

# INTRODUCCIÓN A PYTHON DESDE EL TERMINAL



# PYTHON

# ÍNDICE

[\*\*Breve Historia de Python\*\*](#)

[\*\*Instalación de Python\*\*](#)

[\*\*Editor de Texto Plano Notepad++\*\*](#)

[\*\*Configuración de Terminal CMD\*\*](#)

[\*\*Ejemplos de Aplicación\*\*](#)

[\*\*Videos Complementarios\*\*](#)

[\*\*Libros Recomendados\*\*](#)

[\*\*Referencias\*\*](#)

## Breve Historia de Python

Fue desarrollado inicialmente a fines de la década de 1980 por el programador holandés **Guido van Rossum**, y su primera versión fue lanzada en 1991 en Centrum Wiskunde & Informatica - CWI (Instituto Nacional de Investigación en Matemáticas y Ciencias de la Computación), en los Países Bajos.

La primera versión pública de Python, la versión 0.9.0, fue lanzada en febrero de 1991, luego le seguiría Python 1.0, fue lanzado en enero de 1994, Python 2.0, fue lanzada en octubre de 2000, **Python 3.0**, fue lanzada en Diciembre de 2008 y es la versión que se utiliza en la actualidad, destacándose en Inteligencia Artificial, Análisis de datos y programación en general, recomendándose su uso como primer lenguaje de programación.



# Instalación de Python

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 3.12.4

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

Want to help test development versions of Python 3.13? [Prereleases](#), [Docker images](#)



The 2024 PSF Board Election is Open: If you are a voting member of the PSF that affirmed your intention to participate in this year's election, we need your  
vote! More Info

Active Python Releases

For more information visit the Python Developer's Guide.

(Python Software Foundation, 2024)



Donate



Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

Python » Downloads » Windows

## Python Releases for Windows

- [Latest Python 3 Release - Python 3.12.4](#)

### Stable Releases

- [Python 3.12.4 - June 6, 2024](#)

**Note that Python 3.12.4 cannot be used on Windows 7 or earlier.**

- [Download Windows installer \(64-bit\)](#)
- [Download Windows installer \(ARM64\)](#)
- [Download Windows embeddable package \(64-bit\)](#)
- [Download Windows embeddable package \(32-bit\)](#)
- [Download Windows embeddable package \(ARM64\)](#)
- [Download Windows installer \(32-bit\)](#)
- [Python 3.12.3 - April 9, 2024](#)

### Pre-releases

- [Python 3.13.0b3 - June 27, 2024](#)
  - [Download Windows installer \(64-bit\)](#)
  - [Download Windows installer \(ARM64\)](#)
  - [Download Windows embeddable package \(64-bit\)](#)
  - [Download Windows embeddable package \(32-bit\)](#)
  - [Download Windows embeddable package \(ARM64\)](#)
  - [Download Windows installer \(32-bit\)](#)
- [Python 3.13.0b2 - June 5, 2024](#)
  - [Download Windows installer \(64-bit\)](#)
  - [Download Windows installer \(ARM64\)](#)

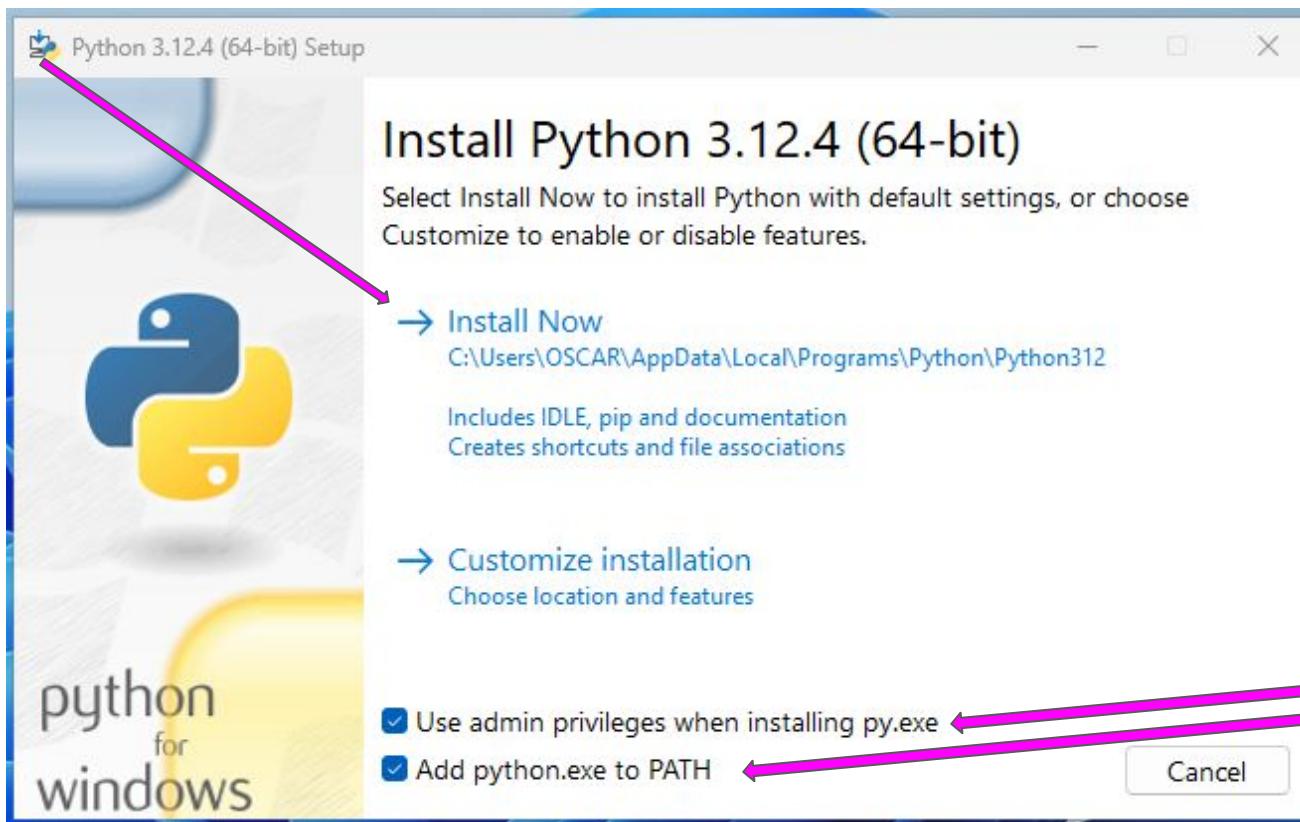
# Instalación de Python

 python-3.12.4-amd64

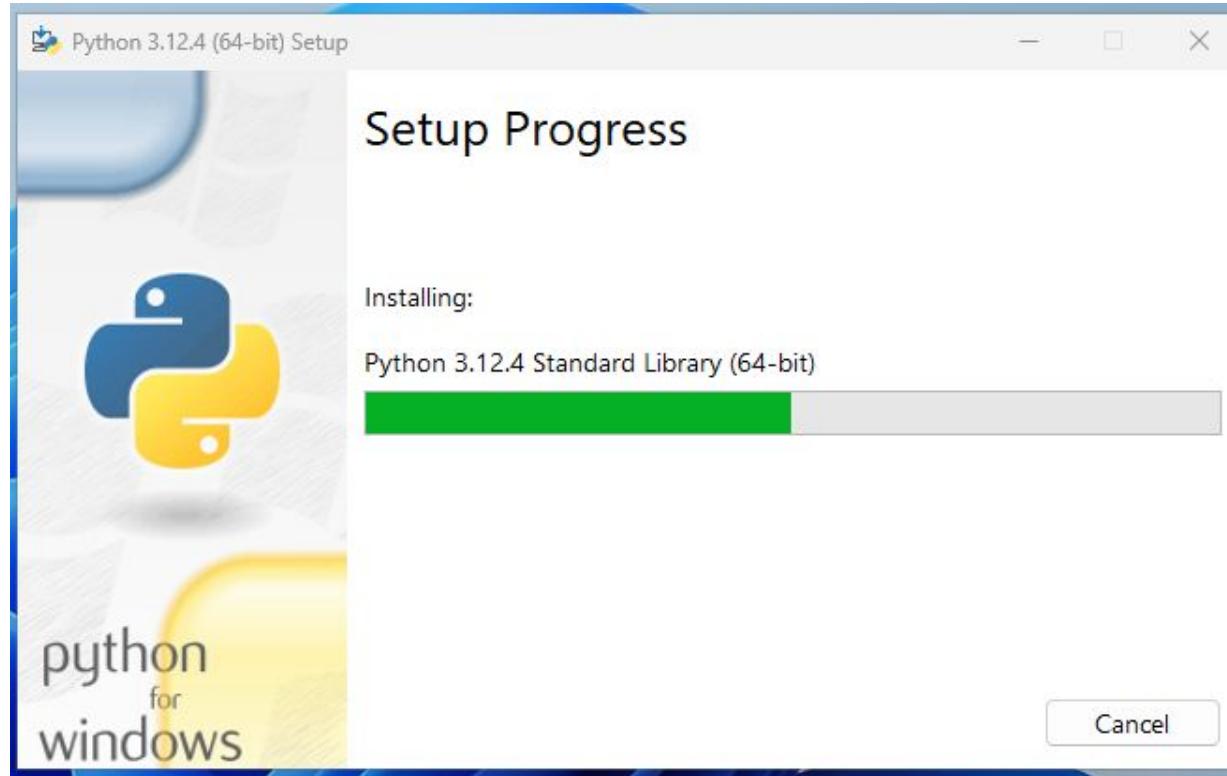
17.07.2024 13:02

Aplicación

26 145 KB



# Instalación de Python



# Editor de Texto Plano: NOTEPAD++



[Current Version 8.6.9](#)

- [Home](#)
- [Download](#)
- [News](#)
- [Online Help](#)
- [Resources](#)
- [RSS](#)
- [Donate](#)
- [Author](#)



Design Pickle - See how over 1,400 agencies trust Design Pickle to deliver their client's creative.  
AUS VIA CARTON

## What is Notepad++

Notepad++ is a free (as in "free speech" and also as in "free beer") source code editor and Notepad replacement that supports several languages. Running in the MS Windows environment, its use is governed by [GNU General Public License](#).

Based on the powerful editing component [Scintilla](#), Notepad++ is written in C++ and uses pure Win32 API and STL which ensures a higher execution speed and smaller program size. By optimizing as many routines as possible without losing user friendliness, Notepad++ is trying to reduce the world carbon dioxide emissions. When using less CPU power, the PC can throttle down and reduce power consumption, resulting in a greener environment.

```
*C:\sources\notepad4ever.cpp - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X  
new 1  
NotePad_plus.cpp notePad4ever.cpp new 1  
1 #include <GPL>  
2 #include <free_software>  
3  
4 void NotePad4ever()  
5 {  
6     while (true)  
7     {  
8         NotePad++ ;  
9     }  
10 }  
length: 108 line:Ln : 8 Col : 21 Pos : 102 Windows (CR LF) UTF-8 INS .d
```

(Don Ho, s.f.)

# Editor de Texto Plano: NOTEPAD++



Current Version 8.6.9

- Home
- Download
- News
- Online Help
- Resources
- RSS
- Donate
- Author

## Notepad++ v8.6.9: Support Taiwan's Independence

Release Date: 2024-07-14

### Download Notepad++ x64



- [Installer | GPG Signature](#)
- [Portable \(zip\) | GPG Signature](#)
- [Portable \(7z\) | GPG Signature](#)
- [Mini-portable \(7z\) | GPG Signature](#)

### Download 32-bit x86

- [Installer | GPG Signature](#)
- [Portable \(zip\) | GPG Signature](#)
- [Portable \(7z\) | GPG Signature](#)



(Don Ho, s.f.)

