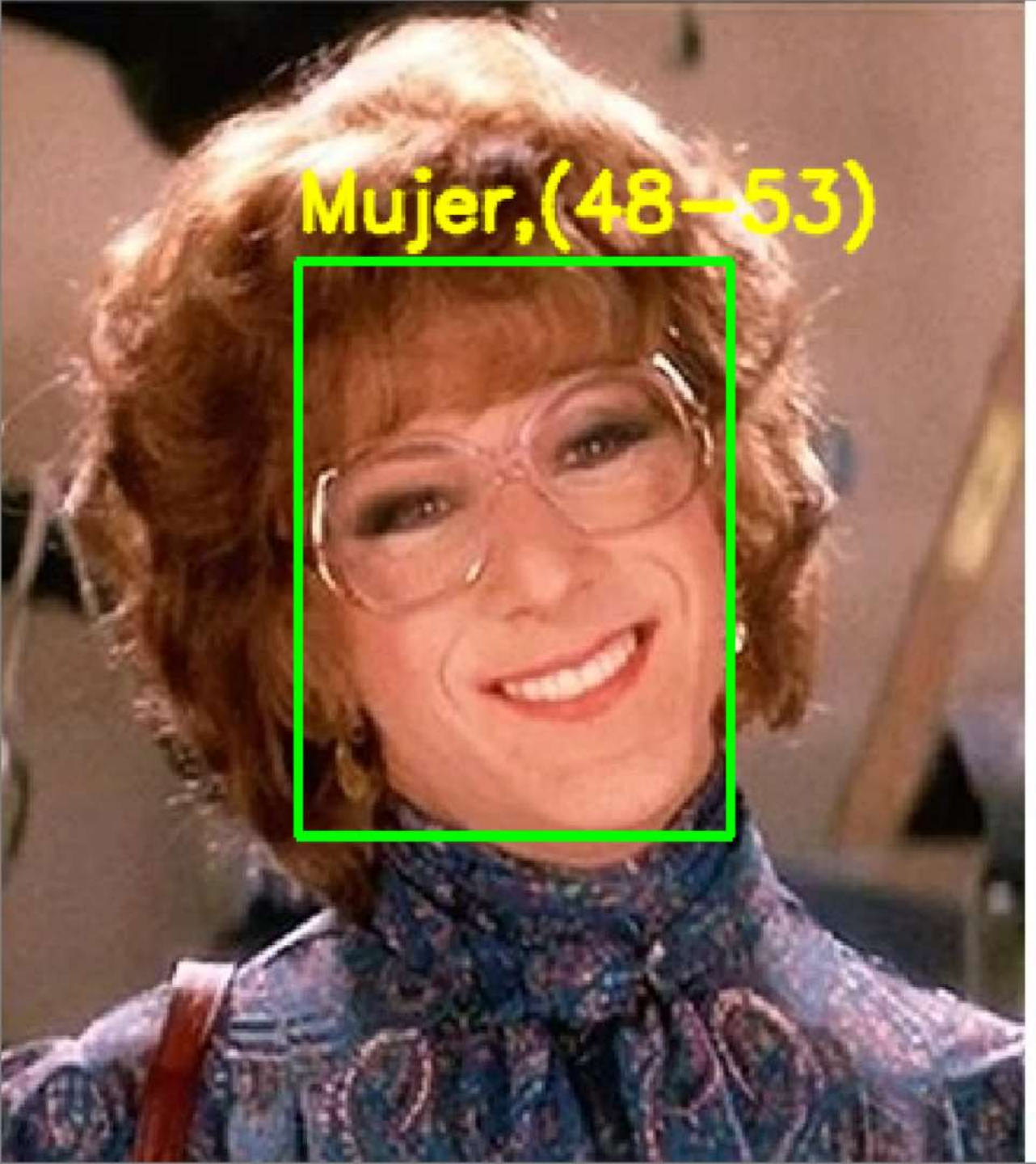
ANOMALY DETECTION

Identifying abnormalities in data

