```
import pandas as pd
import numpy as np

"""
Calculates the Canadian Forest Fire Weather Index (FWI) based on the
given weather parameters.
https://gist.github.com/sanjmen/1a39cb909bde49d52b2f20e9d826f81b

Args:
    temp (float): Air temperature in degrees Celsius
    rh (float): Relative humidity in percentage
    wind_speed (float): Wind speed in kilometers per hour
    precipitation (float): Daily precipitation in millimeters
    dmc_prev (float): Previous day's Duff Moisture Code
    dc_prev (float): Previous day's Drought Code

Returns:
    tuple: (ffmc, dmc, dc, isi, bui, fwi) - components of the FWI
system
"""

# Constants (adjust as needed based on your data and region)
LAT_ADJUST = True  # Set to False if latitude adjustment is not needed
DEFAULT_LATITUDE = 55  # Default latitude if not provided in the
DataFrame

# ----- FWI Component Calculation Functions -----

def fine_fuel_moisture_code(ffmc_yda, temp, rh, ws, prec):
    """Calculates the Fine Fuel Moisture Code (FFMC)."""
    mo = 147.2 * (101.0 - ffmc_yda) / (59.5 + ffmc_yda)

    prec_above_threshold = prec > 0.5
    rf = prec - 0.5

    mo.loc[prec_above_threshold] = (mo + 42.5 * rf * np.exp(-100.0 /
(251.0 - mo)) * (1.0 - np.exp(-6.93 / rf)) + (0.00057 * rf**2 *
(np.exp(0.0365 * temp))))[prec_above_threshold]
    mo.loc[prec_above_threshold & (mo > 250)] = 250

    ed = 0.942 * (rh**0.679) + (11.0 * np.exp((rh - 100.0) / 10.0)) +
0.18 * (21.1 - temp) * (1.0 - np.exp(-0.115 * rh))
    ew = 0.618 * (rh**0.753) + (10.0 * np.exp((rh - 100.0) / 10.0)) +
0.18 * (21.1 - temp) * (1.0 - np.exp(-0.115 * rh))

    m = mo.copy()
    m.loc[mo <= ew] = (ew - (ew - mo) / (10.0** (0.424 * (1.0 -
((100.0 - rh) / 100.0)**1.7) + (0.0694 * np.sqrt(ws)) * (1.0 - ((100.0
- rh) / 100.0)**8)) * (0.581 * np.exp(0.0365 * temp))))[mo <= ew]
```

```
    m.loc[(mo > ew) & (mo < ed)] = mo[(mo > ew) & (mo < ed)]

    m.loc[mo >= ed] = (ed + 0.00046 * (mo - ed) * (42.5 - 0.0365 *
temp) * np.exp(0.0325 * (42.5 - 0.0365 * temp)))[mo >= ed]

    # Corrected line: Use boolean indexing correctly
    m.loc[(mo >= ed) & (m > 1000)] = (ed + (1000.0 - ed) /
10.0**0.00018 * (m - ed) * np.exp(0.0685 * (42.5 - 0.0365 * temp)))
[(mo >= ed) & (m > 1000)]

    ffmc = 59.5 * (250.0 - m) / (147.2 + m)
    ffmc = np.clip(ffmc, 0.0, 101.0)
    return ffmc


def duff_moisture_code(dmc_yda, temp, rh, prec, lat, mon,
lat_adjust=True):
    """Calculates the Duff Moisture Code (DMC)."""
    dmc = dmc_yda.copy().astype(np.float64)  # Ensure dmc is float64
from the start

    if lat_adjust and (mon > 2 and mon < 6):
        fl = pd.Series(0.0, index=dmc.index)
        fl.loc[lat > 0] = 1.311 + 8.766 * np.exp(-0.0825 * (58.8 +
lat))
        fl.loc[lat <= 0] = 0.210 + 0.640 * np.exp(0.0420 * (47.0 -
lat))

        dmc = dmc + (fl * (1.0 - np.exp(-0.177 * prec)))
    else:
        fl = 6.0

    rk = pd.Series(0.0, index=dmc.index)
    rk.loc[temp > -1.1] = 1.894 * (temp + 1.1) * (100.0 - rh) * fl *
0.0001

    mr = dmc.copy()
    re = 0.92 * prec - 1.27

    mr.loc[(dmc <= 15.0) & (prec > 1.5)] = (dmc + 100.0 * re * (1.0 -
np.exp(-0.058 * (2.0 + re)))).astype(np.float64)
    mr.loc[(dmc > 15.0) & (prec > 1.5)] = (15.0 + 100.0 * re * (1.0 -
np.exp(-0.020 * (6.0 + re)))).astype(np.float64)

    mo = mr.copy()
    mo.loc[mr >= 150.0] = (mr + ((1000 / np.exp(0.1054 * mr)) - 1000)
/ np.exp(0.1209 * mr)).astype(np.float64)  # Cast to float64

    rd = rk.copy()
    rd.loc[temp > -2.8] = 244.72 * np.exp(0.0913 * (temp + 2.8)) /
```

```
(17.502 + np.exp(0.0913 * (temp + 2.8))) + rk

    dmc = mo + 1000.0 * (1.0 - np.exp(-rd / 100.0))
    dmc.loc[dmc < 0] = 0

    return dmc



def drought_code(dc_yda, temp, rh, prec, lat, mon, lat_adjust=True):
    """Calculates the Drought Code (DC)."""