# Editor de Texto Plano: NOTEPAD++

 npp.8.6.9.Installer.x64

17.07.2024 13:11

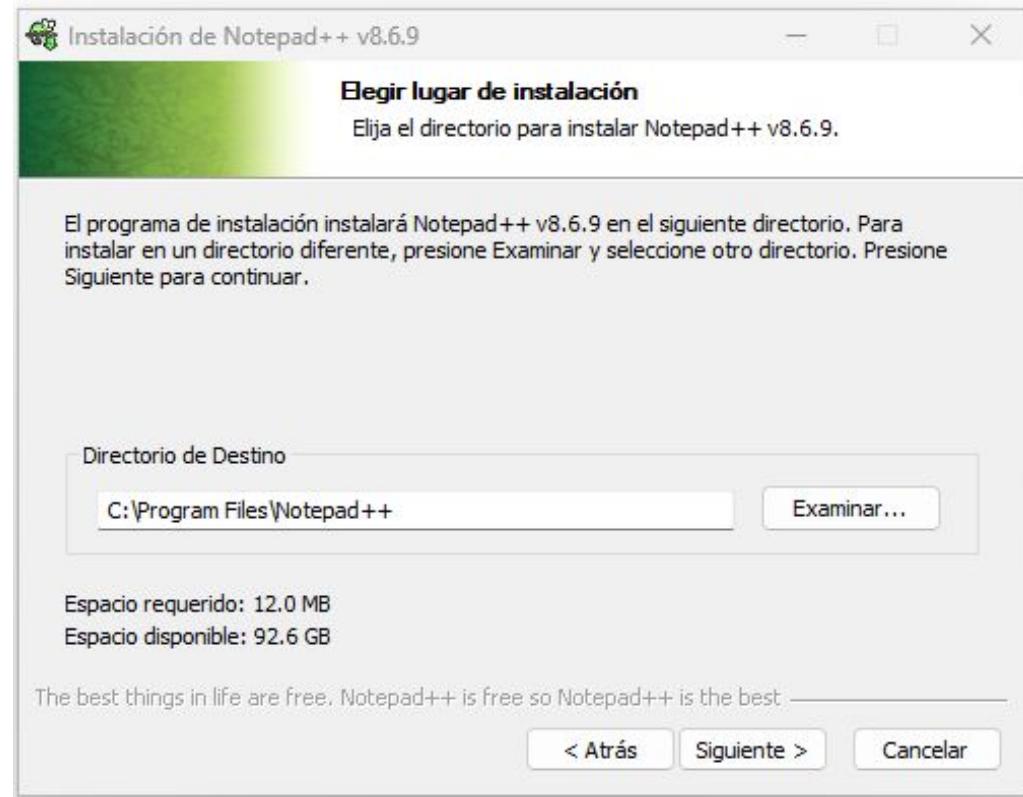
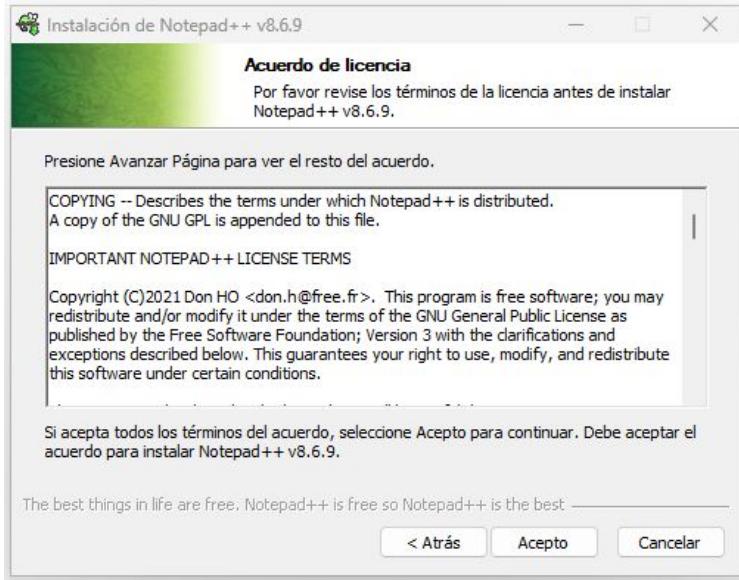
Aplicación

6 388 KB



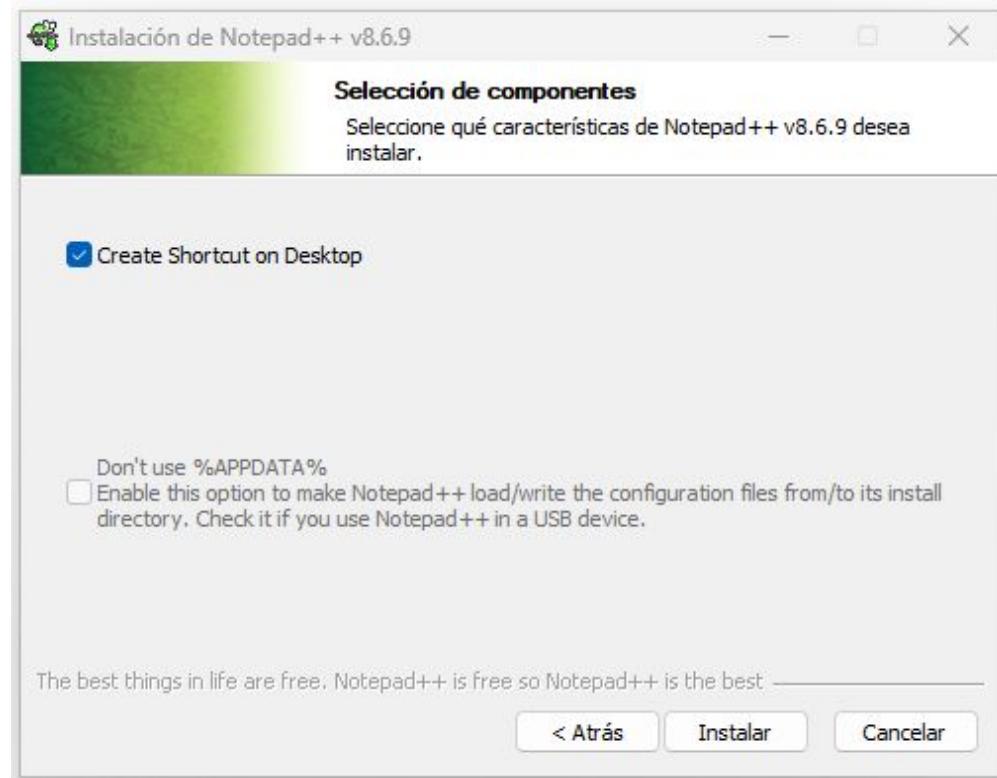
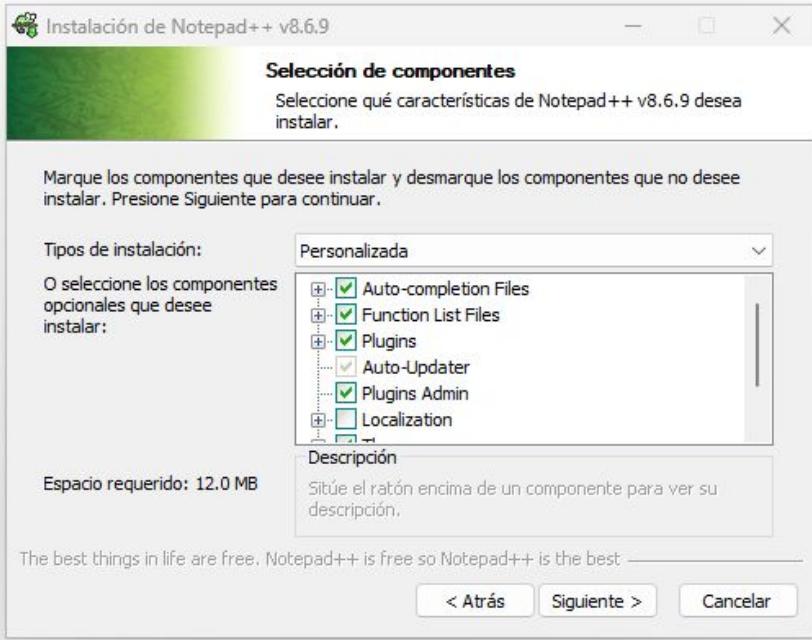
(Don Ho, s.f.)

# Editor de Texto Plano: NOTEPAD++



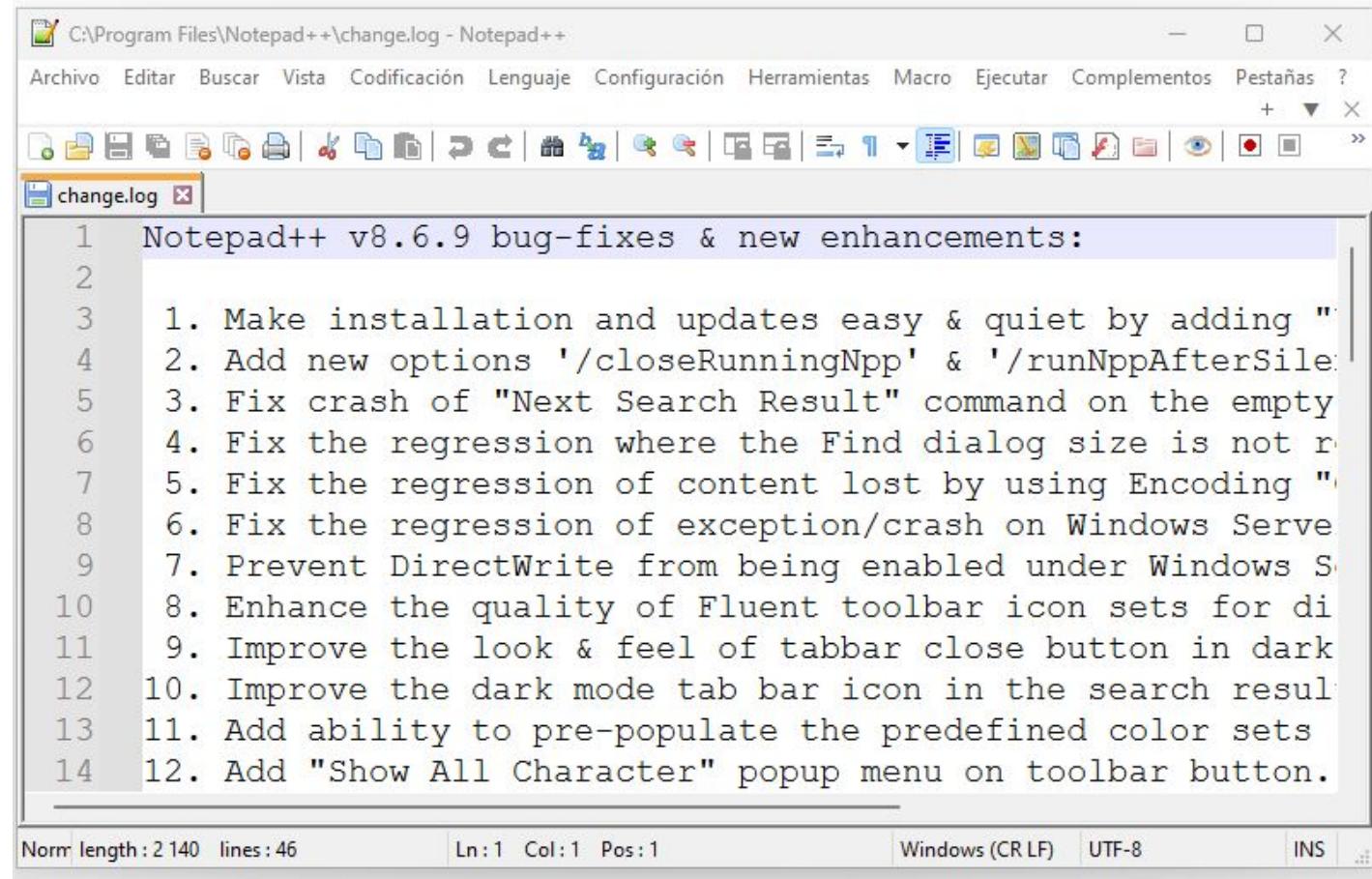
(Don Ho, s.f.)

# Editor de Texto Plano: NOTEPAD++



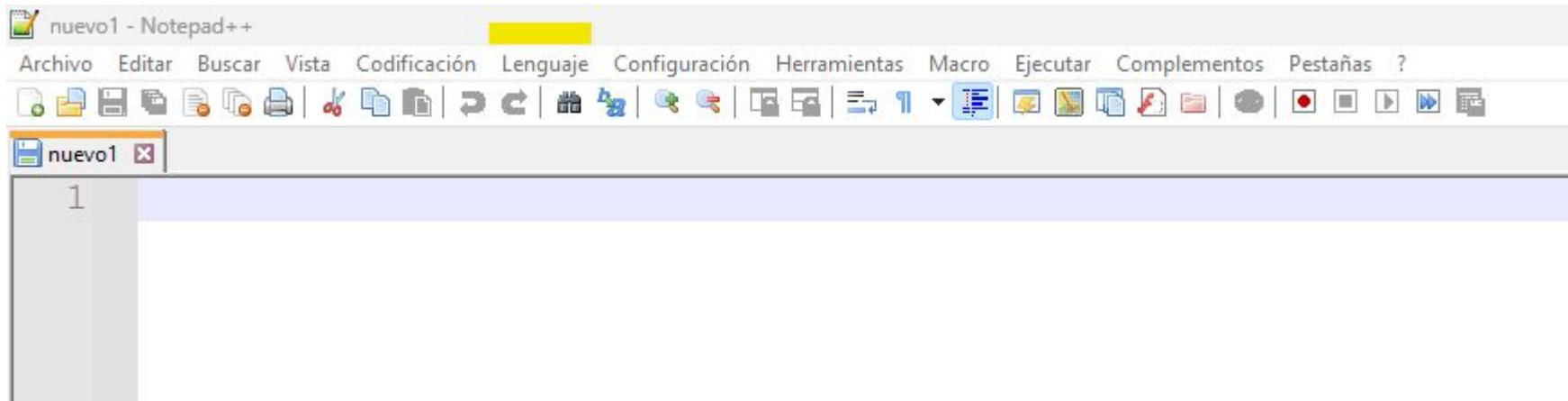
(Don Ho, s.f.)