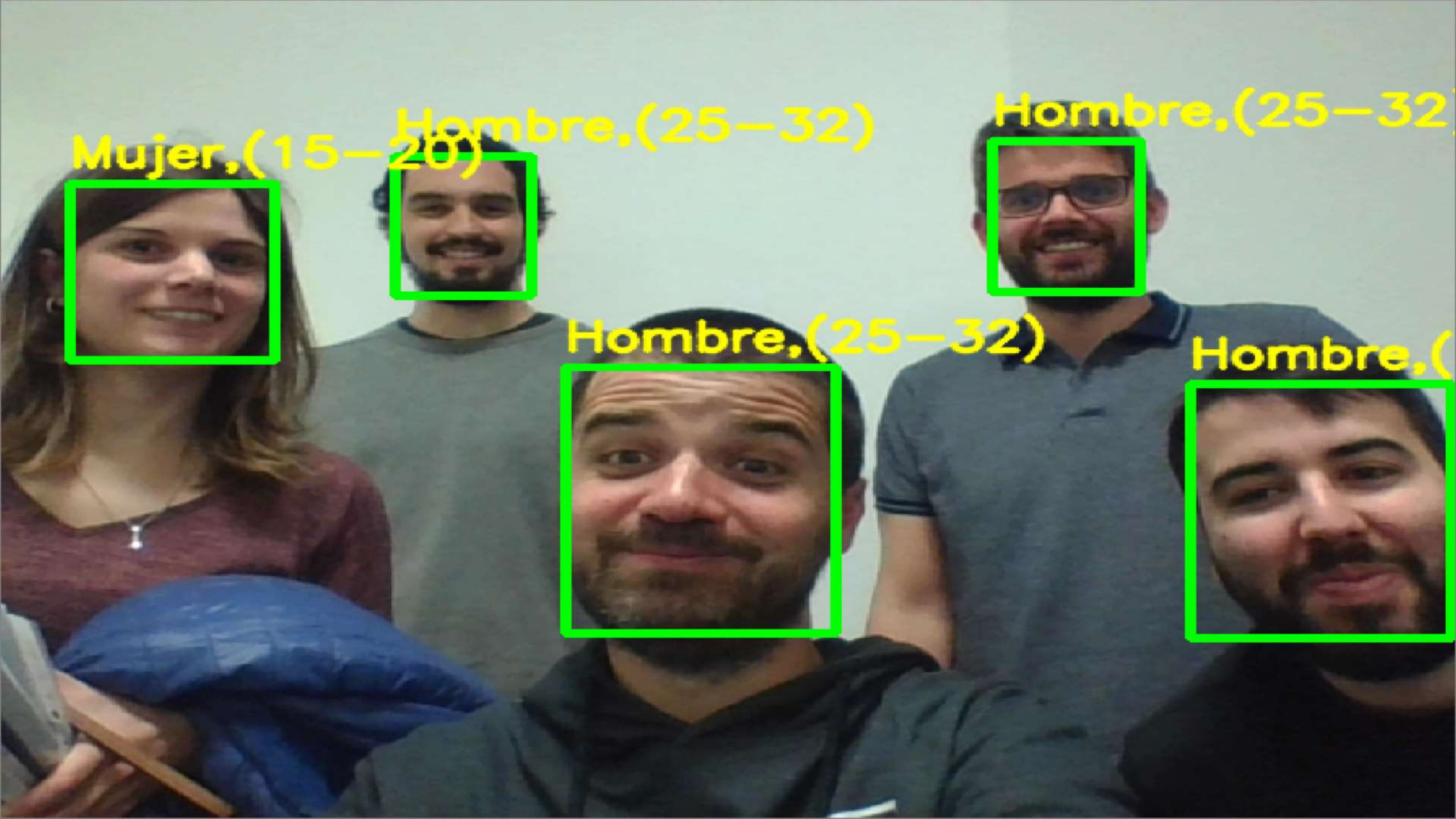
For Example: Is a hacker intruding in our network?

