

**Universidad Nacional Mayor de San Marcos**  
Universidad del Perú, Decana de América  
**FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**



**PROGRAMA:** ESPECIALIZACION EN PYTHON  
MÓDULO BÁSICO

# Clase 01

# Hello!

## Anthony Carrillo.

Bachiller y Titulado en la carrera de Ingeniería de Software por la UNMSM.

Maestría en Desarrollo de Aplicaciones y Servicios Web en la VIU.



A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines, with some nodes highlighted in grey and others in white.

1.

# Introducción, principios y filosofía

¿Por qué Python?

# Sobre la asignatura - Evaluación

## **EVALUACIÓN CONTINUA (Prácticas calificadas) -> 60%**

Durante el transcurso del presente módulo se tomarán dos prácticas las cuáles tienen un **30% c/u de PF del curso**.

## **EXAMEN FINAL-> 40%**

Examen final tiene un peso del **40% del PF del curso**.

## **EJERCICIOS ABORDADOS PARA EL ALUMNO-> Puntos de participación en clase**

Ejercicios de reforzamiento

# ¿Por qué python?

## ¿POR QUÉ USAR PYTHON?

Diseñado para ser claro, con lógica fácil de leer.

Python es un lenguaje de programación multiplataforma.

Existen miles de librerías escritas en Python que nos permiten crear aplicaciones complejas en vez de tener que esforzarnos a volver a inventar la “rueda”.

## OPTIMIZAR TIEMPO DE DESARROLLO

Python tiene como foco en “acortar” el tiempo de desarrollo.

## DOCUMENTACIÓN

<https://docs.python.org/3/>

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines, with some nodes highlighted in grey and others in white.

2.

# Instalación de Python e implementaciones

# Instalación

<https://www.python.org/downloads/>



python.org/downloads/

Python PSF Docs PyPI Jobs Community

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events


### Download the latest version for Windows

[Download Python 3.10.4](#)

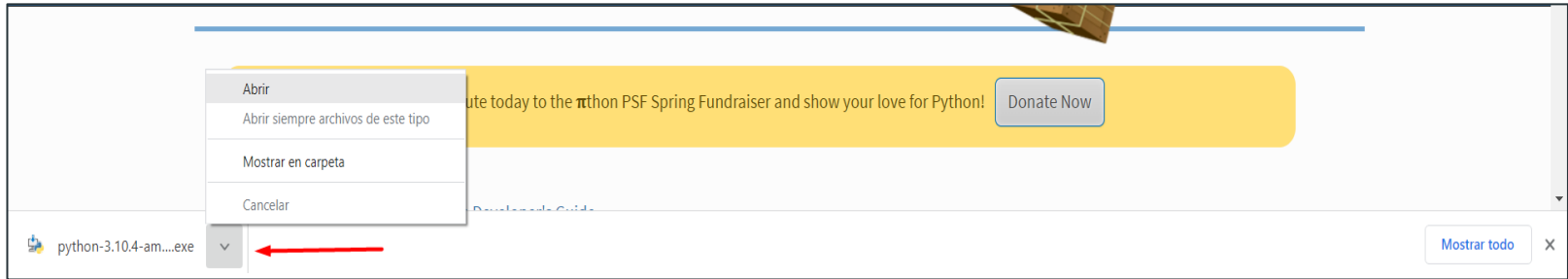
Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)

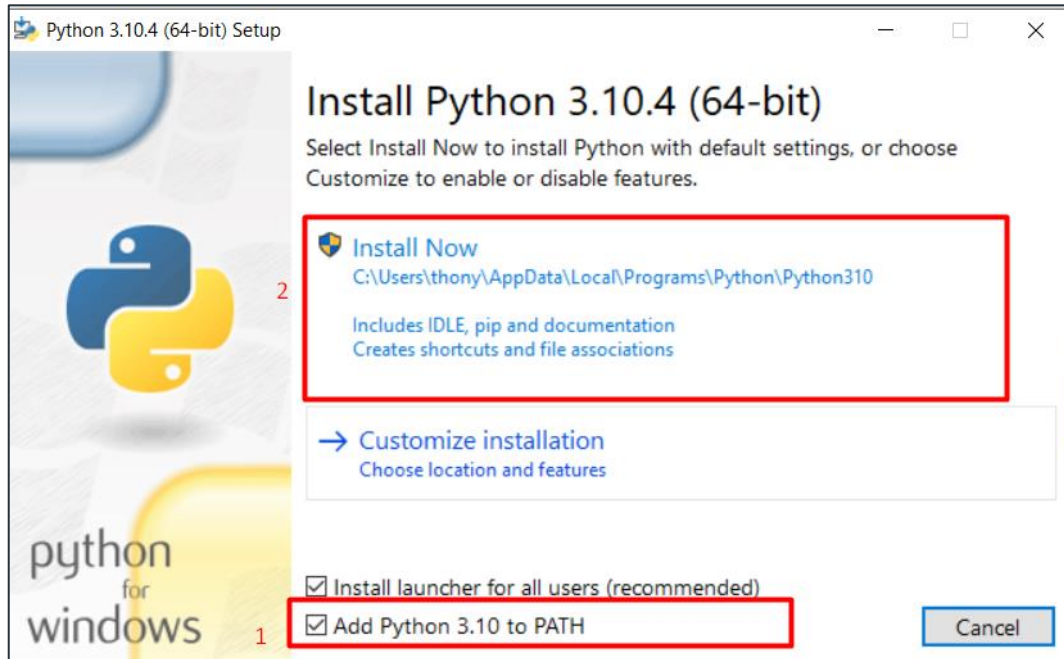
Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