## Editor de Texto Plano: NOTEPAD++

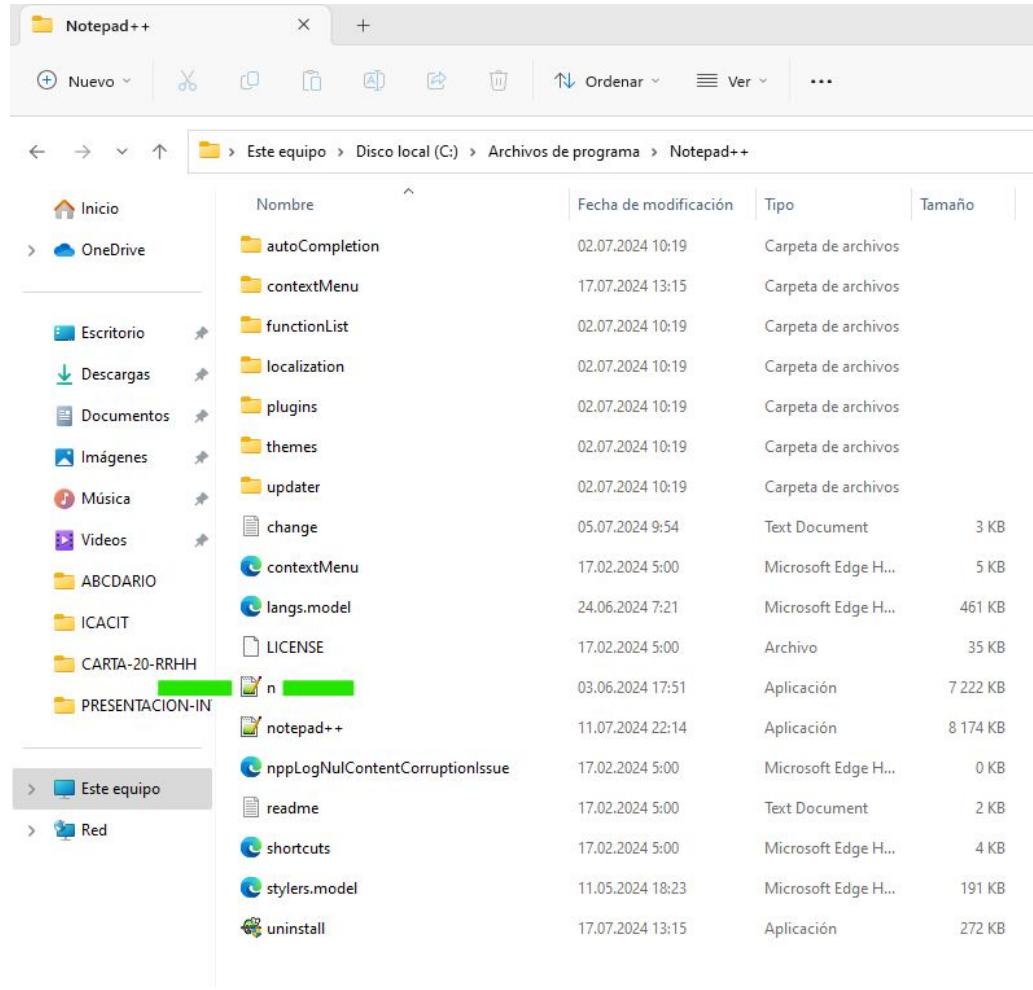


## Editor de Texto Plano: **NOTEPAD++**

Seleccionar **Lenguaje** y allí ubicar la letra **P** y seleccionar **Python**

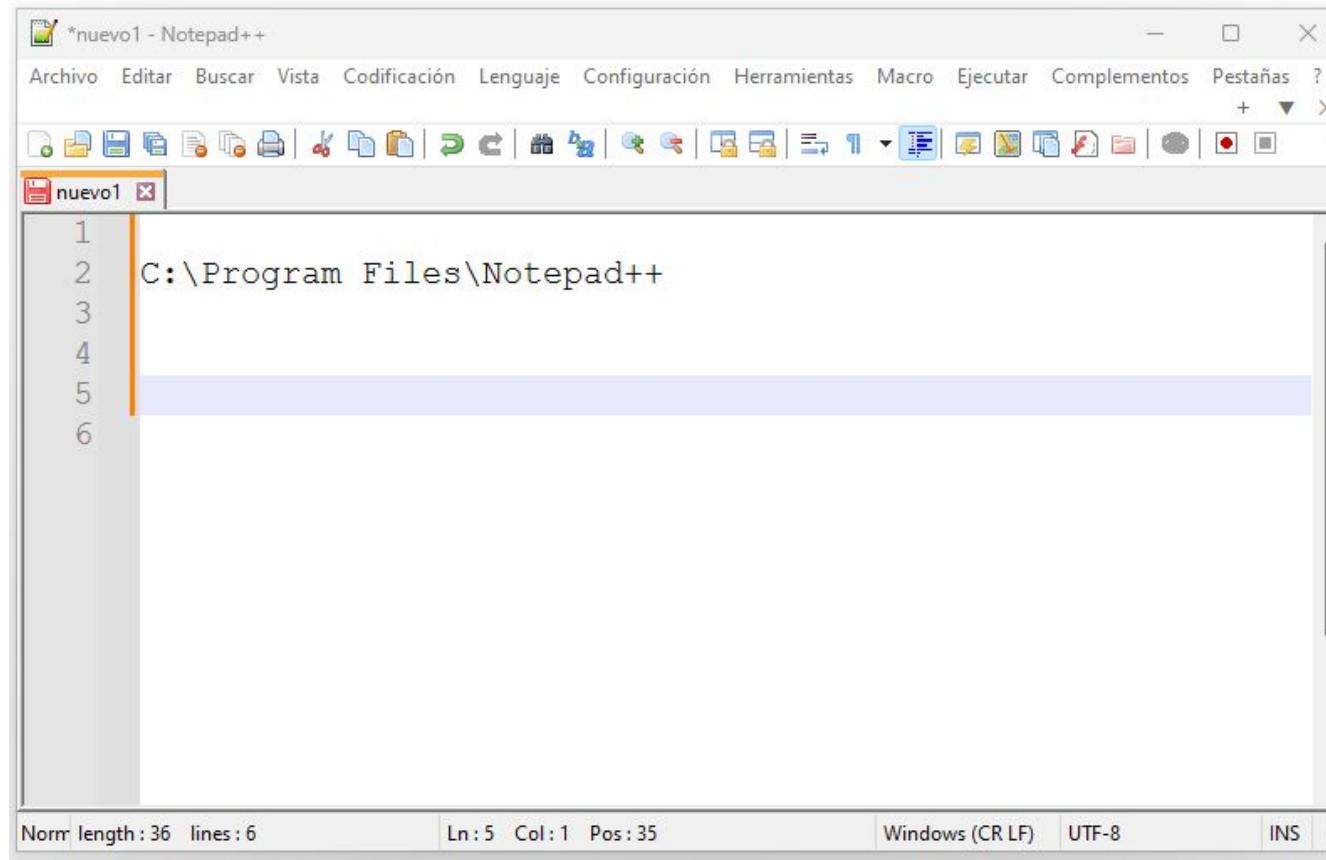


# Editor de Texto Plano: NOTEPAD++



(Don Ho, s.f.)

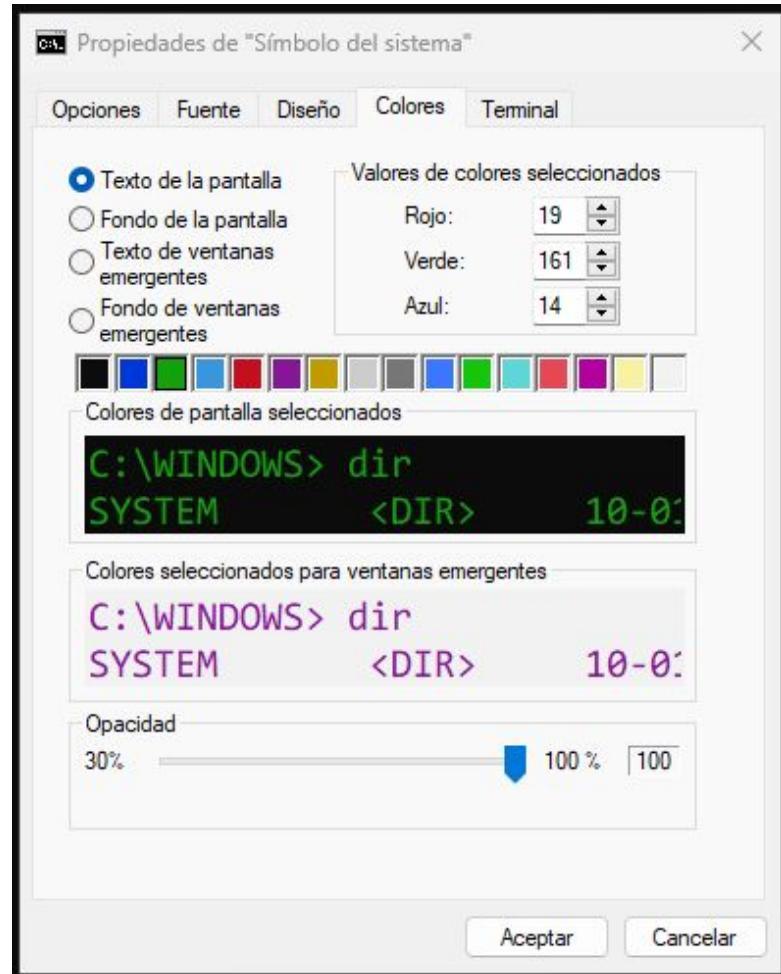
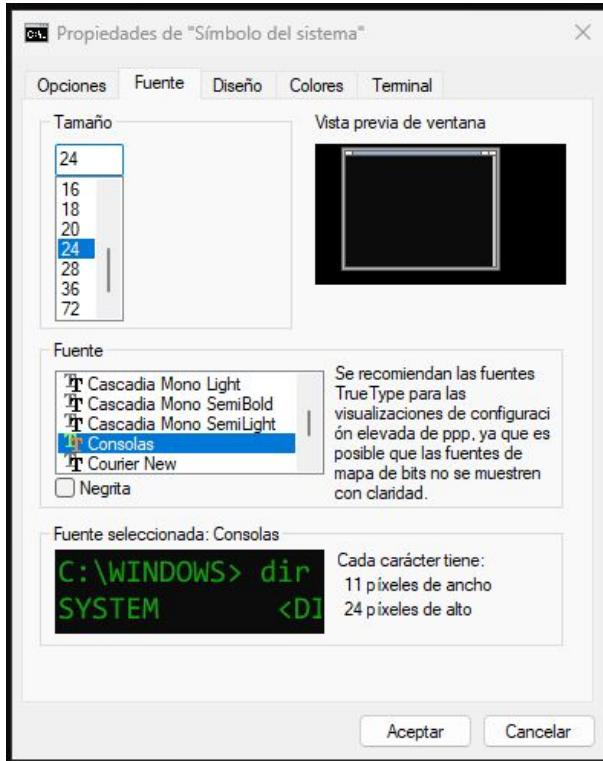
# Editor de Texto Plano: **NOTEPAD++**



# Configuración de Terminal o Consola CMD

En el Buscador de Windows que tiene el símbolo de una lupa digite CMD y activa la opción para que se incorpore en su barra de tareas.