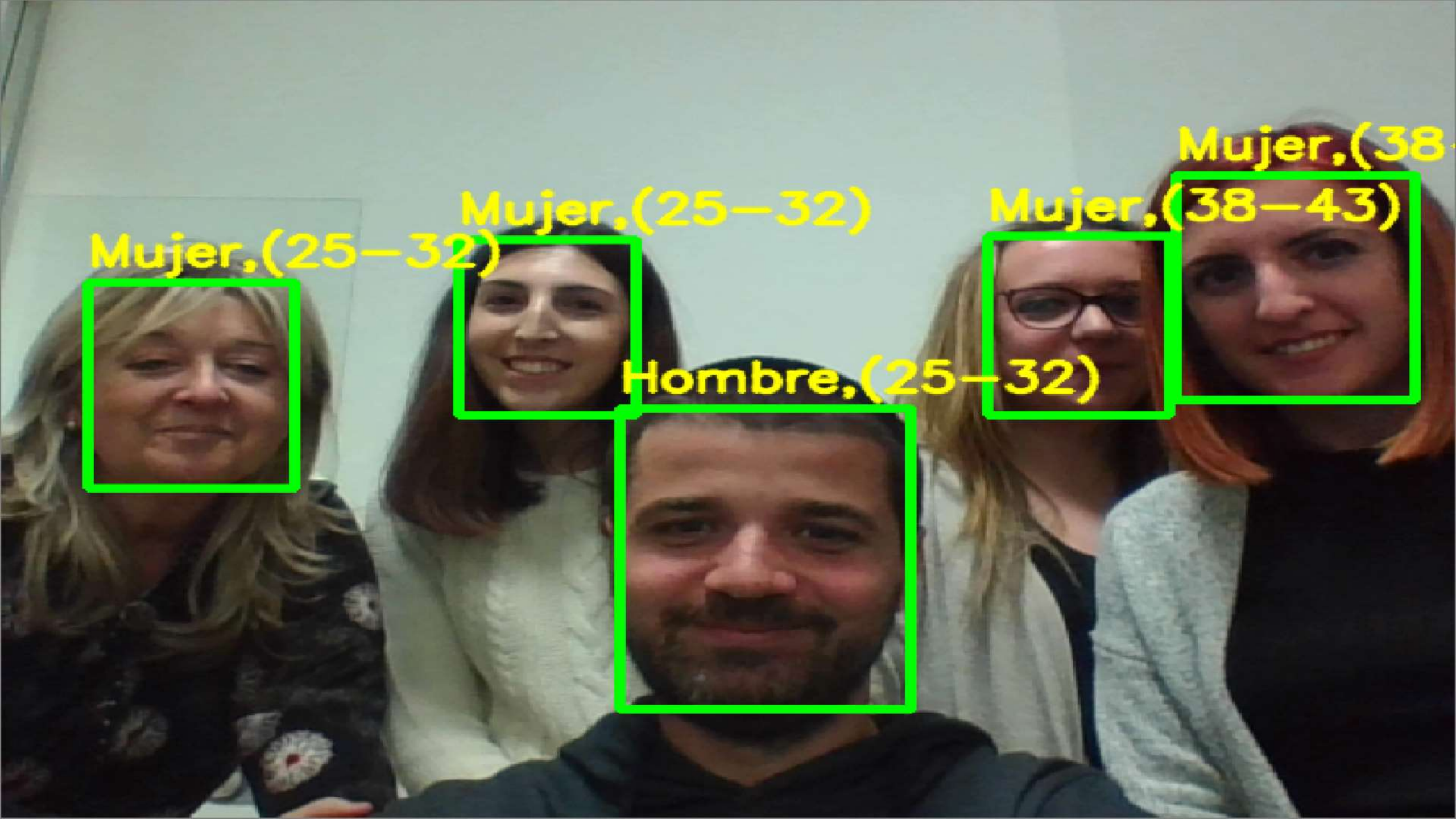

Mujer,(25–32)
Hombre,(4–6)
Hombre,(8–12)