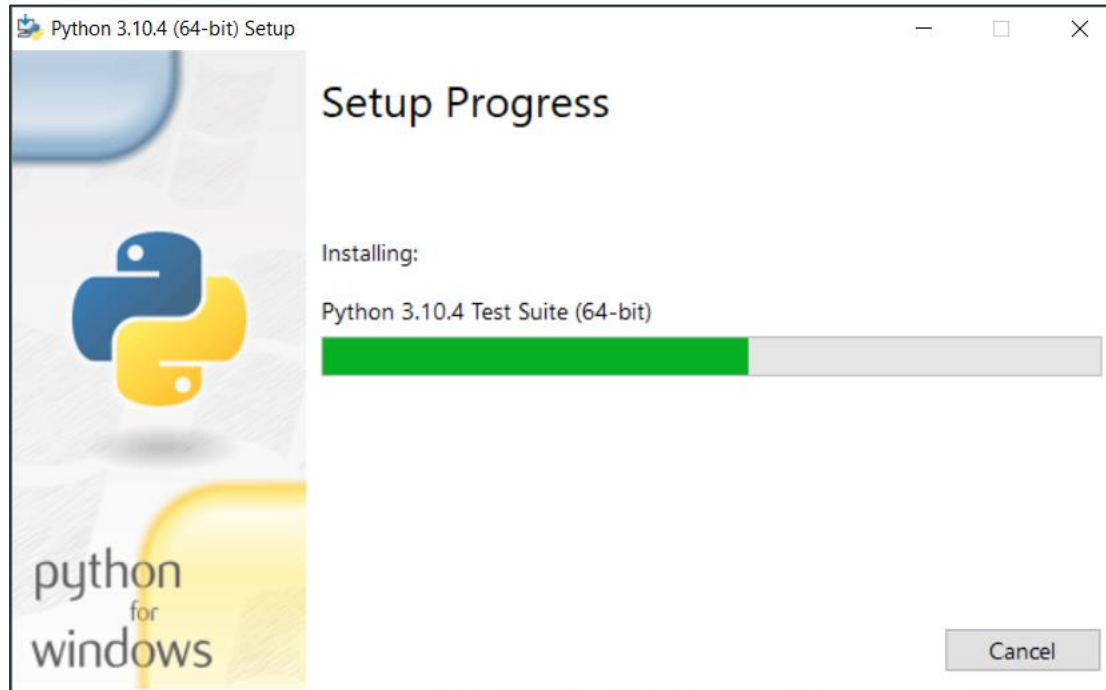
Looking for Python 2.7? See below for specific releases