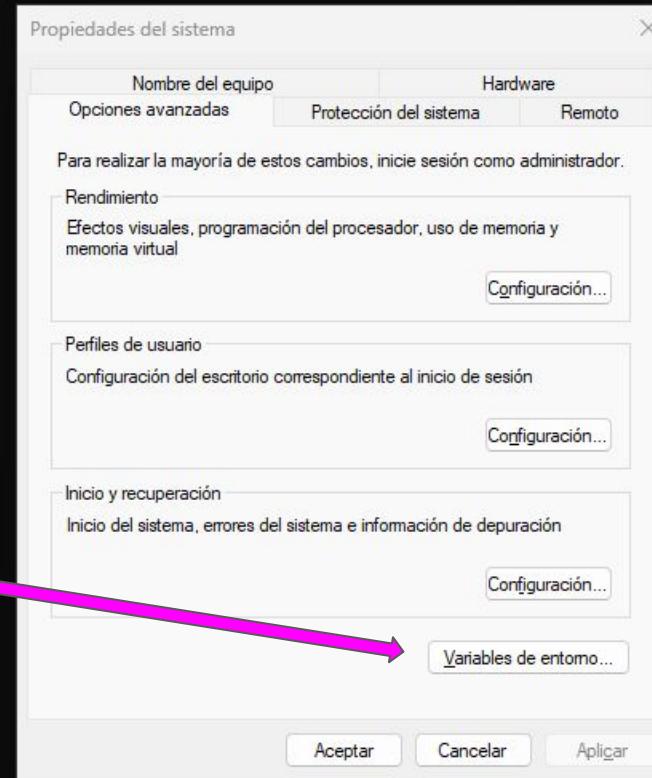
Haciendo anticlick con el mouse en el terminal le aparecerá un menu de propiedades que le permitirá personalizar el terminal o consola.



Símbolo del sistema

C:\Users\OSCAR>SYSTEMPROPERTIESADVANCED

C:\Users\OSCAR>



Terminal CMD

SystemPropertiesAdvanced

# Terminal CMD

Aqui se Instaló Python

Variables de entorno	
Variable	Valor
OneDrive	C:\Users\OSCAR\OneDrive
Path	C:\Users\OSCAR\AppData\Local\Programs\Python\Python312\Scripts\;C:\Users\OSCAR\AppData\Local\Programs\Python\Python312\;C:\Users\OSCAR\AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\OSCAR\AppData\Local\Temp
TMP	C:\Users\OSCAR\AppData\Local\Temp

Nuevo... Editar... Eliminar

## Variables del sistema

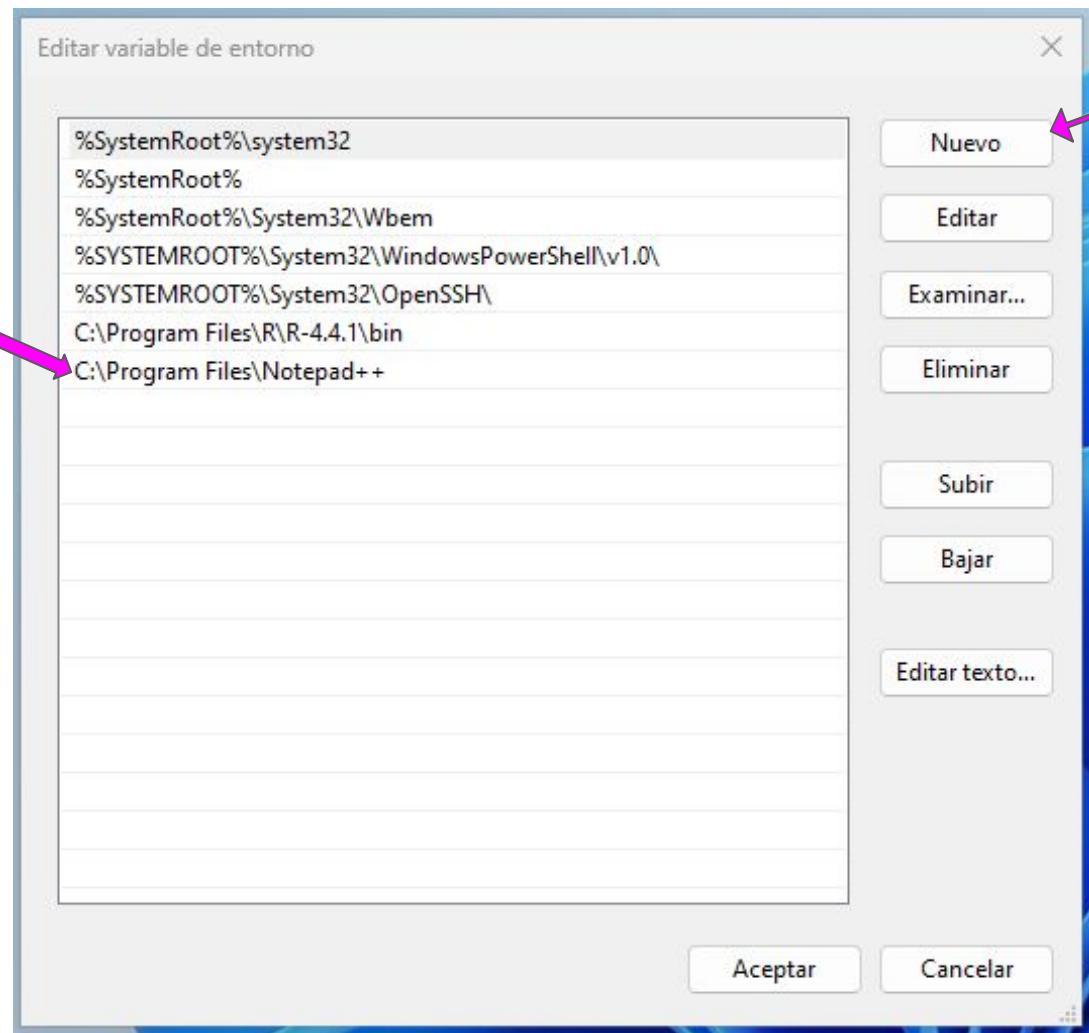
Aqui se Instalará Notepad ++

Variable	Valor
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	16
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH;C:\Program Files\R\R-4.4.1\bin;C:\Progra...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel(R) Core(TM) i7-1165G7 CPU @ 2.80GHz

Nueva... Editar... Eliminar

Aceptar Cancelar

# Terminal CMD



# COMANDOS CMD

- dir** : visualizar el directorio
- cls** : limpiar pantalla
- rd** : borrar carpeta o directorio
- del** : borrar archivo
- mkdir** : crear carpeta
- cd carpeta** : Entrar en la carpeta
- cd ..** : salir de la carpeta
- C:** : ingresar al disco local C
- D:** : ingresar al disco local D
- control** : comando para entrar a ajustes de configuración
- SystemPropertiesAdvanced** : comando para entrar a propiedades del sistema



EBook Gratis

# APRENDIZAJE cmd

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#cmd

(Rip Tutorial, s.f.)

## Ejm 00. Hola

The screenshot shows the Notepad++ application window. The title bar reads "\*C:\Users\OSCAR\code\py\hola\hola.py - Notepad++". The menu bar includes Archivo, Editar, Buscar, Vista, Codificación, Lenguaje, Configuración, Herramientas, Macro, Ejecutar, Complementos, Pestañas, and a help icon. The toolbar contains various file and document-related icons. A status bar at the bottom shows "Pythc length : 46 lines : 4 Ln : 4 Col : 1 Pos : 47" and encoding settings "Windows (CR LF) UTF-8 INS". The main editor window displays the following Python code:

```
1 print("-----")
2 print("Hola Chicos")
3
4
```

En windows:  
**python hola.py**

En Linux:  
**python3 hola.py**

The screenshot shows a Windows Command Prompt window titled "Símbolo del sistema". The command entered is "C:\Users\OSCAR\code\py\hola>notepad++ hola.py". The output shows the execution of the Python script, which prints "-----" and "Hola Chicos". The command prompt then returns to the directory "C:\Users\OSCAR\code\py\hola>".

```
C:\Users\OSCAR\code\py\hola>notepad++ hola.py
C:\Users\OSCAR\code\py\hola>python hola.py
-----
Hola Chicos
C:\Users\OSCAR\code\py\hola>
```

## Ejm 01. Par impar

Las sangrías,  
indentaciones o  
espaciamientos son  
obligatorios.

The screenshot shows a Notepad++ window with the file 'par\_impar.py' open. The code is as follows:

```
1 # python par_impar.py
2
3 # Pedir al usuario que ingrese un número entero
4 numero = int(input("Ingresa un número entero: "))
5
6 # Determinar si el número es par o impar
7 if numero % 2 == 0:
8     print(f"El número {numero} es par.")
9 else:
10    print(f"El número {numero} es impar.")
```

The status bar at the bottom of the Notepad++ window indicates: Python file, length : 296, lines : 13, Ln : 3, Col : 13, Pos : 38, Windows (CR LF), UTF-8, INS.

The terminal window shows the command 'python par\_impar.py' being run. The output is:

```
C:\Users\OSCAR\code\py\par_impar>n par_impar.py
C:\Users\OSCAR\code\py\par_impar>python par_impar.py
Ingresa un número entero: 21
El número 21 es impar.

C:\Users\OSCAR\code\py\par_impar>
```

En windows:  
**python par\_impar.py**

En Linux:  
**python3 par\_impar.py**

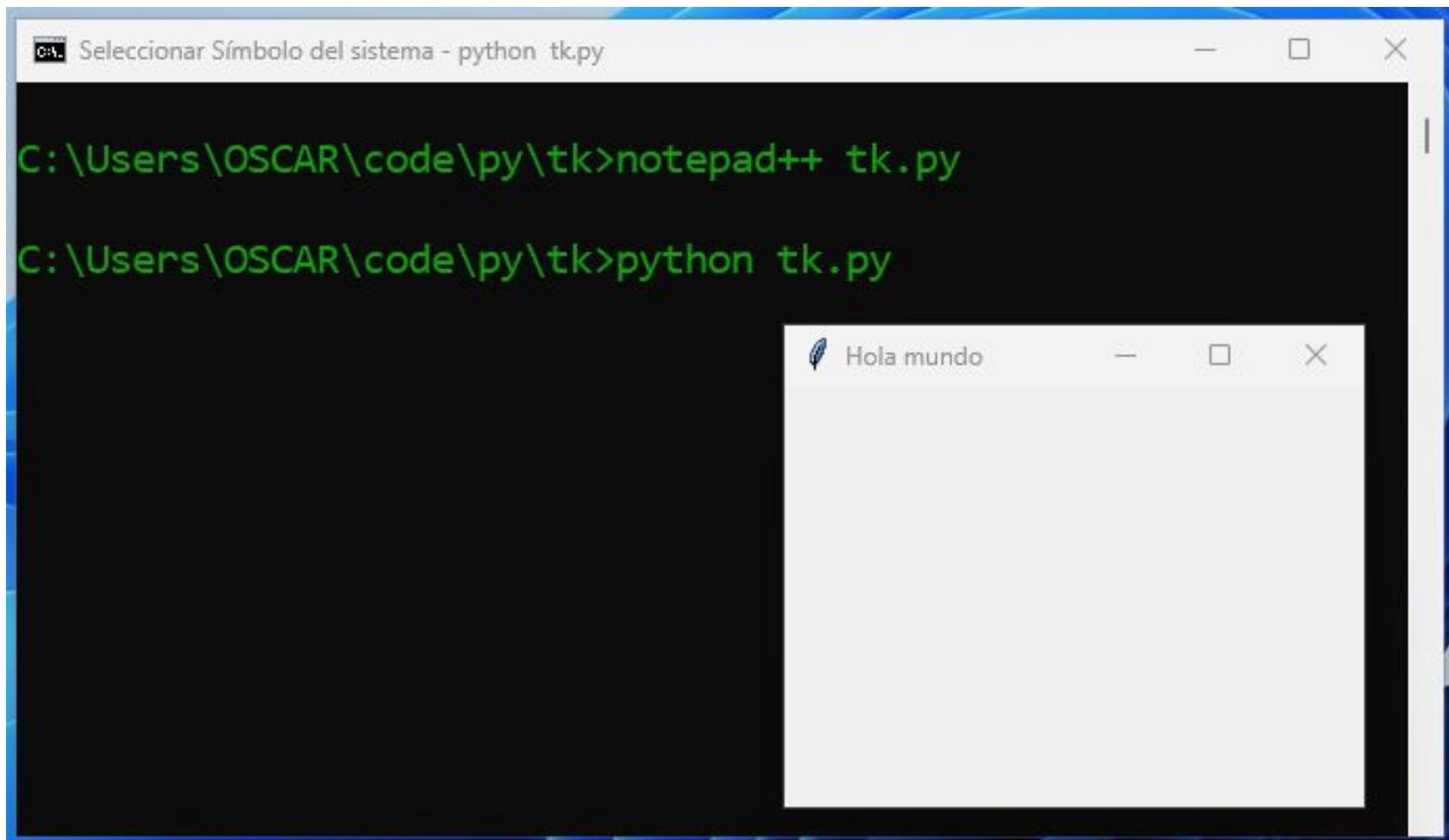
## Ejm 02. Ventana

The screenshot shows a Notepad++ window with the following details:

- Title Bar:** C:\Users\OSCAR\code\py\tk\tk.py - Notepad++
- Menu Bar:** Archivo, Editar, Buscar, Vista, Codificación, Lenguaje, Configuración, Herramientas, Macro, Ejecutar, Complementos, Pestañas, ?
- Toolbar:** Includes icons for file operations like Open, Save, Print, Find, Replace, and various configuration options.
- Code Editor:** The file tk.py contains the following Python code:

```
1  from tkinter import *
2
3  class Alumno:
4      def __init__(self,ventana):
5          self.ventana=ventana
6          self.ventana.title("Hola mundo")
7
8  if __name__=="__main__":
9      ventana=Tk()
10     aplicacion=Alumno(ventana)
11     ventana.mainloop()
12
13 # Referencia
14 # Programador Novato (s.f.). Hola Mundo con Python y TKinter.
15 # https://www.programadornovato.com/hola-mundo-con-python-y-tkinter/
16
17
```
- Status Bar:** Shows Python file, length : 406, lines : 17, Ln : 17, Col : 1, Pos : 407, Windows (CR LF), UTF-8, INS.