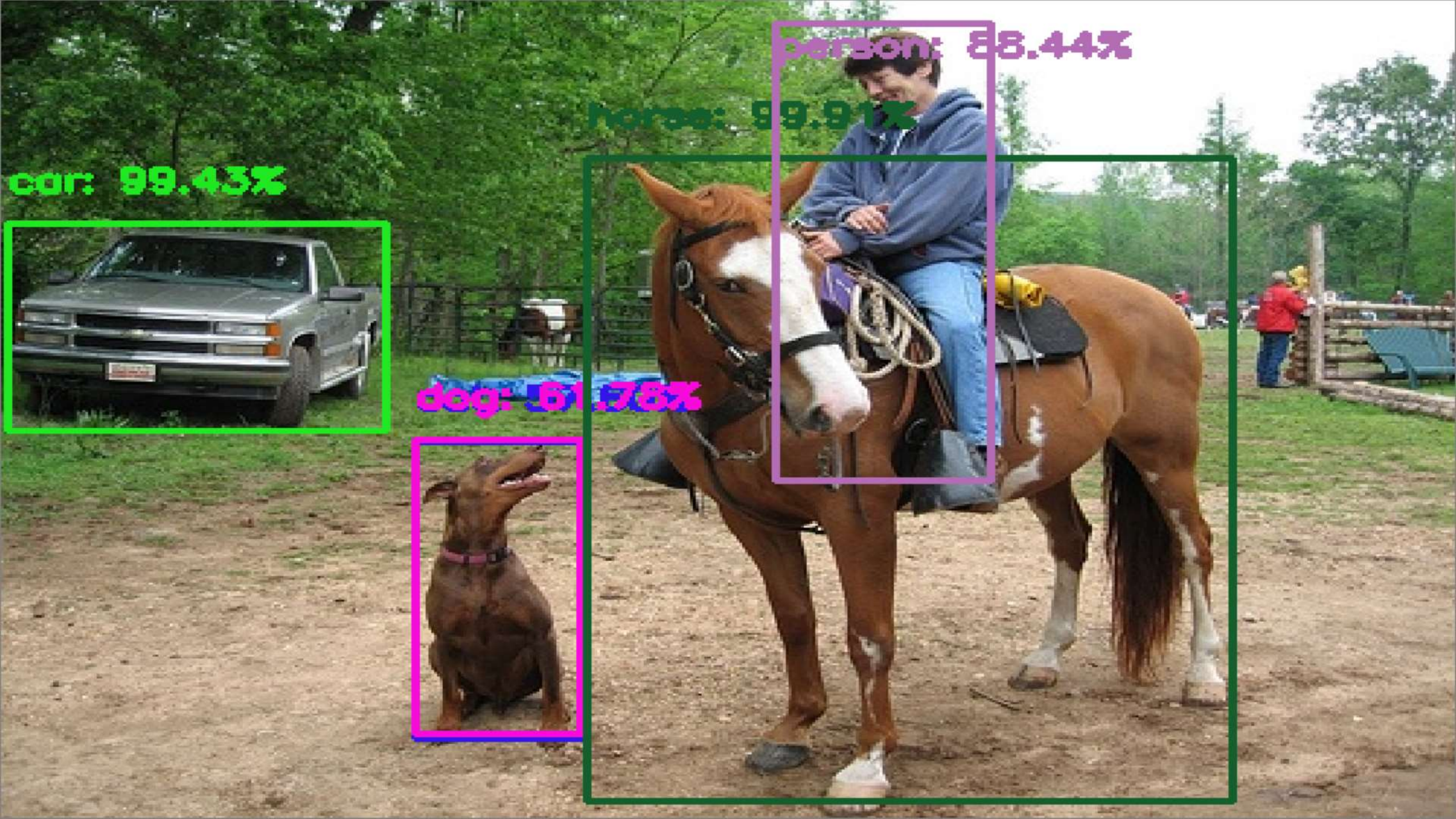

Hombre,(25-32)
Mujer,(15-20)