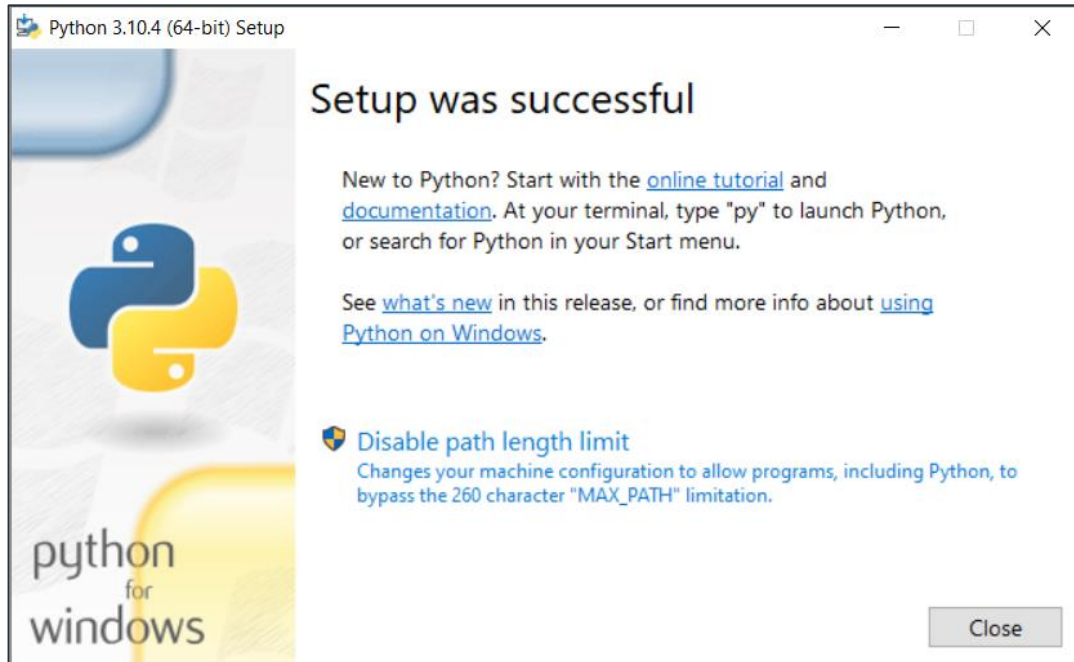






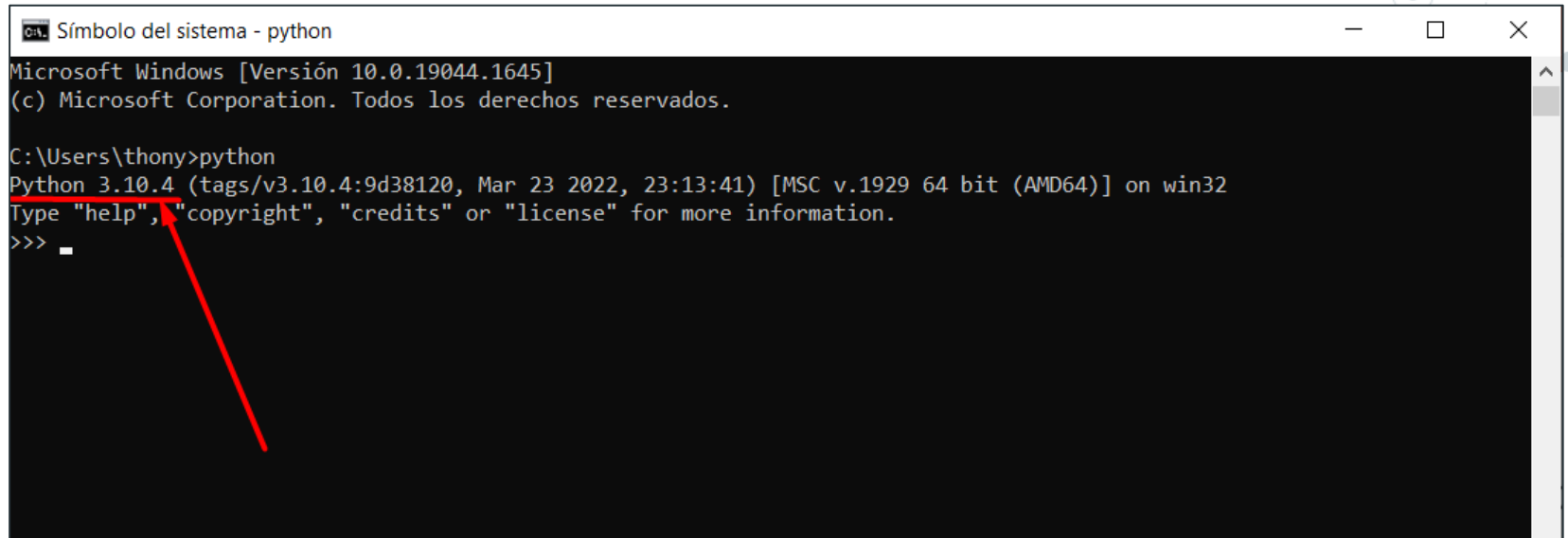






## Comprobar instalación de python:

1. Abrimos una terminal del Sistema Operativo (win + R)
2. Escribimos cmd y damos enter.
3. En la terminal del Sistema Operativo escribimos “python” y damos enter.
4. Nos muestra la version de python que hemos instalado



```
Símbolo del sistema - python
Microsoft Windows [Versión 10.0.19044.1645]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\thony>python
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```


## IDE en el que trabajaremos:



1. Trabajemos con el IDE Pycharm de JetBrains.
2. Abrimos el enlace y desargamos el IDE en: <https://www.jetbrains.com/es-es/pycharm/>
3. **Crearse una cuenta** con su correo educativo, ya que este nos proveerá de una licencia gratuita para acceder a la version **Professional** de **Pycharm**.