## Ejm 02. Ventana



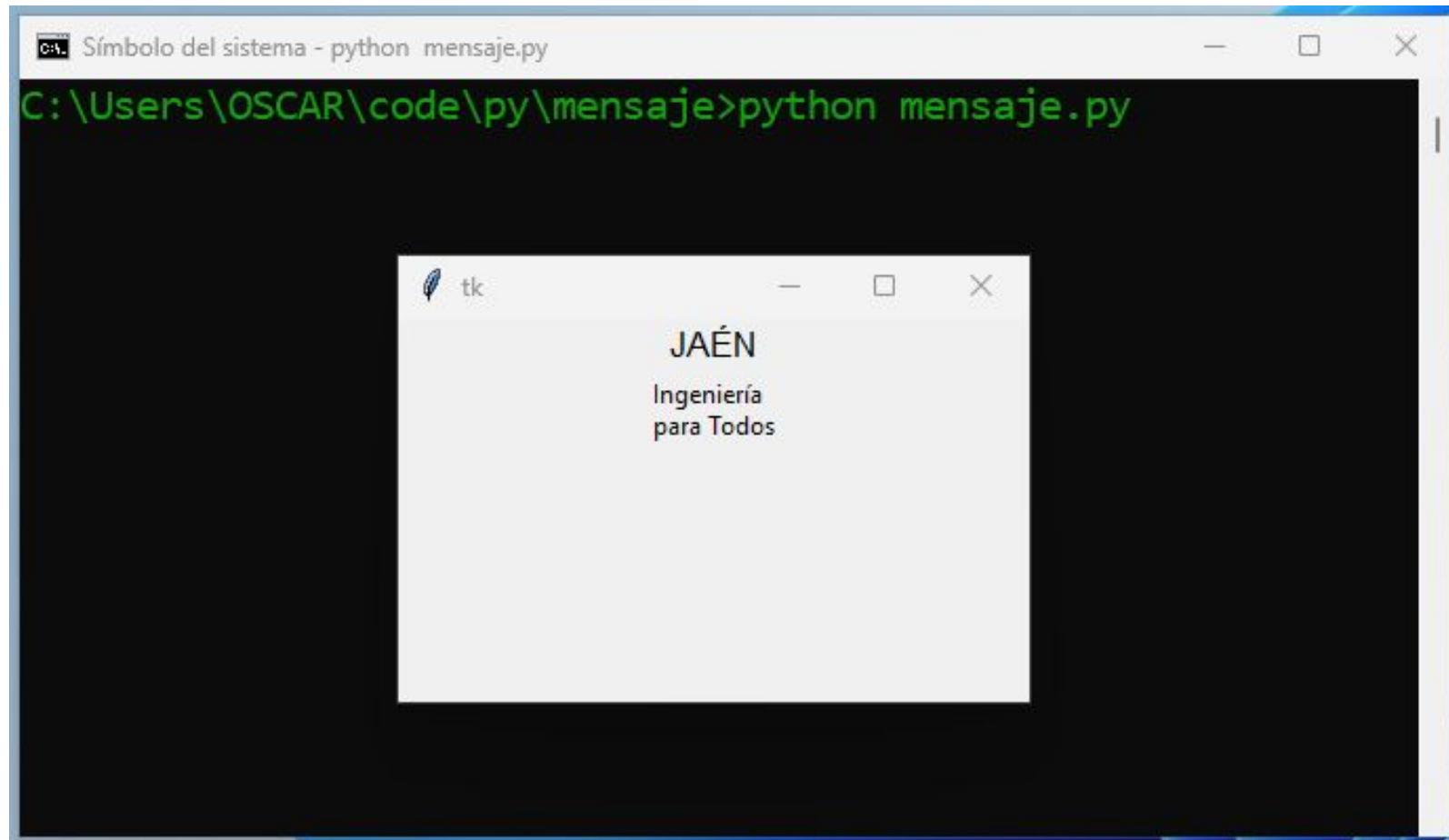
## Ejm 03. Ventana con mensaje

The screenshot shows a Notepad++ window with the following details:

- Title Bar:** \*C:\Users\OSCAR\code\py\mensaje\mensaje.py - Notepad++
- Menu Bar:** Archivo, Editar, Buscar, Vista, Codificación, Lenguaje, Configuración, Herramientas, Macro, Ejecutar, Complementos, Pestañas, ?
- Toolbar:** Includes icons for file operations like Open, Save, Print, and various search and replace functions.
- Code Editor:** The file "mensaje.py" contains the following Python code:

```
1 from tkinter import *
2
3 root = Tk()
4 root.geometry("300x200")
5
6 w = Label(root, text ='JAÉN', font = "50")
7 w.pack()
8
9 msg = Message( root, text = "Ingeniería para Todos")
10
11 msg.pack()
12
13 root.mainloop()
14
15
16 # REFERENCIA
17 # Oscar Núñez Mori, Jaén 17-Jul-2024. basado en:
18 # - Geeks for Geeks (2020). Python Tkinter – Message.
19 #   https://www.geeksforgeeks.org/python-tkinter-message/
20
```
- Status Bar:** Python file, length : 411 lines : 20, Ln : 20 Col : 1 Pos : 412, Windows (CR LF), UTF-8, INS

## Ejm 03. Ventana con mensaje



## Ejm 04. Fibonacci

```
1 # fibonacci.py
2
3 # Solicitamos la posición de la serie hasta la que debemos calcular
4 pos = int(input('Escriba que posición quiere de la serie de Fibonacci: '))
5
6 # Preguntamos si se desea desarrollar la serie
7 ans = input('Desea ver el resultado de toda la serie? ( S/N ) : ')
8
9 # Inicializamos las variables
10 iniFibo = 0
11 finFibo = 1
12
13 # Comenzamos el ciclo de calculo de Fibonacci hasta la posición indicada
14 for x in range(0,pos):
15
16     # Evaluamos las dos primeras posiciones de la serie
17     if x == 0:
18         finFibo = 0
19     elif x == 1:
20         finFibo = 1
21
22     # Si acepto desarrollar la serie se imprime la posición y el valor correspondiente
23     # La posición se le suma 1 ya que el rango comienza en 0 y no en 1
24     if ans =='S' or ans == 's' :
25         print(f'Posición: {x+1:>4} -- Valor en la serie de Fibonacci: {finFibo:>10} ')
26
27     # Guardamos el valor final de la serie antes de actualizarlo
28     sum = finFibo
29
30     # Actualizamos los valores de acuerdo al algoritmo
31     finFibo = finFibo + iniFibo
32     iniFibo = finFibo - iniFibo
33
34 # Imprime el valor final de la serie, independientemente si se desarrollo la serie o no
35 print(f'\nPosición Final: {x+1:>4} -- Valor en la serie de Fibonacci: {sum:>10} ')
36
37 # REFERENCIA
38 # Sagaom (2022, jul 28). Sucesión de Fibonacci en Python.
39 # https://www.sagaom.com/sucesion-de-fibonacci-en-python/
40
```

## Ejm 04. Fibonacci

```
code/py/fibonacci via □ v3.10.12
> python3 fibonacci.py
Escriba que posición quiere de la serie de Fibonacci: 30
Desea ver el resultado de toda la serie? ( S/N ) : s
Posicion: 1 -- Valor en la serie de Fibonacci: 0
Posicion: 2 -- Valor en la serie de Fibonacci: 1
Posicion: 3 -- Valor en la serie de Fibonacci: 1
Posicion: 4 -- Valor en la serie de Fibonacci: 2
Posicion: 5 -- Valor en la serie de Fibonacci: 3
Posicion: 6 -- Valor en la serie de Fibonacci: 5
Posicion: 7 -- Valor en la serie de Fibonacci: 8
Posicion: 8 -- Valor en la serie de Fibonacci: 13
Posicion: 9 -- Valor en la serie de Fibonacci: 21
Posicion: 10 -- Valor en la serie de Fibonacci: 34
Posicion: 11 -- Valor en la serie de Fibonacci: 55
Posicion: 12 -- Valor en la serie de Fibonacci: 89
Posicion: 13 -- Valor en la serie de Fibonacci: 144
Posicion: 14 -- Valor en la serie de Fibonacci: 233
Posicion: 15 -- Valor en la serie de Fibonacci: 377
Posicion: 16 -- Valor en la serie de Fibonacci: 610
Posicion: 17 -- Valor en la serie de Fibonacci: 987
Posicion: 18 -- Valor en la serie de Fibonacci: 1597
Posicion: 19 -- Valor en la serie de Fibonacci: 2584
Posicion: 20 -- Valor en la serie de Fibonacci: 4181
Posicion: 21 -- Valor en la serie de Fibonacci: 6765
Posicion: 22 -- Valor en la serie de Fibonacci: 10946
Posicion: 23 -- Valor en la serie de Fibonacci: 17711
Posicion: 24 -- Valor en la serie de Fibonacci: 28657
Posicion: 25 -- Valor en la serie de Fibonacci: 46368
Posicion: 26 -- Valor en la serie de Fibonacci: 75025
Posicion: 27 -- Valor en la serie de Fibonacci: 121393
Posicion: 28 -- Valor en la serie de Fibonacci: 196418
Posicion: 29 -- Valor en la serie de Fibonacci: 317811
Posicion: 30 -- Valor en la serie de Fibonacci: 514229
Posición Final: 30 -- Valor en la serie de Fibonacci: 514229
```

## Ejm 05. Ecuación Cuadrática

```
1 # ec_2do.py
2
3 from math import sqrt
4
5 # mostramos un mensaje de bienvenida
6 print('¡Hola! Vamos a resolver una ecuación de segundo grado:')
7 print('    ax2 + bx + c = 0\n')
8
9 # pedimos los coeficientes al usuario
10 a, b, c = [float(input(f'Dame el coeficiente {coef}: ')) for coef in ('a', 'b', 'c')]
11
12 # calculamos el discriminante
13 discriminante = b * b - 4 * a * c
14
15 if discriminante < 0: # comprobamos si no existen soluciones reales
16     print(f'La ecuación no tiene soluciones reales.')
17 else:
18     raiz = sqrt(discriminante)      # calculamos la raíz
19     x_1 = (-b + raiz) / (2 * a)    # calculamos una primera solución
20     if discriminante != 0:          # comprobamos si hay otra solución
21         x_2 = (-b - raiz) / (2 * a) # calculamos la segunda solución
22         print(f'Las soluciones son {x_1} y {x_2}.') # mostramos las dos soluciones
23     else:
24         print(f'La única solución es x = {x_1}') # mostramos la única solución
25
26
27 # REFERENCIA
28 # Código Pitón (2024). Cómo Resolver Ecuaciones de Segundo Grado en Python.
29 # https://www.codigopiton.com/resolver-ecuaciones-de-segundo-grado-en-python/
30
```

## Ejm 05. Ecuación Cuadrática

```
code/py/ec_2do via □ v3.10.12
> python3 ec_2do.py
¡Hola! Vamos a resolver una ecuación de segundo grado:
     $ax^2 + bx + c = 0$ 

Dame el coeficiente a: 1
Dame el coeficiente b: 6
Dame el coeficiente c: 8
Las soluciones son -2.0 y -4.0.

code/py/ec_2do via □ v3.10.12 took 4s
> python3 ec_2do.py
¡Hola! Vamos a resolver una ecuación de segundo grado:
     $ax^2 + bx + c = 0$ 

Dame el coeficiente a: 1
Dame el coeficiente b: -4
Dame el coeficiente c: 4
La única solución es  $x = 2.0$ 

code/py/ec_2do via □ v3.10.12 took 4s
> python3 ec_2do.py
¡Hola! Vamos a resolver una ecuación de segundo grado:
     $ax^2 + bx + c = 0$ 

Dame el coeficiente a: 2
Dame el coeficiente b: 1
Dame el coeficiente c: 2
La ecuación no tiene soluciones reales.
```