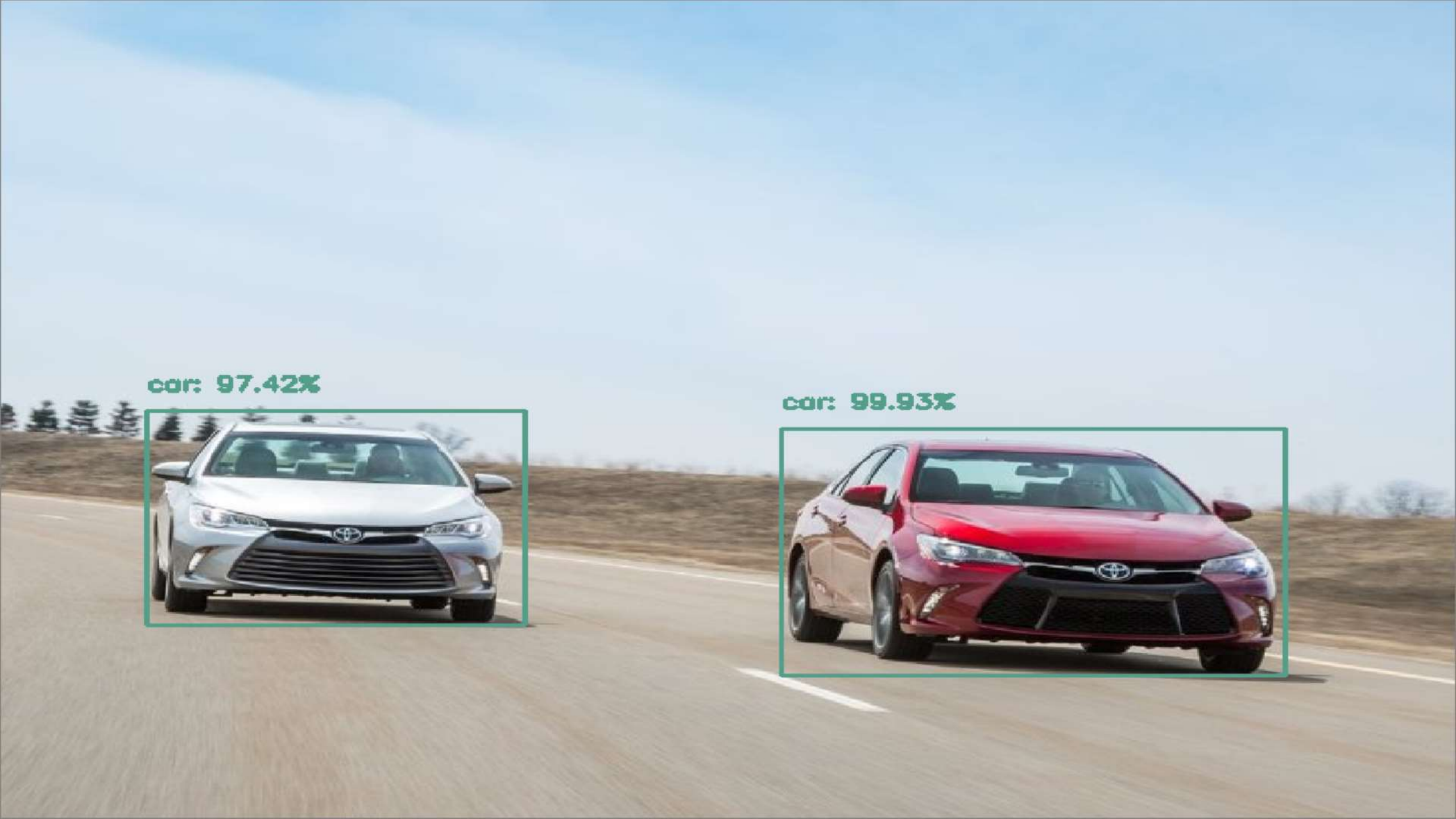

Mujer,(48-53)


Hombre,(48-53)


Mujer,(15-20)
Hombre,(25-32)
Hombre,(25-32)
Hombre,(25-32)
Hombre,(


Mujer,(25-32)
Mujer,(25-32)
Hombre,(25-32)
Mujer,(38-43)
Mujer,(38


person: 88.44%
horse: 99.91%
car: 99.43%
dog: 61.78%


car: 97.42%
car: 99.93%


Cloud AI solutions
Cloud Job Discovery
Contact Center
Recommendation Engine
ML professional services & partners
ASL
Professional services organization
Partner ecosystem
Cloud AI building blocks
Sight
Cloud Video Intelligence
Cloud Vision
NEW
AutoML Vision
Language
Cloud Natural Language
AutoML Natural Language
NEW
Cloud Translation
AutoML Translation
NEW
Conversation
Cloud Speech-to-Text
Dialogflow Enterprise
Cloud Text-to-Speech
Cloud AI platform
Cloud ML Engine
Cloud Dataflow
Cloud Dataproc
ML libraries
Tensorflow
Spark
beam
R
K
Spark MLlib
learn
Kaggle / Datasets
kaggle
Datasets
Google Cloud

Cloud Vision

Extrae información valiosa a partir de imágenes con nuestros potentes modelos de API ya preparados o prepara modelos de visión personalizados de forma sencilla con AutoML Vision[BETA]

PRUÉBALO GRATIS

Consulta la documentación de este producto.

Potente análisis de imágenes

La API Cloud Vision utiliza modelos ya preparados y permite crear modelos personalizados y flexibles que se adaptan a cada caso práctico a través de AutoML Vision.

La **API Cloud Vision** permite que los desarrolladores comprendan el contenido de una imagen mediante el encapsulado de potentes modelos de aprendizaje automático en una API REST fácil de usar. Esta API clasifica las imágenes rápidamente en miles de categorías (por ejemplo, "barco de vela"), detecta objetos y caras determinados dentro de dichas imágenes y es capaz de leer las palabras impresas que contengan. De este modo, puedes crear metadatos en tu catálogo de imágenes, moderar el contenido ofensivo o habilitar nuevas situaciones de marketing mediante el análisis de opinión en imágenes.