Al finalizar la activación de la licencia en Jetbrains se verá del siguiente modo en tu cuenta:

### 1 Educational Pack License

**JetBrains Educational Pack**  
[Download ▾](#)

Licensed to: **Anthony Joffre Carrillo Rosales**


License restriction: For educational use only

Valid through: October 28, 2025

Following products included:






 **JETBRAINS**

Para desarrolladores Para equipos Educación Soluciones Asistencia Tienda 🔍 🧑 🛒 🚫

PyCharm JetBrains IDEs Casos de uso ▼ EAP Novedades Funcionalidades ▼ Aprender ▼ Tarifas [Descargar](#)

[Windows](#) [macOS](#) [Linux](#)

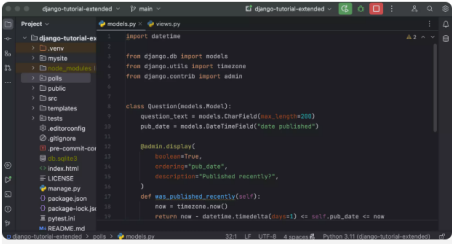


# PyCharm Professional

El IDE de Python para la ciencia de datos y el desarrollo web

[Descargar](#) [.exe \(Windows\) ▼](#)

Prueba gratis de 30 días

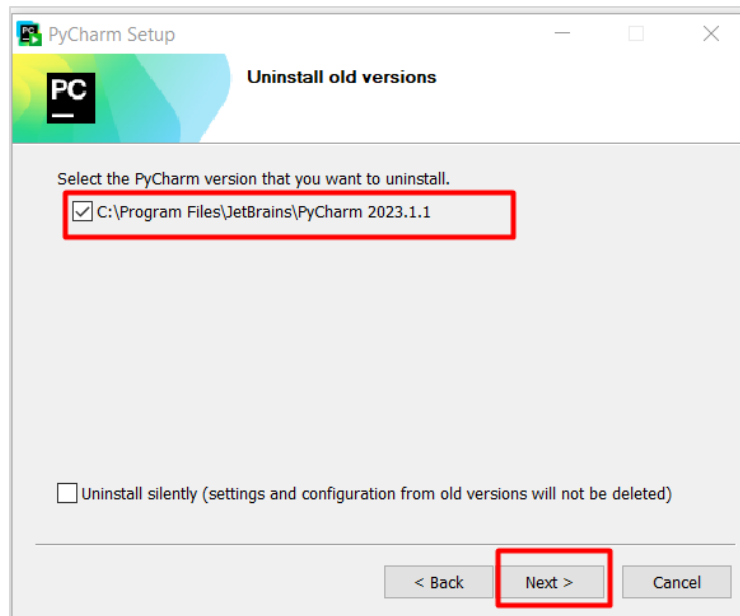


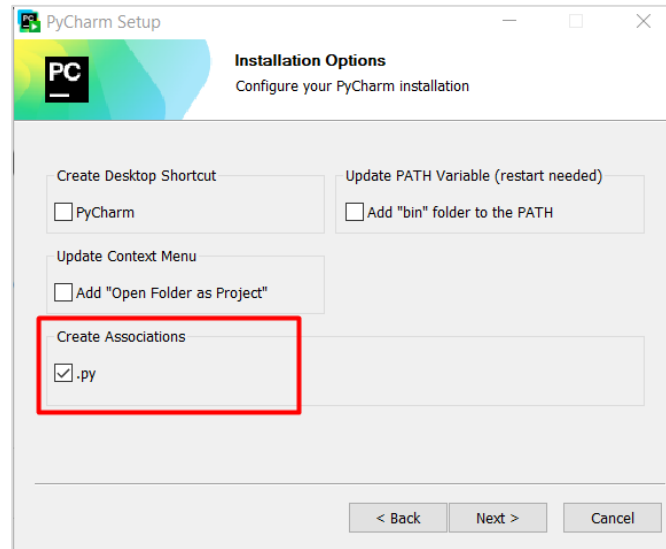
Versión: 2024.3.3  
Build: 243.24978.54  
12 de febrero de 2025

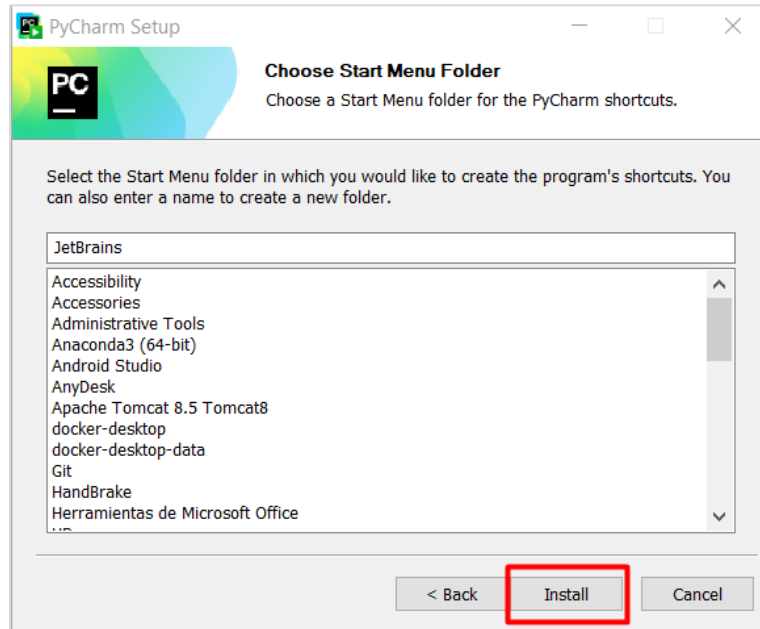