## Ejm 06.

### Generador de datos ficticios de Demanda Residencial Eléctrica

pip install pandas  
pip install numpy

Genera 10000 datos en el  
archivo:

[demandas\\_residencial.csv](#)

```
1 # generador_demanda.py
2 # Generador de Datos ficticios de Demanda Residencial de Energ. Eléctrica
3 # instalar los paquetes pandas y numpy
4 # pip install pandas
5 # pip install numpy
6
7 import pandas as pd
8 import numpy as np
9
10 # Generar datos simulados
11 np.random.seed(42)
12 timestamps = pd.date_range(start='2023-01-01', periods=10000, freq='H')
13
14
15 # consumo promedio de 5 kWh con desviación estándar de 2
16 consumo_kWh = np.random.normal(loc=5, scale=2, size=10000)
17
18 # precio por kWh entre 0.10 y 0.20
19 precio_kWh = np.random.uniform(low=0.10, high=0.20, size=10000)
20
21 # Crear DataFrame
22 data = pd.DataFrame({
23     'timestamp': timestamps,
24     'consumo_kWh': consumo_kWh,
25     'precio_kWh': precio_kWh
26 })
27
28 # Guardar en un archivo CSV
29 data.to_csv('demanda_residencial.csv', index=False)
30
31 # REFERENCIA
32 # OpenAI(2024). ChatGPT (Ver. 20 Jul) [Estadísticas Demanda Energía].
33 # https://chatgpt.com/share/daa2b9a7-3024-454b-90e7-0e0a8ba7c76b
```

## Ejm 06. Generador de datos fictitious de Demanda Residencial Eléctrica

[demanda\\_residencial.csv](#)

timestamp	consumo_kwh	precio_kwh
2023-01-01 00:00:00	5.993428306022466	0.13212551734139127
2023-01-01 01:00:00	4.72347139765763	0.11030191944422062
2023-01-01 02:00:00	6.295377076201385	0.10718075454454692
2023-01-01 03:00:00	8.046059712816051	0.10942759778989361
2023-01-01 04:00:00	4.531693250553328	0.1582869361192043
2023-01-01 05:00:00	4.531726086101639	0.10516621758289897
2023-01-01 06:00:00	8.158425631014783	0.13369268749640384
2023-01-01 07:00:00	6.534869458305818	0.17531997019142295
2023-01-01 08:00:00	4.061051228130096	0.12943481665153783
2023-01-01 09:00:00	6.085120087171929	0.176330807718013
2023-01-01 10:00:00	4.0731646143750755	0.17454670382070553
2023-01-01 11:00:00	4.068540492859486	0.12458271326332164
2023-01-01 12:00:00	5.483924543132068	0.17288221800927456
2023-01-01 13:00:00	1.1734395106844042	0.17477766382721008
2023-01-01 14:00:00	1.5501643349739345	0.1397453925667027
2023-01-01 15:00:00	3.8754249415180544	0.12104204837415183
2023-01-01 16:00:00	2.9743377593311524	0.15212708068918693
2023-01-01 17:00:00	5.628494665190548	0.19213975880001966
2023-01-01 18:00:00	3.1839518489575784	0.1639616732432062
2023-01-01 19:00:00	2.175392597329417	0.11729224976272612
2023-01-01 20:00:00	7.931297537843108	0.168275692473352
2023-01-01 21:00:00	4.548447399026928	0.14272030272818348
2023-01-01 22:00:00	5.1350564093758475	0.15915287973553732
2023-01-01 23:00:00	2.1505036275730864	0.1734064035023108
2023-01-02 00:00:00	3.9112345509496347	0.1514947351514445
2023-01-02 01:00:00	5.2218451794197325	0.18139839745733088
2023-01-02 02:00:00	2.6980128451553944	0.17221555793166277
2023-01-02 03:00:00	5.751396036691344	0.16150733400185108
2023-01-02 04:00:00	3.79872262016239	0.11507928158323516

## Ejm 07.

# Análisis Estadístico de la Demanda Residencial Eléctrica

Si ya instalaste la librería una vez, no necesitas instalarla nuevamente.

pip install pandas

```
1 # estadistica_demanda.py
2 # debes tener la libreria de pandas
3 # pip install pandas
4
5 import pandas as pd
6
7 # Leer el archivo CSV con 9000 registros
8 data = pd.read_csv('demanda_residencial.csv')
9
10 # Calcular estadísticas básicas
11 estadisticas = data.describe()
12
13 # Calcular la mediana
14 mediana_consumo = data['consumo_kWh'].median()
15 mediana_precio = data['precio_kWh'].median()
16
17 # Calcular la moda
18 moda_consumo = data['consumo_kWh'].mode()[0]
19 moda_precio = data['precio_kWh'].mode()[0]
20
21 # Calcular la correlación de Pearson y Spearman
22 correlacion_pearson = data['consumo_kWh'].corr(data['precio_kWh'], method='pearson')
23 correlacion_spearman = data['consumo_kWh'].corr(data['precio_kWh'], method='spearman')
24
25 # Resultados
26 print("Estadísticas Descriptivas:\n", estadisticas)
27 print("\nMediana del Consumo (kWh):", mediana_consumo)
28 print("Mediana del Precio (kWh):", mediana_precio)
29 print("\nModa del Consumo (kWh):", moda_consumo)
30 print("Moda del Precio (kWh):", moda_precio)
31 print("\nCorrelación de Pearson:", correlacion_pearson)
32 print("Correlación de Spearman:", correlacion_spearman)
33
34 # REFERENCIA
35 # OpenAI (2024). ChatGPT (Ver. 20 Jul.)[Estadística Demanda Energía].
36 # https://chatgpt.com/share/daa2b9a7-3024-454b-90e7-0e0a8ba7c76b
```

## Ejm 07.

### Análisis Estadístico de la Demanda Residencial Eléctrica

Estadísticas Descriptivas:		
	consumo_kwh	precio_kwh
count	10000.000000	10000.000000
mean	4.995728	0.150713
std	2.006925	0.028951
min	-2.844801	0.100005
25%	3.654819	0.125754
50%	4.994810	0.151127
75%	6.342162	0.175720
max	12.852475	0.199992

Mediana del Consumo (kWh): 4.994810048414244  
Mediana del Precio (kWh): 0.1511269305560145

Moda del Consumo (kWh): -2.8448005032366845  
Moda del Precio (kWh): 0.1000048123894311

Correlación de Pearson: -0.0029254350180253244  
Correlación de Spearman: -0.0031107822071078216

## Ejm 08.

### Gráficos Estadísticos de la Demanda Residencial Eléctrica

Si ya instalaste la librería una vez, no necesitas instalarla nuevamente.

pip install pandas  
pip install matplotlib  
pip install seaborn

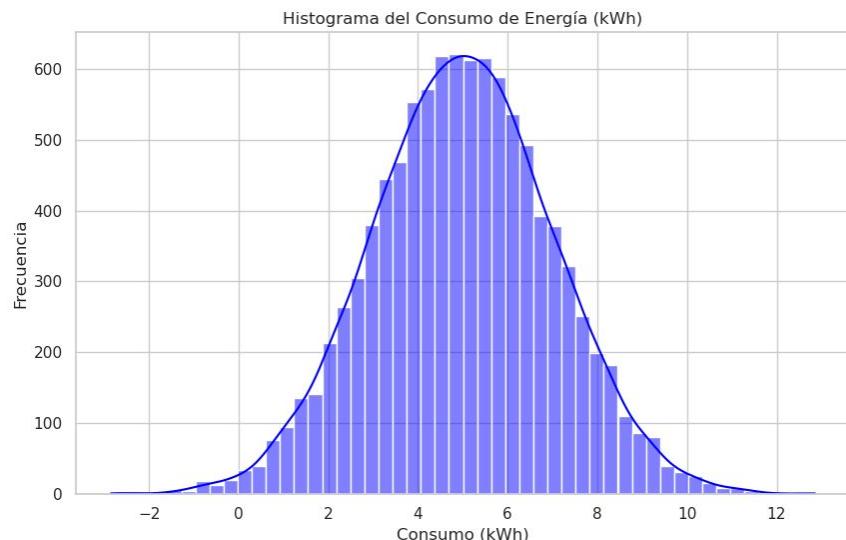
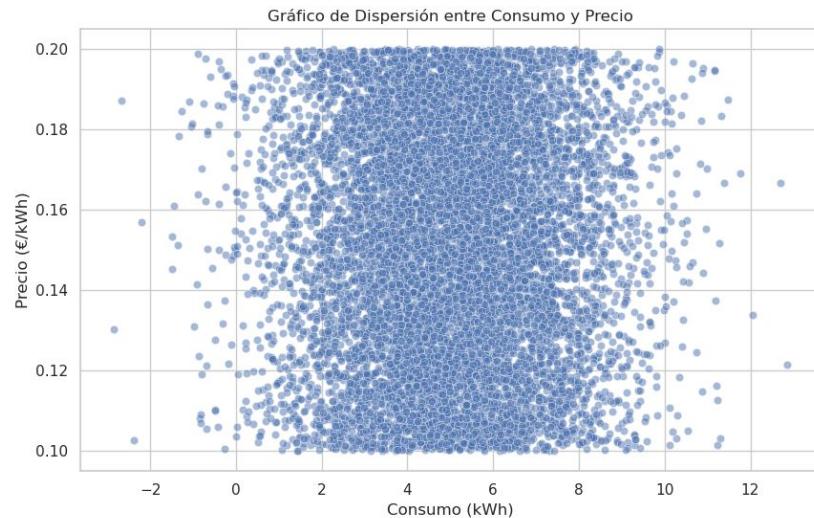
```
1 # graficos_demanda.py
2 # Instalar Librerias
3 # pip install pandas matplotlib seaborn
4
5 import pandas as pd
6 import matplotlib.pyplot as plt
7 import seaborn as sns
8
9 # Leer el archivo CSV con 9000 registros
10 data = pd.read_csv('demanda_residencial.csv')
11
12 # Configurar el estilo de los gráficos
13 sns.set(style="whitegrid")
14
15 # Crear un histograma del consumo de energía
16 plt.figure(figsize=(10, 6))
17 sns.histplot(data['consumo_kWh'], bins=50, kde=True, color='blue')
18 plt.title('Histograma del Consumo de Energía (kWh)')
19 plt.xlabel('Consumo (kWh)')
20 plt.ylabel('Frecuencia')
21 plt.savefig('histograma_consumo.png')
22 plt.show()
23
24 # Crear un boxplot del consumo de energía
25 plt.figure(figsize=(10, 6))
26 sns.boxplot(x=data['consumo_kWh'], color='green')
27 plt.title('Boxplot del Consumo de Energía (kWh)')
28 plt.xlabel('Consumo (kWh)')
29 plt.savefig('boxplot_consumo.png')
30 plt.show()
```