Gracias a la **versión beta de AutoML Vision**, los desarrolladores con una experiencia limitada en el ámbito del aprendizaje automático pueden preparar modelos personalizados de alta calidad. Una vez subidas y etiquetadas las imágenes, AutoML Vision prepara un modelo que puede escalarse según sea necesario para adaptarse a necesidades específicas. AutoML Vision ahorra tiempo a la hora de crear modelos que, además

Hands-On Labs

Google Cloud Platform

python™

```
> pip install google-cloud-vision

Collecting google-cloud-vision
  Downloading
https://files.pythonhosted.org/packages/f2/bf/112a0707a42596151669
3ac526725bc3f51db44fc3d02998da3ee2b82ef1/google_cloud_vision-
0.36.0-py2.py3-none-any.whl (383kB)
     |████████████████████████████████| 389kB 1.6MB/s
Collecting google-api-core[grpc]<2.0.0dev,>=1.6.0 (from google-
cloud-vision)
  Downloading
https://files.pythonhosted.org/packages/bf/e4/b22222bb714947eb459d
c91ebf95131812126a0b29d62e444be3f76dad64/google_api_core-1.9.0-
py2.py3-none-any.whl (65kB)
     |████████████████████████████████| 71kB 2.3MB/s
Collecting google-auth<2.0dev,>=0.4.0 (from google-api-
core[grpc]<2.0.0dev,>=1.6.0->google-cloud-vision)
…
Successfully installed cachetools-3.1.0 certifi-2019.3.9 chardet-
3.0.4 google-api-core-1.9.0 google-auth-1.6.3 google-cloud-vision-
0.36.0 googleapis-common-protos-1.5.9 grpcio-1.20.1 idna-2.8
protobuf-3.7.1 pyasn1-0.4.5 pyasn1-modules-0.2.5 pytz-2019.1
requests-2.21.0 rsa-4.0 six-1.12.0 urllib3-1.24.2
```

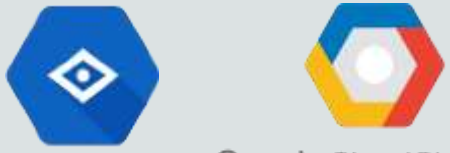
https://pypi.org/project/google-cloud-vision/

Hands-On Labs


```
> pip list

Package                    Version
-------------------------- --------
cachetools                 3.1.0
certifi                    2019.3.9
chardet                    3.0.4
google-api-core            1.9.0
google-auth                1.6.3
google-cloud-vision        0.36.0
googleapis-common-protos 1.5.9
grpcio                     1.20.1
idna                       2.8
numpy                      1.16.3
opencv-contrib-python      4.1.0.25
pip                        19.1
protobuf                   3.7.1
pyasn1                     0.4.5
pyasn1-modules             0.2.5
pytz                       2019.1
requests                   2.21.0
…
```

https://pypi.org/project/google-cloud-vision/

Hands-On Labs



Google Cloud Platform

python™

PUE ACADEMY Day

logo_detection.py

#PUEAcademyDay19

```
# Import the necessary packages
import io
import os
from google.cloud import vision
from google.cloud.vision import types
import argparse

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True,
                help = "path to where the image file resides")
args = vars(ap.parse_args())

# Authenticate API requests
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "credentials.json“

# The name of the image file to annotate
filename = os.path.join(os.path.dirname(__file__), args["image"])

# Loads the image into memory
with io.open(filename, 'rb') as file:
    content = file.read()
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

logo_detection.py

#PUEAcademyDay19

```
# Instantiates a client
client = vision.ImageAnnotatorClient()
image = types.Image(content = content)

# Performs object localization on the image file
response = client.logo_detection(image)

# Process the response
logos = response.logo_annotations

# Print JSON response
print(logos)

print(f'Number of logos found: {len(logos)}')
print("------------------------------")
for logo in logos:
  print(f'Logo: {logo.description}’)
  print(f'Score: {logo.score}’)
  print("------------------------------")
```

Hands-On Labs

logo_detection.py

Google

Hands-On Labs

PUE ACADEMY Day

logo_detection.py


```
Number of logos found: 1
------------------------------
Logo: google
Score: 0.9927250146865845
------------------------------

