Escuela Politécnica Nacional

Nombre: Francisco Ulloa
Fecha: Quito, 28 de enero de 2026
Tema: Formatters de Python
Repositorio:
https://github.com/Fu5CHAR/Metodos_numericos_2025B_Ulloa-Francisco/tree/main

Black

Black es un formatter **opinionated**: toma todas las decisiones por ti y siempre produce el mismo resultado.

- Se instala fácilmente con `pip install black`.
- Tiene extensión oficial para VS Code.
- Permite formatear archivos completos, carpetas y proyectos.

autopep8

autopep8 es un formatter más **flexible**, orientado a corregir violaciones del estándar PEP 8.

- Instalación sencilla con `pip install autopep8`.
- Muy bien integrado con VS Code.
- Permite formatear archivos completos **o solo fragmentos de código**.

Ruff (formatter)

- Se instala con `pip install ruff`.
- Necesario modificar el archivo settings.jason de VS Code.

-Solo se ejecuta sobre **archivos completos o proyectos**,

YAPF

YAPF (Yet Another Python Formatter) fue creado por Google.

- Se instala con `pip install yapf`.
- Solo se usa para formatear archivos o proyectos completos.
- Menos integración directa con formateo por selección en editores.
- Puede producir resultados distintos según la configuración elegida.

Ejemplos

In [ ]:
```
##Con black
import numpy as np
```

```
import matplotlib.pyplot as plt


def jacobi_modificado(A, b, X0, TOL=1e-6, N=100):
    A = np.array(A, dtype=float)
    b = np.array(b, dtype=float)
    X0 = np.array(X0, dtype=float)

    n = len(b)

    # Historial de iteraciones (iteración 0)
    tabla = [X0.copy()]

    #  🔹 Verificación previa (iteraciones = 0)
    if np.all(np.abs(A @ X0 - b) < TOL):
        print("Número de iteraciones: 0")
        return np.array(tabla)

    k = 1

    while k <= N:
        x = np.zeros(n)

        # Paso 3
        for i in range(n):
            suma = 0.0
            for j in range(n):
                if j != i:
                    suma += A[i, j] * X0[j]

            x[i] = (b[i] - suma) / A[i, i]

        tabla.append(x.copy())

        # Paso 4: condición componente a componente
        if np.all(np.abs(x - X0) < TOL):
            print("Número de iteraciones:", k - 1)
            return np.array(tabla)

        X0 = x.copy()
        k += 1

    return "Número máximo de iteraciones excedido"
```

In [ ]:

```
## Con autopep8
import numpy as np
import matplotlib.pyplot as plt


def jacobi_modificado(A, b, X0, TOL=1e-6, N=100):
    A = np.array(A, dtype=float)
    b = np.array(b, dtype=float)
    X0 = np.array(X0, dtype=float)

    n = len(b)

    # Historial de iteraciones (iteración 0)
    tabla = [X0.copy()]

    #  🔹 Verificación previa (iteraciones = 0)
```

```
    if np.all(np.abs(A @ X0 - b) < TOL):
        print('Número de iteraciones: 0')
        return np.array(tabla)

    k = 1

    while k <= N:
        x = np.zeros(n)

        # Paso 3
        for i in range(n):
            suma = 0.0
            for j in range(n):
                if j != i:
                    suma += A[i, j] * X0[j]

            x[i] = (b[i] - suma) / A[i, i]

        tabla.append(x.copy())

        # Paso 4: condición componente a componente
        if np.all(np.abs(x - X0) < TOL):
            print('Número de iteraciones:', k - 1)
            return np.array(tabla)

        X0 = x.copy()
        k += 1

    return "Número máximo de iteraciones excedido"
```

In [ ]:

```
## Con Ruff
import numpy as np
import matplotlib.pyplot as plt


def jacobi_modificado(A, b, X0, TOL=1e-6, N=100):
    A = np.array(A, dtype=float)
    b = np.array(b, dtype=float)
    X0 = np.array(X0, dtype=float)

    n = len(b)

    # Historial de iteraciones (iteración 0)
    tabla = [X0.copy()]

    #  🔹 Verificación previa (iteraciones = 0)
    if np.all(np.abs(A @ X0 - b) < TOL):
        print("Número de iteraciones: 0")
        return np.array(tabla)

    k = 1

    while k <= N:
        x = np.zeros(n)

        # Paso 3
        for i in range(n):
            suma = 0.0
            for j in range(n):
                if j != i:
```

```
                    suma += A[i, j] * X0[j]

            x[i] = (b[i] - suma) / A[i, i]

        tabla.append(x.copy())

        # Paso 4: condición componente a componente
        if np.all(np.abs(x - X0) < TOL):
            print("Número de iteraciones:", k - 1)
            return np.array(tabla)

        X0 = x.copy()
        k += 1

    return "Número máximo de iteraciones excedido"
```

In [ ]:

```
## Con YAPF
import numpy as np
import matplotlib.pyplot as plt


def jacobi_modificado(A, b, X0, TOL=1e-6, N=100):
    A = np.array(A, dtype=float)
    b = np.array(b, dtype=float)
    X0 = np.array(X0, dtype=float)

    n = len(b)

    # Historial de iteraciones (iteración 0)
    tabla = [X0.copy()]

    #  🔹 Verificación previa (iteraciones = 0)
    if np.all(np.abs(A @ X0 - b) < TOL):
        print('Número de iteraciones: 0')
        return np.array(tabla)

    k = 1

    while k <= N:
        x = np.zeros(n)

        # Paso 3
        for i in range(n):
            suma = 0.0
            for j in range(n):
                if j != i:
                    suma += A[i, j] * X0[j]

            x[i] = (b[i] - suma) / A[i, i]

        tabla.append(x.copy())

        # Paso 4: condición componente a componente
        if np.all(np.abs(x - X0) < TOL):
            print('Número de iteraciones:', k - 1)
            return np.array(tabla)

        X0 = x.copy()
        k += 1
```

```
    return 'Número máximo de iteraciones excedido'
```