[Requisitos del sistema](#)  
[Instrucciones de instalación](#)

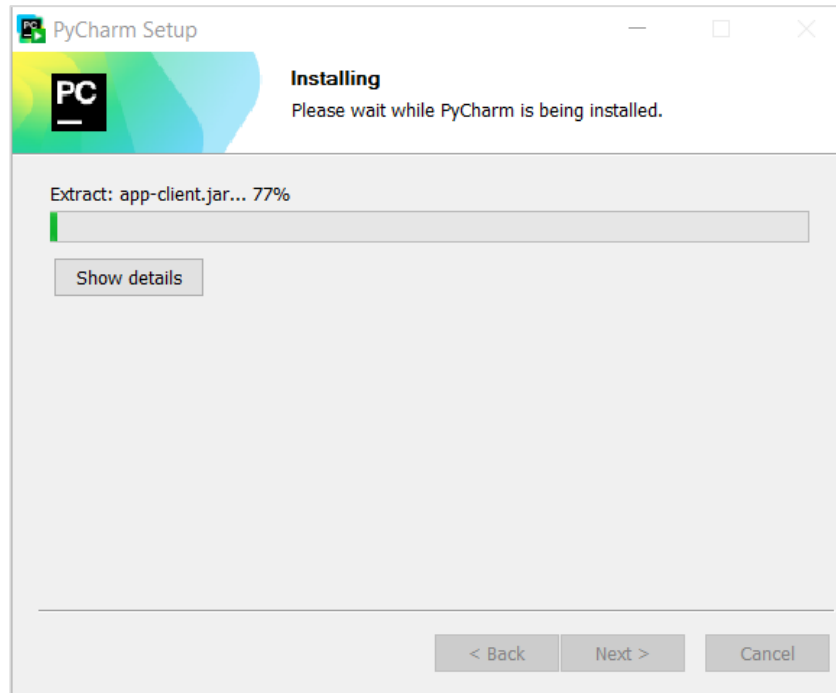
[Otras versiones](#)  
[Software de terceros](#)

[Feedback](#)

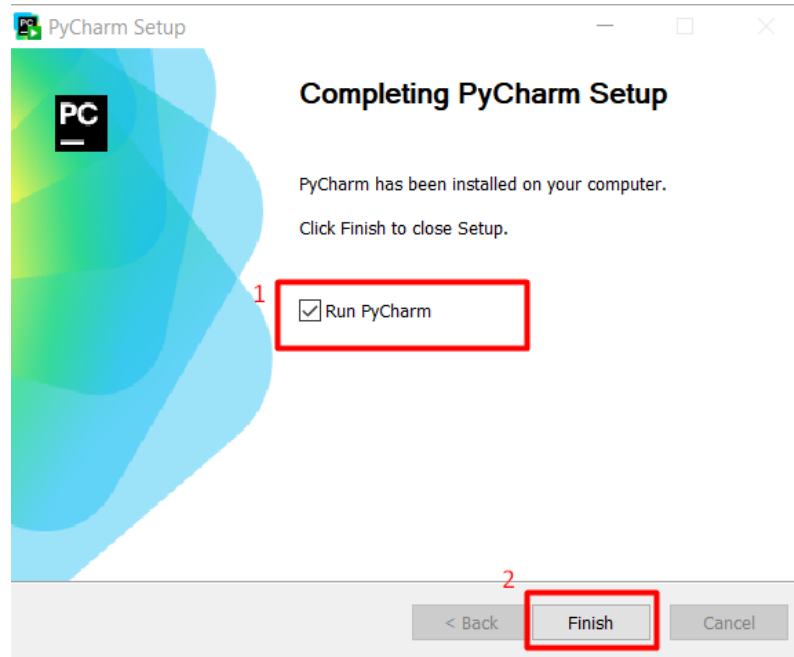




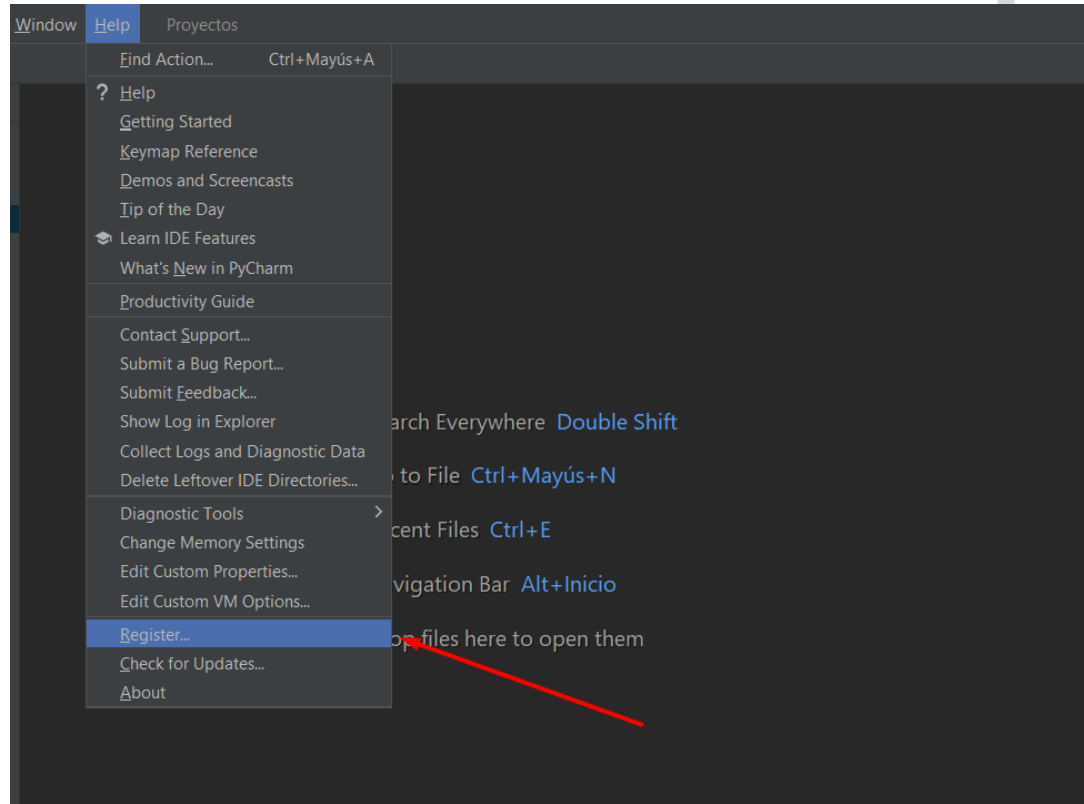




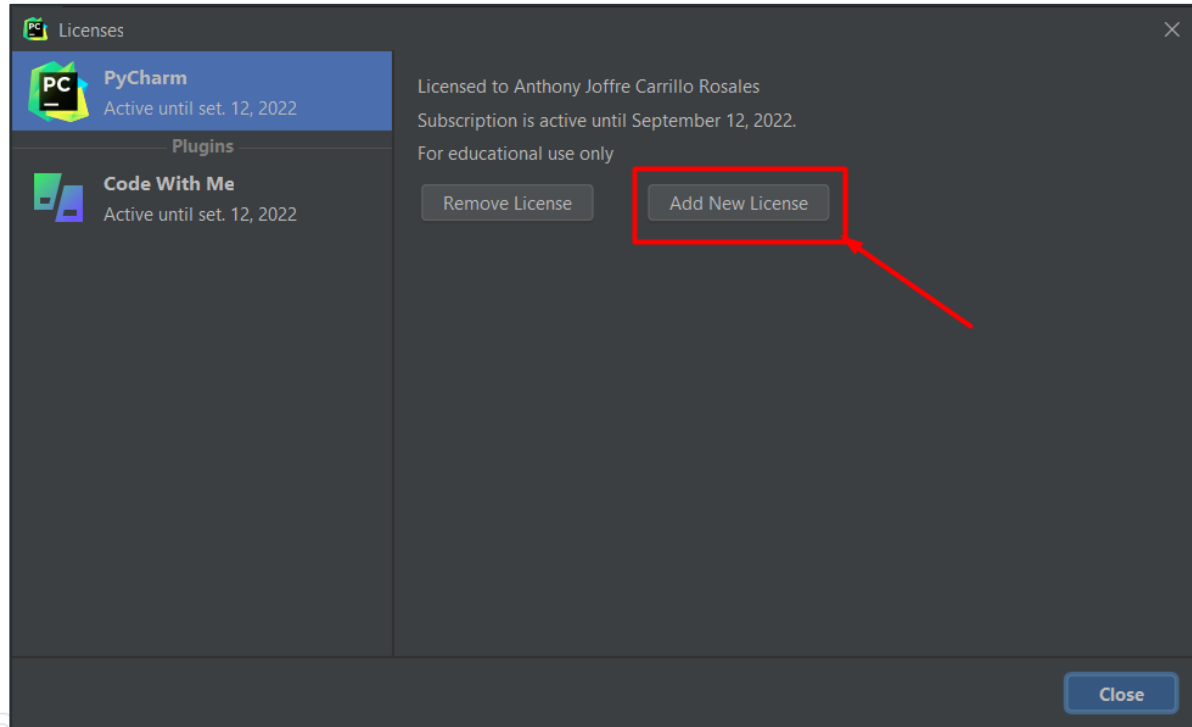
✓ Al ver esta pantalla Pycharm ya ha sido instalado correctamente

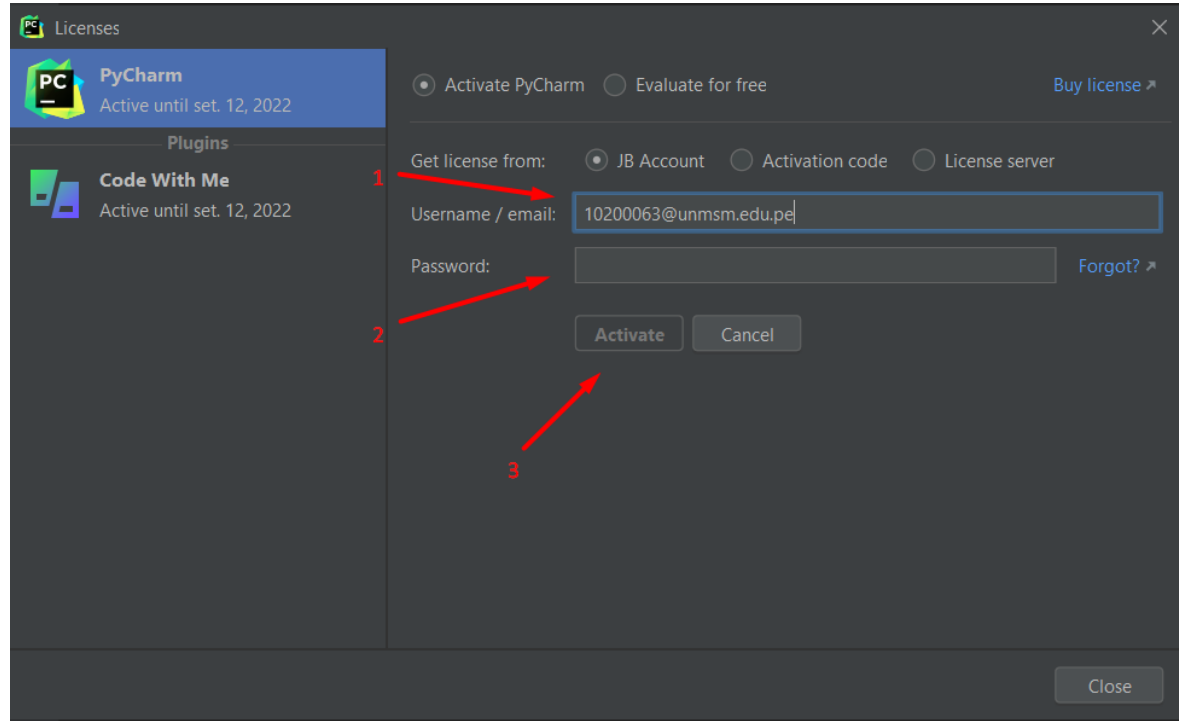




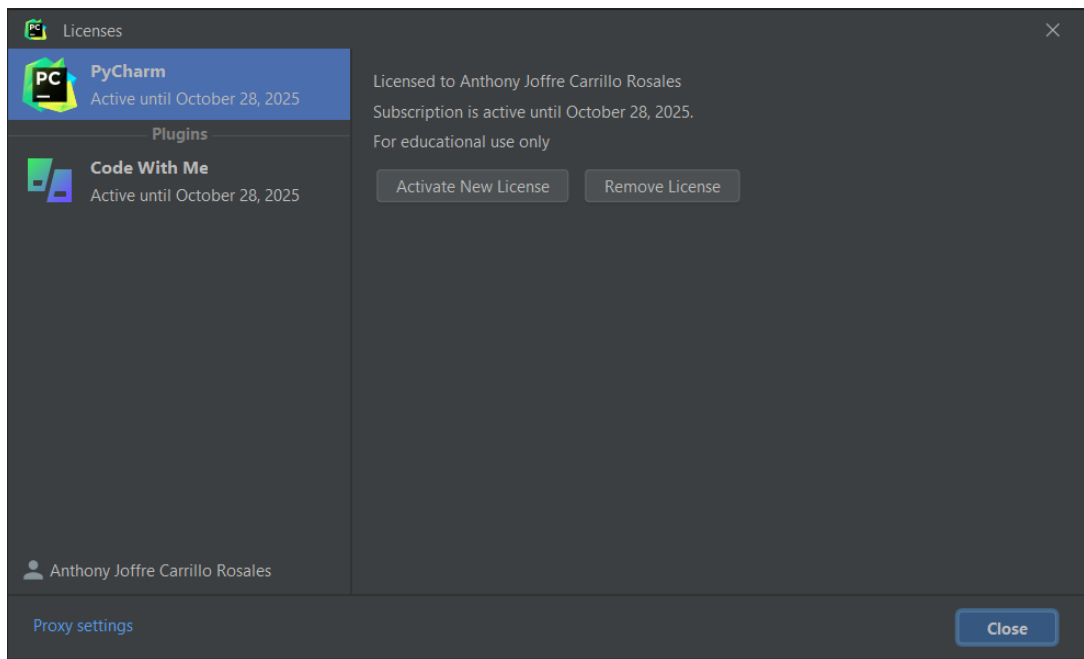








- ✓ Finalmente se verá la siguiente pantalla pero son su nombre y cuando la licencia ya fue vinculada correctamente a Pycharm



# Implementación

Priorización de versiones en nuestro S.O.



## Priorización:

Se habla de priorización de instaladores cuando se tiene ambas implementaciones de 32 y 64 bits de la misma versión de python instalas en el Sistema Operativo.

Reglas:

1. Se priorizará la instalación de Python de 64 bits.