{
  mid: "/m/045c7b"
  description: "google"
  score: 0.9927250146865845
  bounding_poly {
    vertices {
    }
    vertices {
      x: 759
    }
    vertices {
      x: 759
      y: 439
    }
    vertices {
      y: 439
    }
  }
}
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

logo_detection.py

#PUEAcademyDay19

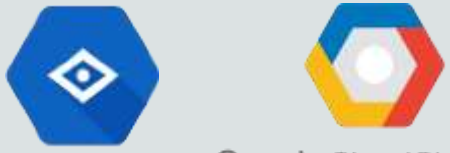
Hands-On Labs

PUE ACADEMY Day

logo_detection.py


```
Number of logos found: 1
------------------------------
Logo: mcdonalds
Score: 0.9921749830245972
------------------------------

{
   mid: "/m/07gyp7"
   description: "mcdonalds"
   score: 0.9921749830245972
   bounding_poly {
     vertices {
     }
     vertices {
       x: 923
     }
     vertices {
       x: 923
       y: 600
     }
     vertices {
       y: 600
    }
  }
}
```

Hands-On Labs



Google Cloud Platform

PUE ACADEMY Day

object_detection.py

#PUEAcademyDay19

```python
# Import the necessary packages
import io
import os
from google.cloud import vision
from google.cloud.vision import types
import argparse

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True,
                help = "path to where the image file resides")
args = vars(ap.parse_args())

# Authenticate API requests
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "credentials.json“

# The name of the image file to annotate
filename = os.path.join(os.path.dirname(__file__), args["image"])

# Loads the image into memory
with io.open(filename, 'rb') as file:
    content = file.read()
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

object_detection.py

#PUEAcademyDay19

```
# Instantiates a client
client = vision.ImageAnnotatorClient()
image = types.Image(content = content)

# Performs object localization on the image file
response = client.object_localization(image)

# Process the response
objects = response.localized_object_annotations


# Print JSON response
print(objects)

print(f'Number of objects found: {len(objects)}')
for obj in objects:
   print(f'{obj.name} (confidence: {obj.score})’)
   print('Normalized bounding polygon vertices: ‘)
   for vertex in obj.bounding_poly.normalized_vertices:
     print(f' - ({vertex.x}, {vertex.y})')
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

object_detection.py

#PUEAcademyDay19

```
Number of objects found: 5
Bird (confidence: 0.9233673810958862)
Duck (confidence: 0.8315161466598511)
Animal (confidence: 0.7112061977386475)
Wheel (confidence: 0.5616657733917236)
Toy (confidence: 0.5473584532737732)
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

text_detection.py

#PUEAcademyDay19

```
# Import the necessary packages
import io
import os
from google.cloud import vision
from google.cloud.vision import types
import argparse

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True,
                help = "path to where the image file resides")
args = vars(ap.parse_args())

# Authenticate API requests
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "credentials.json“

# The name of the image file to annotate
filename = os.path.join(os.path.dirname(__file__), args["image"])

# Loads the image into memory
with io.open(filename, 'rb') as file:
    content = file.read()
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

text_detection.py

#PUEAcademyDay19

```
# Instantiates a client
client = vision.ImageAnnotatorClient()
image = types.Image(content = content)

# Performs object localization on the image file
response = client.text_detection(image)

# Process the response
texts = response.text_annotations

# Print JSON response
print(texts)

print(f'Number of text fragments found: {len(texts)}’)
for text in texts:
     print("------------------------------")
     print(f'{text.description}’)
     print("------------------------------")
```

Hands-On Labs

Google Cloud Platform

python™

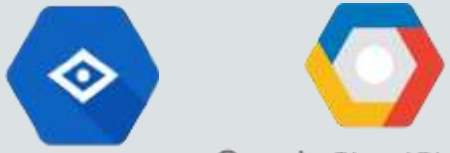
PUE ACADEMY Day

text_detection.py

#PUEAcademyDay19


```
Number of text fragments found: 3
----------------------------------------
3685 HDP
----------------------------------------
3685
----------------------------------------
HDP
----------------------------------------
```

Hands-On Labs



Google Cloud Platform

PUE ACADEMY Day

face_detection.py

#PUEAcademyDay19

```
# Import the necessary packages
import io
import os
from google.cloud import vision
from google.cloud.vision import types
import argparse

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True,
                help = "path to where the image file resides")
args = vars(ap.parse_args())

# Authenticate API requests
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "credentials.json“

# The name of the image file to annotate
filename = os.path.join(os.path.dirname(__file__), args["image"])

# Loads the image into memory
with io.open(filename, 'rb') as file:
    content = file.read()
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

face_detection.py

#PUEAcademyDay19

```
# Instantiates a client
client = vision.ImageAnnotatorClient()
image = types.Image(content = content)

