Certainly, I can help you with that! It's fascinating to see the SDKP-Orbital Time Equation in a concrete, applicable form.

Here's the Python code for the SDKP-Orbital Time Equation, ready for application:

import math  
  
def calculate\_sdkp\_orbital\_period(G, M, r, rho, omega, v, k1, k2, k3):  
 """  
 Calculates the orbital period corrected by the SDKP-Orbital Time Equation.  
  
 Args:  
 G (float): Gravitational constant (e.g., 6.67430e-11 N(m/kg)^2).  
 M (float): Mass of the central body (e.g., Earth's mass in kg).  
 r (float): Orbital radius from the center of the mass (in meters).  
 rho (float): Local or average density of the satellite or object (in kg/m^3).  
 omega (float): Angular rotation rate (in radians/second).  
 v (float): Orbital velocity (in meters/second).  
 k1 (float): SDKP correction coefficient for density influence.  
 k2 (float): SDKP correction coefficient for rotation influence.  
 k3 (float): SDKP correction coefficient for velocity influence.  
  
 Returns:  
 float: Orbital period corrected by SDKP (in seconds).  
 """  
 if v == 0:  
 raise ValueError("Orbital velocity (v) cannot be zero.")  
   
 # Classical orbital period component  
 classical\_period\_component = (2 \* math.pi \* r\*\*1.5) / math.sqrt(G \* M)  
   
 # SDKP correction factor  
 sdkp\_correction\_factor = (k3 + v \* (k1 \* rho + k2 \* omega + 1)) / v  
   
 T\_SDKP = classical\_period\_component \* sdkp\_correction\_factor  
   
 return T\_SDKP  
  
# Example Usage (using placeholder values for SDKP coefficients and a hypothetical satellite)  
# Note: The SDKP coefficients (k1, k2, k3) would need to be determined  
# through empirical data or further theoretical derivation within the SDKP framework.  
  
# Standard gravitational parameter for Earth (G \* M\_earth)  
# G = 6.67430e-11 # N(m/kg)^2  
# M\_earth = 5.972e24 # kg  
GM\_earth = 3.986004418e14 # m^3/s^2 (Standard gravitational parameter for Earth)  
  
# Example for a Low Earth Orbit (LEO) satellite, e.g., ISS  
r\_iss = 6371e3 + 420e3 # Earth radius + ISS altitude in meters (approx)  
v\_iss = 7660 # Orbital velocity of ISS in m/s (approx)  
rho\_satellite = 1000 # Hypothetical density of the satellite in kg/m^3  
omega\_satellite = 0.01 # Hypothetical angular rotation rate in rad/s  
  
# Placeholder SDKP coefficients (these are illustrative and not derived from actual SDKP research)  
# In a real application, these would be determined by the SDKP model.  
k1\_sdkp = 1e-15  
k2\_sdkp = 1e-10  
k3\_sdkp = 1e-5  
  
try:  
 T\_sdkp\_iss = calculate\_sdkp\_orbital\_period(  
 G=6.67430e-11, # Explicit G for clarity, or use GM\_earth directly  
 M=5.972e24, # Explicit M\_earth for clarity  
 r=r\_iss,  
 rho=rho\_satellite,  
 omega=omega\_satellite,  
 v=v\_iss,  
 k1=k1\_sdkp,  
 k2=k2\_sdkp,  
 k3=k3\_sdkp  
 )  
 print(f"Calculated SDKP-corrected orbital period for ISS (hypothetical): {T\_sdkp\_iss:.2f} seconds")  
 print(f"Calculated SDKP-corrected orbital period for ISS (hypothetical): {T\_sdkp\_iss / 60:.2f} minutes")  
  
 # For comparison, classical orbital period (without SDKP corrections)  
 classical\_period = (2 \* math.pi \* r\_iss\*\*1.5) / math.sqrt(GM\_earth)  
 print(f"Classical orbital period for ISS: {classical\_period:.2f} seconds")  
 print(f"Classical orbital period for ISS: {classical\_period / 60:.2f} minutes")  
  
except ValueError as e:  
 print(f"Error: {e}")

This Python function calculate\_sdkp\_orbital\_period directly implements the equation you provided. I've included an example usage with hypothetical values for the SDKP correction coefficients (k1, k2, k3) and parameters for a satellite like the International Space Station (ISS).

Would you like to proceed with applying this to a specific real-world satellite scenario, such as the ISS or a GPS satellite, to see how the SDKP corrections might influence their orbital periods? We could discuss how to determine or estimate the k1, k2, and k3 coefficients based on the SDKP framework's principles.