2. No importará el orden el cuál fueron instalados.
3. Si tenemos la version 3 y 3.1 para los commando python y python3 utilizarán ambos por detrás la version 3.1. específicamente

# Comandos de instalación de dependencias:

Para el uso e instalación de dependencias o librerías tenemos las siguientes líneas de comando.

**1. python**

Nos muestra la versión de python instalada.

**2. pip install “nombreLibreria”**

Instala la última versión de la librería.

**3. pip install “nombreLibreria”==NroVersionExactadeLibreria**

Instala la versión indicada de la librería en nuestro entorno virtual.

**4. pip uninstall “nombreLibreria”**

Desinstala la librería instalada en nuestro entorno virtual.

**5. pip freeze**

Nos muestra la lista de todas las dependencias que hemos instalado.

## Comandos de instalación de dependencias:

### 6. **pip install -r nombreArchivo.txt**

Nos instala todas las dependencias que se mencionan dentro del archivo .txt con una sola línea de comando y ya no instalarlas de una en una.

Para los casos que un proyecto o nuestro proyecto necesite de varias librerías o dependencias.

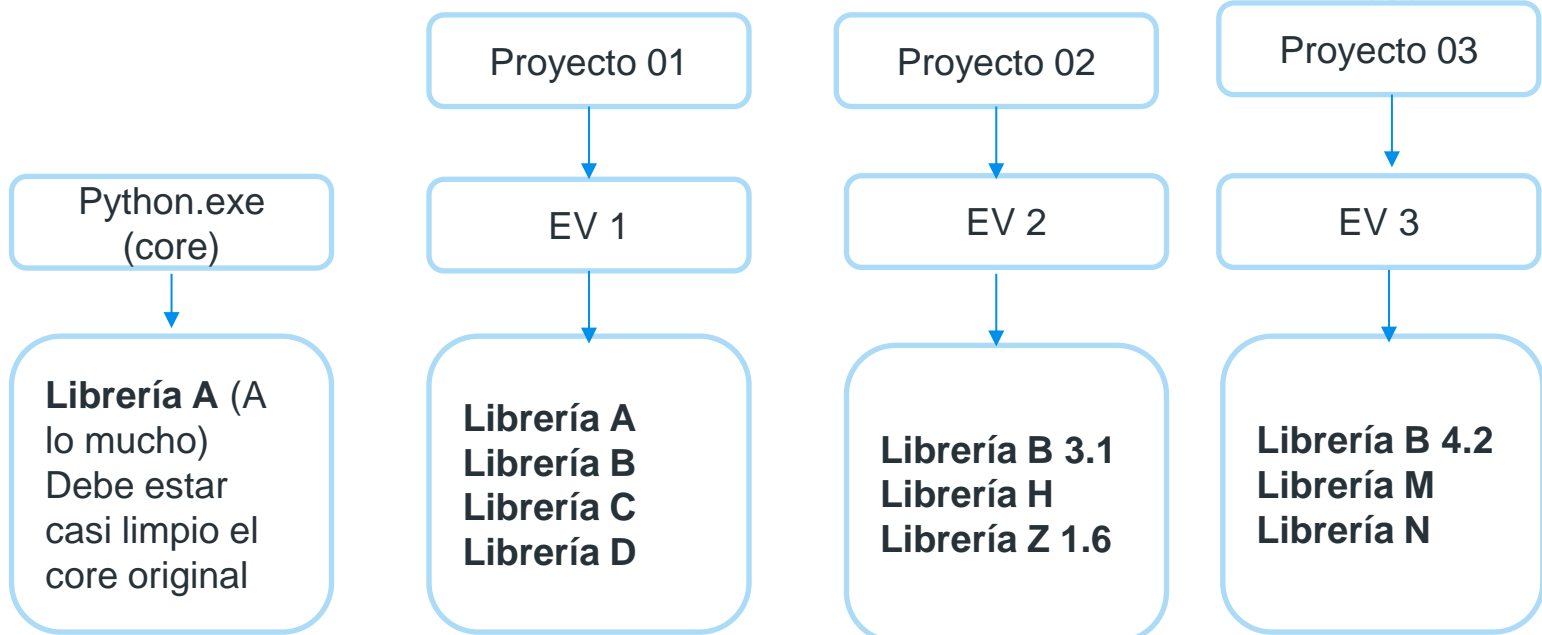
### 7. **pip freeze > ./requirements.txt**

Obtenemos la lista de las librerías y sus respectivas versiones instaladas en el entorno virtual con el nombre de **requirements.txt**

### 8. **pip install -r requirements.txt**

Se instalará todos las librerías que se mencionan en el archivo **requirements.txt** en tu respectivo **virtualenv**

## Entornos virtuales (virtualenv):







3.

# Tipos de datos en Python

# Tipos básicos de datos



Enteros, flotantes, cadena de caracteres.

# Tipo básico de datos:

## 1. Tipo numéricos

Python define tres tipos de datos numéricos básicos: enteros, números de punto flotante y complejos.

a) **Número enteros:** Es el tipo de los número enteros *int*