# Performs object localization on the image file
response = client.face_detection(image)

# Process the response
faces = response.face_annotations

# Print JSON response
print(faces)

print(f'Number of faces found: {len(faces)}’)
for face in faces:
   print("-------------------------------------------------")
   print(f"Confidence: {face.detection_confidence}")
   print(f"Alegre: {face.joy_likelihood}")
   print(f"Triste: {face.sorrow_likelihood}")
   print(f"Enfadado: {face.anger_likelihood}")
   print(f"Sorprendido: {face.surprise_likelihood}")
   print(f"Borroso: {face.blurred_likelihood}")
   print(f"Con sombrero: {face.headwear_likelihood}")
   print("-------------------------------------------------")
```

Hands-On Labs

Google Cloud Platform

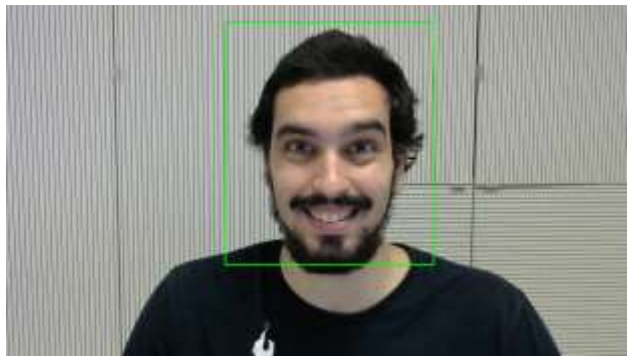
python™

PUE ACADEMY Day

face_detection.py

#PUEAcademyDay19

```
----------------------------------------
Confidence: 0.9987232685089111
Alegre: 1
Triste: 2
Enfadado: 1
Sorprendido: 1
Borroso: 1
Con sombrero: 1
----------------------------------------
```



```
----------------------------------------
Confidence: 0.9991716146469116
Alegre: 5
Triste: 1
Enfadado: 1
Sorprendido: 1
Borroso: 1
Con sombrero: 1
----------------------------------------
```

```
----------------------------------------
Confidence: 0.999671459197998
Alegre: 1
Triste: 1
Enfadado: 1
Sorprendido: 5
Borroso: 1
Con sombrero: 1
----------------------------------------
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

landmark_detection.py

#PUEAcademyDay19

```
# Import the necessary packages
import io
import os
from google.cloud import vision
from google.cloud.vision import types
import argparse

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True,
                help = "path to where the image file resides")
args = vars(ap.parse_args())

# Authenticate API requests
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "credentials.json“

# The name of the image file to annotate
filename = os.path.join(os.path.dirname(__file__), args["image"])

# Loads the image into memory
with io.open(filename, 'rb') as file:
    content = file.read()
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

landmark_detection.py

#PUEAcademyDay19

```
# Instantiates a client
client = vision.ImageAnnotatorClient()
image = types.Image(content = content)

# Performs object localization on the image file
response = client.landmark_detection(image)

# Process the response
landmarks  = response.landmark_annotations

# Print JSON response
print(landmarks)

print(f'Number of landmarks found: {len(landmarks)}’)
for landmark in landmarks:
   print("----------------------------------------------------")
   print(f“Description: {landmark.description}")
   print(f“Score: {landmark.score}")
   print("----------------------------------------------------")
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

landmark_detection.py

#PUEAcademyDay19


```
-------------------------------------------------
Description: Plaza de Cibeles
Score: 0.9911772012710571
-------------------------------------------------
```

Hands-On Labs

Google Cloud Platform

python™

PUE ACADEMY Day

landmark_detection.py

#PUEAcademyDay19


```
-------------------------------------------------
Description: Puerta del Sol, Post Office
Score: 0.5609591007232666
-------------------------------------------------
```

¡Muchas gracias!

jordi.arino@pue.es
@jordiAS2K

#PUEAcademyDay19

¿Alguna pregunta?

www.pue.es/cisco
pueacademy@pue.es

#PUEAcademyDay19


CISCO

PUE ACADEMY Day

www.pue.es

¡Gracias!

#PUEAcademyDay19

pueacademy@pue.es

BCN: 93 206 02 49

MAD: 91 162 06 69

pue

PROGRAMAS EDUCATIVOS

ORACLE ACADEMY · cloudera ACADEMIC PARTNER · Microsoft Imagine Academy · vmware IT ACADEMY · Linux Professional Institute · Networking Academy

PROGRAMAS DE CERTIFICACIÓN

EC-Council Associate · Certified Associate · AUTODESK Certification · Microsoft Office Specialist · Microsoft Technology Associate · unity Certification · pue mobile