## Ejm 08.

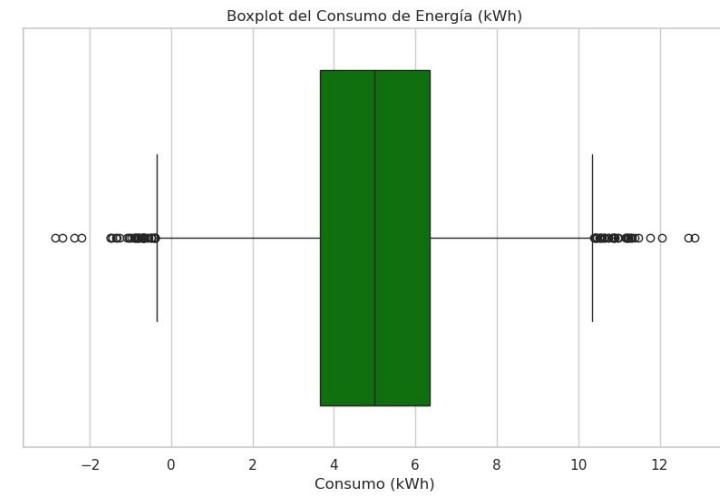
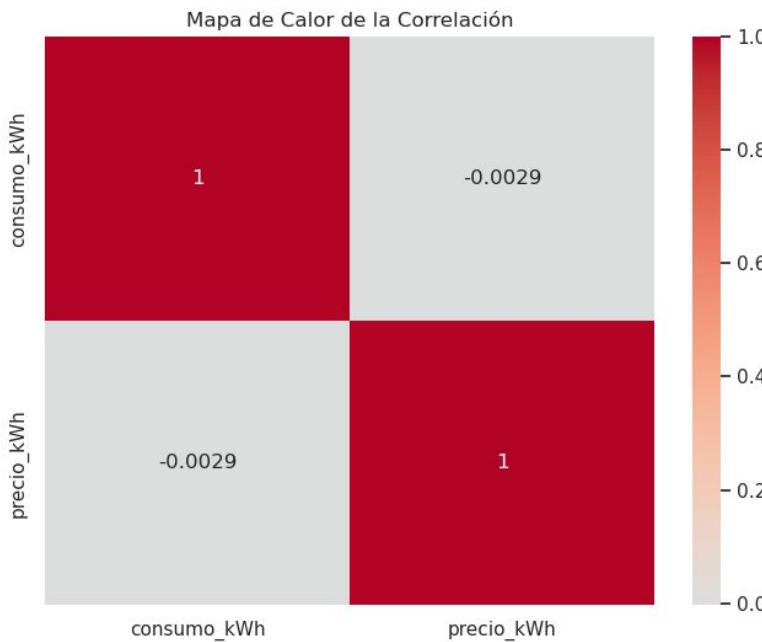
### Gráficos Estadísticos de la Demanda Residencial Eléctrica

```
31
32 # Crear un gráfico de dispersión entre el consumo y el precio
33 plt.figure(figsize=(10, 6))
34 sns.scatterplot(x=data['consumo_kWh'], y=data['precio_kWh'], alpha=0.5)
35 plt.title('Gráfico de Dispersión entre Consumo y Precio')
36 plt.xlabel('Consumo (kWh)')
37 plt.ylabel('Precio (€/kWh)')
38 plt.savefig('scatter_consumo_precio.png')
39 plt.show()
40
41 # Calcular la matriz de correlación
42 correlation_matrix = data.corr()
43
44 # Crear un mapa de calor de la correlación
45 plt.figure(figsize=(8, 6))
46 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
47 plt.title('Mapa de Calor de la Correlación')
48 plt.savefig('heatmap_correlacion.png')
49 plt.show()
50
51 # REFERENCIA
52 # OpenAI (2024). ChatGPT (Ver. 20 Jul.)[Estadística Demanda Energía].
53 # https://chatgpt.com/share/daa2b9a7-3024-454b-90e7-0e0a8ba7c76b
54
```

## Ejm 08. Gráficos Estadísticos de la Demanda Residencial Eléctrica



## Ejm 08. Gráficos Estadísticos de la Demanda Residencial Eléctrica



## Ejm 09.

### Estimación de pi Método Monte Carlo

```
1 # pi_monte_carlo.py
2 #
3 # Método Monte Carlo para encontrar pi.
4 import random # librería de números aleatorios
5
6 def pi_estimado( num_muestras ):
7     circulo_interior = 0
8
9     for _ in range( num_muestras ):
10         x = random.random()
11         y = random.random()
12         distancia = x ** 2 + y ** 2
13         if distancia <= 1:
14             circulo_interior += 1
15
16     return ( circulo_interior / num_muestras ) * 4
17
18 # Número de muestras aleatorias para la estimación
19 num_muestras = 10000000
20 pi_estimado = pi_estimado( num_muestras )
21 print(" ")
22 print( f"Valor estimado de pi después de {num_muestras} muestras: {pi_estimado}" )
23
24 # Oscar Núñez Mori. Jaén, Perú. 21-Jul-2024. Basado en:
25 # Pustam Raut (@pustam_egr) (s.f.). Pi using Monte Carlo method.
26 # https://www.mycompiler.io/view/KHbP70vHDpm
27
```

## Ejm 09. *Estimación de pi Método Monte Carlo*

```
code/py/pi via └ v3.10.12
› python3 pi_monte_carlo.py
```

```
Valor estimado de pi después de 10000000 muestras: 3.1416924
```

## Ejm 10.

### Estimación de pi con Fórmula de Ramanujan y Algoritmo de Chudnovsky

```
1 # pi.py
2 # Estimación de pi basado en:
3 # la fórmula de Ramanujan y el algoritmo de Chudnovsky
4
5 import math
6
7 def pi_ramanujan(iteraciones):
8     # Inicializa la suma
9     sum_term = 0.0
10
11    # Realiza la sumatoria
12    for k in range(iteraciones):
13        term1 = math.factorial(4 * k) * (1103 + 26390 * k)
14        term2 = (math.factorial(k) ** 4) * (396 ** (4 * k))
15        sum_term += term1 / term2
16
17    # Calcula el reciproco de la suma
18    reciproco_sum = 2 * math.sqrt(2) / 9801 * sum_term
19
20    # Estimación de pi
21    pi_estimado = 1 / reciproco_sum
22
23    return pi_estimado
24
25 def pi_chudnovsky(iteraciones):
26     sum_term = 0
27     for k in range(iteraciones):
28         numerador = ((-1) ** k) * math.factorial(6 * k) * (13591409+545140134*k)
29         denominador = math.factorial(3 * k) * (math.factorial(k) ** 3) * (640320 ** (3 * k))
30         sum_term += numerador / denominador
31         total_sum = math.sqrt(10005)/4270934400 * sum_term
32     return 1/total_sum
33
```

## Ejm 10. Estimación de pi con Fórmula de Ramanujan y Algoritmo de Chudnovsky

```
34 # Numero de iteraciones para la aproximación
35 iteraciones = 10
36
37 # Calcula el valor estimado de pi
38 pi_r = pi_ramanujan(iteraciones)
39 pi_c = pi_chudnovsky(iteraciones)
40
41 # Compara con el valor actual de pi
42 # pi_actual = 3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170679
43 pi_actual = math.pi
44
45 print("")
46 print(f"PI actual (aprox.) : ", "%.50f" % pi_actual)
47 print(f"PI estimado (Ramanujan) : ", "%.50f" % pi_r)
48 print(f"PI estimado (Chudnovsky) : ", "%.50f" % pi_c)
49
50 # REFERENCIA
51 # Oscar Núñez Mori. Jaén, Perú. 21-Jul.-2024. Basado en:
52 # - Pustam Raut (@pustam_egr) (s.f.). Pi using Ramanujan's formula and Chudnovsky algorithm.
53 #   https://www.mycompiler.io/view/EaUqvGDExgh
54
```

## Ejm 10. *Estimación de pi con Fórmula de Ramanujan y Algoritmo de Chudnovsky*

```
code/py/pi via ⓘ v3.10.12
› python3 pi.py

PI actual      (Aprox.)      : 3.14159265358979311599796346854418516159057617187500
PI estimado   (Ramanujan)   : 3.14159265358979311599796346854418516159057617187500
PI estimado   (Chudnovsky)  : 3.14159265358979356008717331860680133104324340820312
```

## Ejm 11. Conversor Térmico

Normalmente la librería Tk viene por defecto en la instalación de python en windows, pero sino hay que instalarla solo una vez.

pip install tk

```
1 # temperatura.py
2
3 import tkinter as tk
4
5 # Función para convertir de grados Celsius a Fahrenheit
6 def celsius_a_fahrenheit():
7     try:
8         celsius = float(celsius_entry.get())
9         fahrenheit = (celsius * 9/5) + 32
10        result_label.config(text=f"{celsius:.2f}°C = {fahrenheit:.2f}°F")
11    except ValueError:
12        result_label.config(text="Entrada Invalida")
13
14 # Función para convertir de grados Fahrenheit a Celsius
15 def fahrenheit_a_celsius():
16    try:
17        fahrenheit = float(fahrenheit_entry.get())
18        celsius = (fahrenheit - 32) * 5/9
19        result_label.config(text=f"{fahrenheit:.2f}°F = {celsius:.2f}°C")
20    except ValueError:
21        result_label.config(text="Entrada Invalida")
22
23 # Creando la ventana principal
24 principal = tk.Tk()
25 principal.geometry("500x250") # Ajusta el tamaño de la ventana
26 principal.title("Conversor Térmico")
27
28 # Conversión de grados Celcius a Fahrenheit
29 celsius_label = tk.Label(principal, text="Ingrese Celsius:")
30 celsius_label.grid(row=0, column=0)
31
32 celsius_entry = tk.Entry(principal)
33 celsius_entry.grid(row=0, column=1)
34
35 c_to_f_button = tk.Button(principal, text="Convierte a Fahrenheit", command=celsius_a_fahrenheit)
36 c_to_f_button.grid(row=0, column=2)
37
```

## Ejm 11. Conversor Térmico

```
38 # Conversión de Grados Fahrenheit a Celsius
39 fahrenheit_label = tk.Label(principal, text="Ingrese Fahrenheit:")
40 fahrenheit_label.grid(row=1, column=0)
41
42 fahrenheit_entry = tk.Entry(principal)
43 fahrenheit_entry.grid(row=1, column=1)
44
45 f_to_c_button = tk.Button(principal, text="Convierte a Celsius", command=fahrenheit_a_celsius)
46 f_to_c_button.grid(row=1, column=2)
47
48 # Muestra los resultados
49 result_label = tk.Label(
50     principal, text="", font=("Helvetica", 18), bg="lightblue", height=3, width=30, bd=3,
51     cursor="hand2", fg="red", padx=15, pady=15, justify=tk.CENTER, relief=tk.RAISED,
52     wraplength=250 )
53 result_label.grid(row=2, columnspan=3)
54
55 # relief=tk.RAISED
56 # Iniciar el bucle de eventos de Tkinter
57 principal.mainloop()
58
59
60 # REFERENCIA
61 # Oscar Núñez Mori. Jaén, Perú. 21-Jul-2024. Basado en:
62 # - w3resource (2023, Set 13). Create a temperature converter in Python using Tkinter
63 #   https://www.w3resource.com/python-exercises/tkinter/python-tkinter-basic-exercise-15.php
64 
```

### Conversor Térmico

Ingrese Celsius:  Convierte a Fahrenheit

Ingrese Fahrenheit:  Convierte a Celsius

40.00°C = 104.00°F

### Ejm 11. Conversor Térmico

### Conversor Térmico

Ingrese Celsius:  Convierte a Fahrenheit

Ingrese Fahrenheit:  Convierte a Celsius

30.00°F = -1.11°C

# Ejm 12.

## Análisis de Mantenimiento Industrial

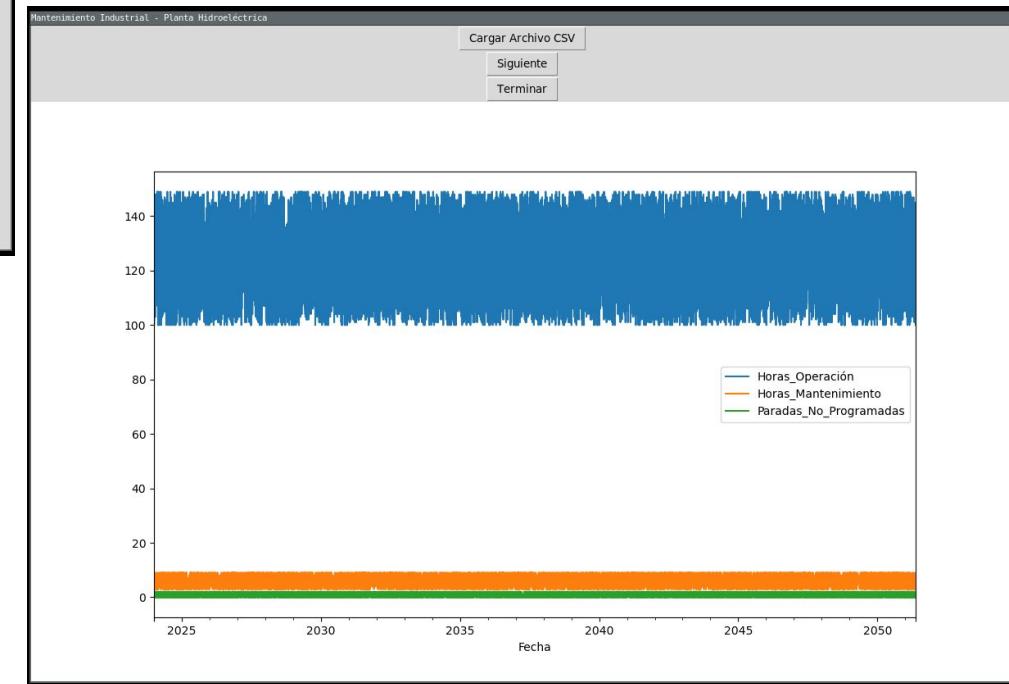
```
1 # mantenimiento.py
2
3 # pip install tk pandas numpy matplotlib seaborn # instalar estas librerías
4 import tkinter as tk
5 from tkinter import filedialog
6 import pandas as pd
7 import matplotlib.pyplot as plt
8 import seaborn as sns
9 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
10
11 # Creación de la clase Mantenimiento
12 class Mantenimiento:
13     def __init__(self, raiz):
14         self.raiz = raiz
15         self.raiz.title("Mantenimiento Industrial - Planta Hidroeléctrica")
16         # Botón de Carga de Archivo .csv
17         self.load_button = tk.Button(raiz, text="Cargar Archivo CSV", command=self.load_file)
18         self.load_button.pack()
19         # Botón de cambio de imagen
20         self.next_button = tk.Button(raiz, text="Siguiente", command=self.show_next_graph)
21         self.next_button.pack()
22         self.next_button.config(state=tk.DISABLED)
23         # Botón para salir del programa
24         self.quit_button = tk.Button(raiz, text="Terminar", command=raiz.quit)
25         self.quit_button.pack()
26
27         self.canvas = None
28         self.graphs = []
29         self.current_graph = 0
30
31         # Configurar tamaño de la ventana
32         self.raiz.geometry("1200x800")
33         # Función para cargar el archivo .csv
34     def load_file(self):
35         file_path = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
36         if file_path:
37             data = pd.read_csv(file_path)
38             self.prepare_graphs(data)
39             self.current_graph = 0
40             self.show_next_graph()
41             self.next_button.config(state=tk.NORMAL)
```

