Explicación de la rutina en Python del problema del caudal.

Importaciones y constantes:

```
8   import math
9   import tkinter as tk
10  from tkinter import messagebox
11
12  # -------------------- CONSTANTES --------------------
13  G = 9.81
14  PI = math.pi
```

Se importan las librerías necesarias (math y tkinter) y se definen las constantes gravedad (G) y pi (PI), que se usarán en las ecuaciones de flujo.

tkinter se utiliza para la interfaz gráfica y messagebox permite mostrar mensajes de error.

Funciones básicas: área y número de Reynolds:

```
19  def area(D):
20      return PI * (D ** 2) / 4
21
22
23  def reynolds(Q, D, nu):
24      return (Q / area(D)) * D / nu
```

Estas funciones calculan el **área de la tubería** y el **número de Reynolds**, que indica si el flujo es laminar o turbulento.
Ambas son fundamentales para los cálculos hidráulicos del programa.

Ecuación de Colebrook–White:

```
16  def colebrook(Re, k, D, f0=0.02, tol=1e-8):
17      if Re < 2000:
18          return 64 / Re, 0  # flujo laminar
19
20      f = f0
21      # enough big to enter the loop
22      er = 100
23      while(er > tol):
24          f_new = (1 / (-2 * math.log10((k / (3.7 * D)) + (2.51 / (Re * math.sqrt(f)))))) ** 2
25          er = abs((f_new - f) / f_new)
26
27          # update f
28          f = f_new
29      return f
```

Implementa la **ecuación de Colebrook–White** para calcular el factor de fricción f de forma iterativa. Si el flujo es laminar, aplica directamente $f = 64/Re$. En turbulento, repite hasta que el error entre iteraciones sea mínimo.

Estimación de Swamee–Jain:

```python
def swamee_jain_f(Re, k, D):
    """Estimación explícita de f por Swamee-Jain (1976)."""
    if Re < 2000:
        return 64 / Re
    return 0.25 / (math.log10((k / (3.7 * D)) + (5.74 / (Re ** 0.9)))) ** 2
```

Esta función calcula una **aproximación rápida** del factor de fricción f mediante la fórmula de **Swamee–Jain**, que evita iteraciones y sirve como punto de partida para el método de Colebrook.

Cálculo del caudal a partir de la pérdida de carga:

```python
def q_from_hf(f, D, L, Hf, g=G):
    """Despeje de Darcy-Weisbach para Q."""
    num = (PI ** 2) * g * Hf * (D ** 5)
    den = 8 * f * L
    return math.sqrt(num / den)
```

A partir de la ecuación de **Darcy–Weisbach**, despeja el caudal Q en función de f, D, L y H_f.
Es la ecuación base del programa.

Estimación inicial del caudal — función initialGuessQ()

```python
def initialGuessQ(D , L, k, v, Hf):
    aFactor = -0.965*math.sqrt(g*(D ** 5)*Hf/(L))
    bFactor = math.log((k/(3.7*D)) + math.sqrt((3.17*(v**2)*L)/(g*(D**3)*Hf)))
    return aFactor * bFactor
```

Esta función calcula una **estimación inicial del caudal** Q para iniciar las iteraciones.
Combina el efecto del diámetro, la longitud, la rugosidad, la viscosidad y la pérdida de carga mediante una expresión empírica.
El valor obtenido sirve como **punto de partida** para el método de Colebrook–White, acelerando la convergencia del cálculo.

Función principal [solve_Q]:

```python
def solveQ(D, L, k, v, Hf, tol=1e-6):
    Q = initialGuessQ(D , L, k, v, Hf)
    Re_est = reynolds(Q, D, v)
    f = swameeJainF(Re_est, k, D)

    # enough big error value to enter the loop
    err = 100
    while(err > tol):
        V = Q / area(D)
        Re = reynolds(Q, D, v)

        f = colebrook(Re, k, D, f)
        Q_new = getQfromHf(f, D, L, Hf)
        err = abs(Q_new - Q) / Q_new

        #Update Q
        Q = Q_new

    V = Q / area(D)
    Re = reynolds(Q, D, v)

    if Re < 2300:
        regimen = "Laminar"
    else:
        regimen = "Turbulent"

    return Q, Re, f, regimen
```

Esta función realiza el **proceso iterativo completo**: estima Q, actualiza el número de Reynolds, calcula el nuevo f y repite hasta alcanzar la convergencia. Finalmente devuelve todos los resultados: caudal, velocidad, fricción, régimen y número de iteraciones.

Función process():

```python
def process():
    try:
        D = float(entry_D.get())
        L = float(entry_L.get())
        k = float(entry_k.get())
        v = float(entry_nu.get())
        Hf = float(entry_Hf.get())
        er = float(entry_Er.get())

        Q, Re, f, regime = solveQ(D, L, k, v, abs(Hf), (er/100))

        output_var.set(f"Q (volumetric flow m3/s): {Q:.6f}     \nf (friction factor): {f:.6f}     \nRe (reynolds)  {Re:.4f}     \nregime: {re

    except ValueError as e:
        messagebox.showerror("Input error", str(e))
    except Exception as e:
        messagebox.showerror("Unexpected error", str(e))
```

Esta función se ejecuta al pulsar el botón. Lee los datos introducidos por el usuario, ejecuta el cálculo con solve_Q() y muestra los resultados en pantalla. También gestiona errores si algún dato es incorrecto.

Interfaz gráfica:

```python
root = tk.Tk()
root.title("Calculo de Caudal : ")
root.geometry("800x400")
root.resizable(True, True)

tk.Label(root, text="Input data (SI)", font=("Arial", 12, "bold")).pack(pady=5)

frame_inputs = tk.Frame(root)
frame_inputs.pack(pady=5)

labels = ["D [m]", "L [m]", "k [m]", "ν [m²/s]", "Hf [m]", "Er [%]"]
defaults = ["0.03", "1000", "0.001", "0.000001", "-41.743", "2"]
entries = []

for i, (lab, val) in enumerate(zip(labels, defaults)):
    tk.Label(frame_inputs, text=lab, anchor="w", width=25).grid(row=i, column=0, pady=3, sticky="w")
    e = tk.Entry(frame_inputs, width=15)
    e.insert(0, val)
    e.grid(row=i, column=1, pady=3)
    entries.append(e)

entry_D, entry_L, entry_k, entry_nu, entry_Hf, entry_Er= entries

tk.Button(root, text="Get Q", command=process, bg="#4CAF50", fg="white", font=("Arial", 12, "bold"), width=20).pack(pady=10)


# Output
tk.Label(root, text="Output:").pack(pady=2, anchor="w")
output_var = tk.StringVar()
output = tk.Entry(root, textvariable=output_var, state="readonly", justify="center")
output.pack(pady=5, fill="x")

root.mainloop()
```

Aquí se diseña la interfaz gráfica donde el usuario introduce los datos (D, L, k, ν, Hf).
Se colocan etiquetas, campos de texto y el botón verde “Calcular Q”.
La ventana permanece activa gracias a **root.mainloop()**, que mantiene el programa en ejecución hasta que el usuario la cierra.