```
1. >>> a = -1 # a es de tipo int y su valor es -1
2. >>> b = a + 2 # b es de tipo int y su valor es 1
3. >>> print(b)
4. 1
```

b) **Número de punto flotante:** Es el tipo de dato que representa números reales o sea de tipo *float*.

```
1. >>> real = 1.1 + 2.2 # real es un float
2. >>> print(real)
3. 3.3000000000000003 # Representación aproximada de 3.3
4. >>> print(f'{real:.2f}')
5. 3.30 # real mostrando únicamente 2 cifras decimales
```

- c) Número complejo: Es el tipo de los números complejos. Y se representan con un *float*

```
1. >>> complejo = 1+2j
2. >>> complejo.real
3. 1.0
4. >>> complejo.imag
5. 2.0
```

- d) *Aritmética de los números complejos: Python permite la operación de números de distintos tipo.*

```
1. >>> 1 + 2.0
2. 3.0
3. >>> 2+3j + 5.7
4. (7.7+3j)
```

## 2. Tipo booleano

En Python la clase que representa los booleanos es *bool*. Valor *True* para representar verdadero y *False* para representar falso.

Los siguientes objetos son consideradas falsas:

- ✓ `None`
- ✓ `False`
- ✓ El valor cero de cualquier tipo numérico: `0`, `0.0`, `0j`, ...
- ✓ Secuencias y colecciones vacías (veremos estos tipos más adelante): `''`, `()`, `[]`, `{}`, `set()`, `range(0)`

### 3. Tipo de cadena de caracteres:

Otro tipo básico de dato en Python, e imprescindible, son las secuencias o cadenas de caracteres. Este tipo es conocido como ***string***, ***str***.

```
1. >>> hola = 'Hola "Pythonista"'
2. >>> hola_2 = 'Hola \'Pythonista\''
3. >>> hola_3 = "Hola 'Pythonista'"
4.
5. >>> print(hola)
6. Hola "Pythonista"
7.
8. >>> print(hola_2)
9. Hola 'Pythonista'
10.
11. >>> print(hola_3)
12. Hola 'Pythonista'
```

# Tipos fundamentales de datos

list, tuple, set y dict



## Otro tipos

Y no menos importantes... Ya se vieron los tipos de datos básicos, a continuación veremos los más trascendentes dentro del uso del lenguaje.


**Las listas** son secuencias mutables de valores.

**Las tuplas** son secuencias inmutables de valores.

**Los conjuntos** se utilizan para representar conjuntos únicos de elementos, es decir, en un conjunto no pueden existir dos objetos o datos iguales.

**Los diccionarios** son tipos especiales de contenedores en los que se puede acceder a sus elementos a partir de una clave (**llave o key**) única.





```
1. >>> lista = [1, 2, 3, 8, 9]
2. >>> tupla = (1, 4, 8, 0, 5)
3. >>> conjunto = set([1, 3, 1, 4])
4. >>> diccionario = {'a': 1, 'b': 3, 'z': 8}
5. >>> print(lista)
6. [1, 2, 3, 8, 9]
7. >>> print(tupla)
8. (1, 4, 8, 0, 5)
9. >>> print(conjunto)
10. {1, 3, 4}
11. >>> print(diccionario)
12. {'a': 1, 'b': 3, 'z': 8}
```



## Conocer el tipo de variable:

```
type(3)  
class 'int'
```

## Conversión de tipos:

**str():** Devuelve la representación en cadena de caracteres del objeto que se pasa como parámetro.

**int():** Devuelve un *int*

**float():** Devuelve un *float*



A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by circles of varying sizes, some with concentric rings, and the lines are thin and grey. The diagram is partially cut off by the top and left edges of the slide.

3.

# Operaciones comunes

## Operadores aritméticos:

**suma: +**

**resta: -**

**multiplicación: \***

**división: /**

**módulo: %**

**potencia: \*\***

**división con resultado entero: //**

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting different levels of connectivity or importance. The lines are thin and gray, creating a mesh-like structure.

4.

# Estructura de datos

## Listas ([])

*list.append(x): Agrega un ítem al final de la lista*

*list.pop([i]): Quitar el ítem en la posición dada. También posible sin indicar la posición.*

*list.insert(i, x): Inserta un ítem en una posición indicada*

*list.remove(x): Quita el valor de la lista con valor x.*

*list.clear(): Elimina los elementos de la lista.*

*list.count(x): Devuelve la cantidad de veces que aparece x en una lista.*

*list.reverse(x): Invierte todos los elementos de la lista.*

*list.sort(): ordena los elementos de la lista.*

*list.copy(): Devuelve una copia de la lista.*

*del list[i]: Elimina un ítem de la lista enviando una posición existente.*

## Diccionarios.

Un diccionario es una estructura de datos que se almacenan en pares o sea en **claves(key)/valores(value)**. Los diccionarios son mutables. Aceptan cambios después de crearlos. Todas las claves/valores deben ser únicos.

Una vez que creaste un diccionario, puedes acceder a los elementos en él utilizando la clave(**key**) para cada valor.

```
var = {}      // Diccionario vacío
```

```
var[i]        // Obtienes el valor de un key que exista en el diccionario
```

```
var['key'] = tuValor    // Agrega un valor y un key nuevo al diccionario
```

```
del var['key'] // Elimina la llave y el valor del diccionario
```

```
sorted(var)    // Verifica un valor en lista.
```

```
list(var)      // Convierte tu diccionario en una lista
```

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines, with some nodes highlighted in grey and others in white.

5.

# Diferencias entre Python 2.x y 3.x



Comparamos	Python 2	Python 3
Fecha de lanzamiento	2000	2008
Función print	print "Hello Pythonista"	print("Hello Pythonista")
División de enteros	Cuando se dividen 2 enteros, siempre proporciona un valor entero.	Cuando se dividen 2 enteros, obtienes un valor flotante.
Sintaxis	Relativamente difícil de entender	Con Python 3 se han simplificado reglas y el código se volvió completamente intuitivo.
Iteración	El xrange() se usa para iteraciones	Su nueva función Range() se usa para realizar iteraciones
Compatibilidad con versiones anteriores.	Python 3 no es retrocompatible con Python 2.	No es difícil portar Python 2 a Python 3, pero no necesariamente confiable.
Biblioteca	Muchas bibliotecas antiguas creadas para Python 2 no son compatibles con versiones anteriores	Muchos developers están creando bibliotecas que solo puede usar Python 3.



# Universidad Nacional Mayor de San Marcos