# Ejm 12.

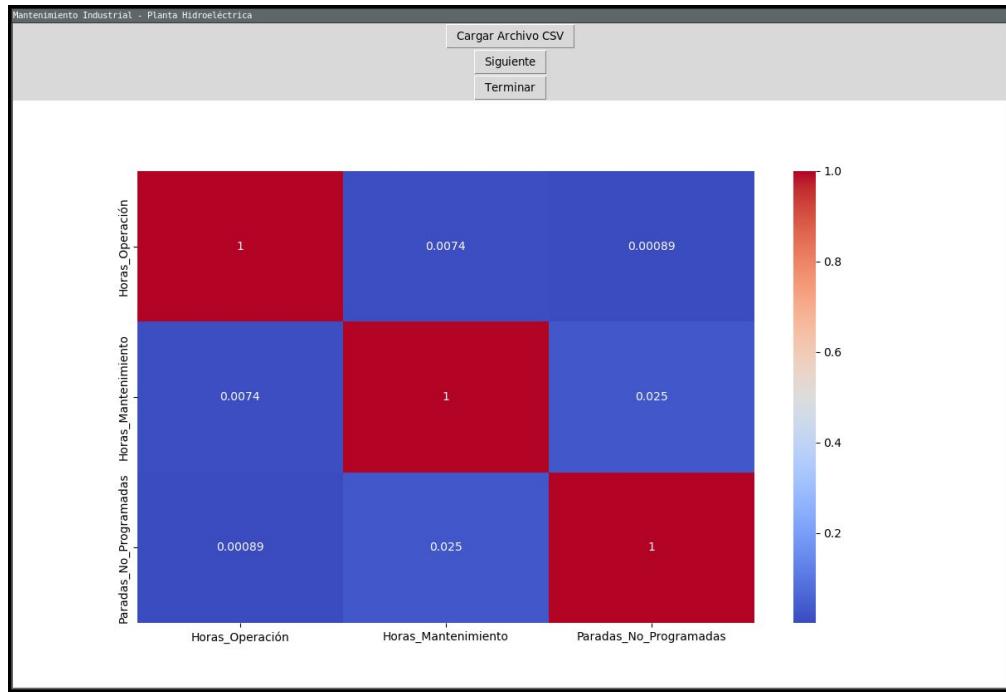
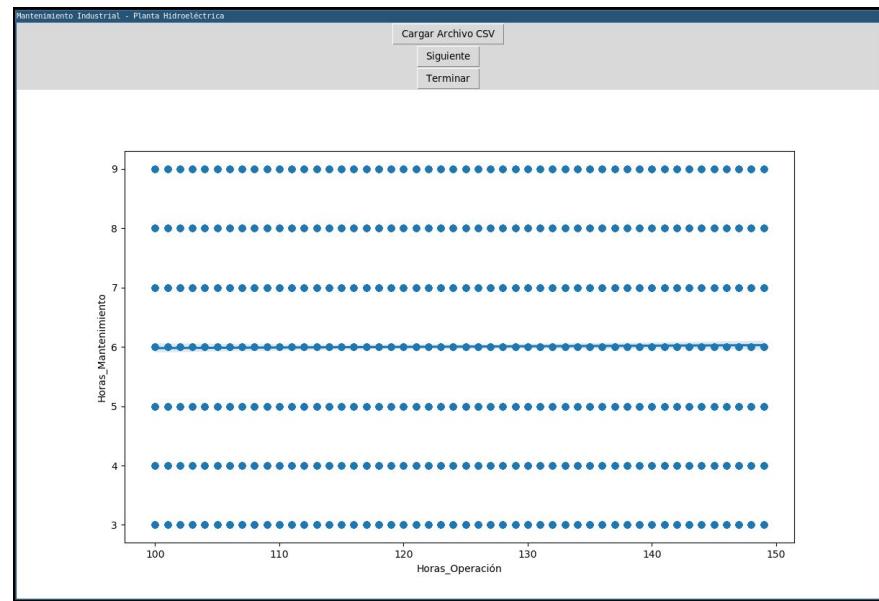
## Análisis de Mantenimiento Industrial

```
42 # Función para generar los gráficos del archivo .csv
43 def prepare_graphs(self, data):
44     self.graphs = []
45
46     # Preparar primer gráfico (gráfico de series temporales)
47     fig, ax = plt.subplots(figsize=(12, 8))
48     data["Fecha"] = pd.to_datetime(data["Fecha"])
49     data.set_index("Fecha", inplace=True)
50     data.plot(ax=ax)
51     self.graphs.append(fig)
52
53     # Preparar segundo gráfico (mapa de calor de correlación)
54     fig_corr, ax_corr = plt.subplots(figsize=(12, 8))
55     sns.heatmap(data.corr(), annot=True, cmap='coolwarm', ax=ax_corr)
56     self.graphs.append(fig_corr)
57
58     # Preparar tercer gráfico (gráfico de regresión)
59     fig_reg, ax_reg = plt.subplots(figsize=(12, 8))
60     sns.regplot(x=data['Horas_Operación'], y=data['Horas_Mantenimiento'], ax=ax_reg)
61     self.graphs.append(fig_reg)
62
63 # Función para mostrar los gráficos en la ventana
64 def show_next_graph(self):
65     if self.canvas:
66         self.canvas.get_tk_widget().pack_forget()
67
68         fig = self.graphs[self.current_graph]
69         self.canvas = FigureCanvasTkAgg(fig, master=self.raiz)
70         self.canvas.draw()
71         self.canvas.get_tk_widget().pack()
72
73         self.current_graph = (self.current_graph + 1) % len(self.graphs)
74
75 # Función Principal
76 if __name__ == "__main__":
77     raiz = tk.Tk()
78     aplicacion = Mantenimiento(raiz)
79     raiz.mainloop()
80
81 # Oscar Núñez Mori. Jaén, Perú. 22-Jul-2024. Basado en:
82 # - OpenAI (2024). ChatGPT (Ver. 21 Jul.) [Tkinter CSV Data Visualization].
83 # - https://chatgpt.com/share/d40d1ef5-1f12-4bcd-9ba3-040f993cb67c
```

## Ejm 12. Análisis de Mantenimiento Industrial



## Ejm 12. Análisis de Mantenimiento Industrial



# Videos Complementarios

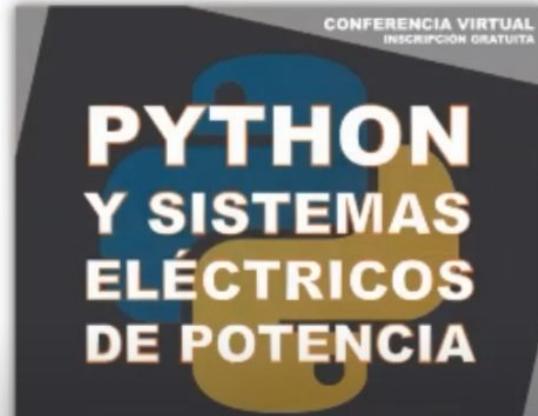
```
1 self._file = None
2 self._fingerprints = {}
3 self._logdups = True
4 self._debug = debug
5 self._logger = logging.getLogger(__name__)
6 if path:
7     self._file = open(os.path.join(path), 'w')
8     self._file.seek(0)
9     self._fingerprints = json.load(self._file)
10
11 @classmethod
12 def from_settings(cls, settings):
13     debug = settings.getboolean('DEBUG', False)
14     return cls(job_dir(settings), debug)
15
16 def request_seen(self, request):
17     fp = self.request_fingerprint(request)
18     if fp in self._fingerprints:
19         return True
20     self._fingerprints.add(fp)
21     if self._file:
22         self._file.write(fp + os.linesep)
23
24 def request_fingerprint(self, request):
25     return request_fingerprint(request)
```



## Aplicaciones de Python en la Industria Eléctrica



(INEL, 2021)



(CECACIER, s.f.)



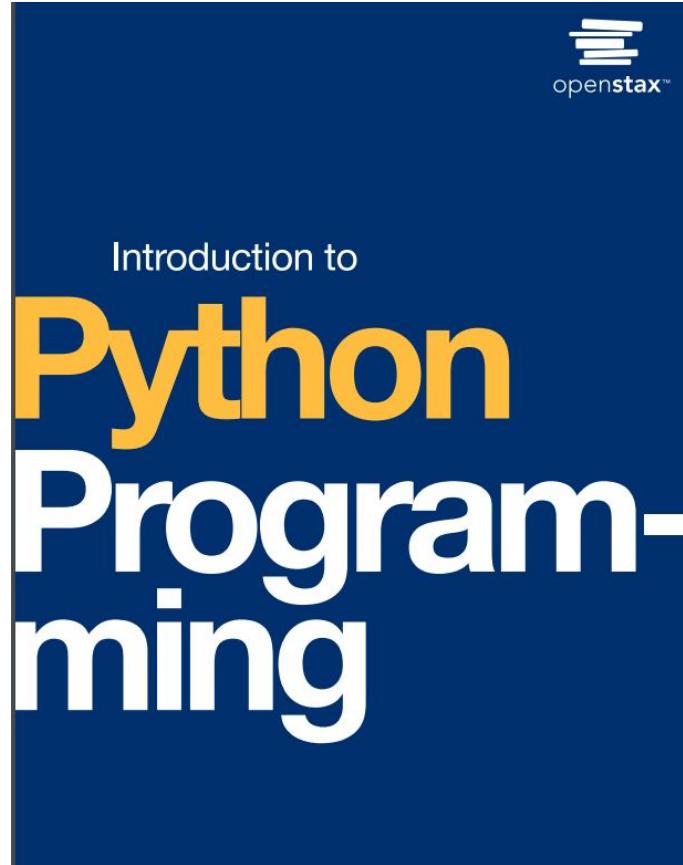
## Aplicaciones de Python para Ingenieros Mecánicos de Fluidos

(Castillo, s.f.)

## Libros Recomendados



(Libro de Python, 2024)



(Das, Lawson, Mayfield, Madison y Norouzi, 2024)



Меня зовут Данा.

СПАСИБО  
БОЛЬШОЕ

# Referencias

Atul Harsha (2024). *What is Python? A Beginner's Guide to Learn Python in 2023.* <https://www.shiksha.com/online-courses/what-is-python-st619-tg21>

Python Software Foundation (2024). <https://www.python.org/>

Don Ho (s.f.). *Notepad++*. <https://notepad-plus-plus.org/>

Rip Tutorial (s.f.). *Aprendizaje CMD*. <https://riptutorial.com/Download/cmd-es.pdf>

Wikipedia (s.f.). *Python*. <https://es.wikipedia.org/wiki/Python>

INEL (2021, Abr 29). *Aplicaciones de Python en la Industria Eléctrica*. <https://www.youtube.com/watch?v=MEtGPaOTE-Q>

CECACIER (s.f.). *Python y sistemas eléctricos de potencia*. <https://www.youtube.com/watch?v=nn1K40pVqMA>

Castillo, F. (s.f.). *Aplicaciones de Python para Ingenieros Mecánicos de Fluidos*. [https://www.youtube.com/watch?v=r\\_cCdIEAvxQ](https://www.youtube.com/watch?v=r_cCdIEAvxQ)

Libro de Python (2024). <https://ellibrodepython.com/>

Das, U., Lawson, A., Mayfield, C., Madison, J. y Norouzi, N.(2024). *Introduction to Python Programming*. Openstax. [https://assets.openstax.org/oscms-prodcms/media/documents/Introduction\\_to\\_Python\\_Programming\\_-\\_WEB.pdf](https://assets.openstax.org/oscms-prodcms/media/documents/Introduction_to_Python_Programming_-_WEB.pdf)