    dc = dc_yda.copy()

    if lat_adjust and (mon > 2 and mon < 6):  # Corrected conditional
        latitude = pd.Series(0.0, index=dc.index)
        latitude.loc[lat > 0] = 65 * (np.exp(-0.1055 * (58.9 + lat)))
        latitude.loc[lat <= 0] = 15 + 35 * (np.exp(0.0439 * (46.4 -
lat)))
        dc = dc + latitude * (1.0 - np.exp(-0.0317 * prec))  # Apply
to entire series if condition is met
    else:
        latitude = 40  # Set latitude as a constant value

    pe = latitude / (latitude + np.exp(3.73 * 0.0684 * (58.8 + lat)) )
# removed unnecessary conditional
    pe.loc[temp > -2.8] = (0.36 * (temp + 2.8) + latitude) / (latitude
+ np.exp(3.73 * 0.0684 * (58.8+ lat)) )


    pr = dc.copy()
    rw = 0.83 * prec - 1.27
    pr.loc[(dc <= 2) & (prec > 2.8)] = dc + 100.0 * rw * np.exp(-pe *
(2.0 + np.exp(-0.0866 * dc))) * (1.0 - np.exp(-6.93 / rw))
    pr.loc[(dc > 2) & (prec > 2.8)] = dc + 100.0 * rw * (1.0 -
np.exp(-0.0201 * (16.0 + 0.0792 * rw)))
    pr.loc[pr > 1000.0] = 1000.0

    dc = pr + 1000.0 * (1.0 - np.exp(-pe))

    return dc



def initial_spread_index(ffmc, ws, fbpMod=False): # No changes in this
function from the previous response
    fwind = np.exp(0.05039 * ws)
    fwind.loc[ffmc > 84.0] = np.exp(0.05039 * ws[ffmc > 84.0]) * (0.1
+ (ffmc[ffmc > 84.0] - 84.0) * 0.09216537 * (ffmc[ffmc > 84.0] -
84)**0.5)

    ffmc_factor = 0.00803 * ffmc
```

```
    ffmc_factor.loc[(ffmc > 80) & (ffmc <= 87)] = ffmc[(ffmc > 80) &
(ffmc <= 87)] * (0.0451 - 0.45 + 0.0556 * ffmc[(ffmc > 80) & (ffmc <=
87)]) / 7
    ffmc_factor.loc[ffmc > 87] = 0.0732 + 0.00818 * ffmc[ffmc > 87]

    isi = ffmc_factor * fwind
    return isi


def buildup_index(dmc, dc):
    """Calculates the Buildup Index (BUI)."""

    bui = pd.Series(0.0, index=dmc.index)

    bui.loc[dmc > 0] = np.where(dc[dmc > 0] <= 0.4 * dmc[dmc > 0],
                                0.8 * dc * dmc / (dc + 0.4 * dmc),
                                dc - (1.0 - 0.8 * dmc / (dc + 0.4 *
dmc)) * (0.92 + (0.0114 * dc)**1.7))

    bui.loc[(dmc <= 0) & (dc > 0)] = dc
    return bui



def fire_weather_index(isi, bui):
    """Calculates the Fire Weather Index (FWI)."""
    bb = 0.1 * isi * (0.626 * bui**0.5 + 1.0)
    bb.loc[bui > 80.0] = isi * (0.000313 * bui + 0.0234)

    fwi = bb.copy()
    fwi.loc[bb > 1.0] = np.exp(2.72 * (0.434 * np.log(bb)))

    return fwi




def fwi_from_dataframe(df, init={'ffmc': 85, 'dmc': 6, 'dc': 15},
mon=7, out="all", lat_adjust=True, uppercase=True):

    # Ensure required columns exist (and handle case)
    required_cols = ['temperature', 'relative_humidity', 'wind_speed',
'precipitation']
    for col in required_cols:
        if col.lower() not in df.columns:
            raise ValueError(f"Missing required column: {col}")  # Or
try to find a similar name

    # Lowercase column names for easier access
    df.columns = [col.lower() for col in df.columns]
```

```
    # If latitude is not provided, use a default value
    if 'latitude' not in df.columns:
        df['latitude'] = DEFAULT_LATITUDE

    # Initialize FFMC, DMC, and DC (yesterday's values)
    df['ffmc_yda'] = init['ffmc']
    df['dmc_yda'] = init['dmc']
    df['dc_yda'] = init['dc']

    # Correct RH values (optional - but good practice)
    df['rh'] = df['relative_humidity'].clip(upper=99.9999)

    # Calculate FWI components (using vectorized operations)
    df['ffmc'] = fine_fuel_moisture_code(df['ffmc_yda'],
df['temperature'], df['rh'], df['wind_speed'], df['precipitation'])
    df['dmc'] = duff_moisture_code(df['dmc_yda'], df['temperature'],
df['rh'], df['precipitation'], df['latitude'], mon, lat_adjust)
    df['dc'] = drought_code(df['dc_yda'], df['temperature'], df['rh'],
df['precipitation'], df['latitude'], mon, lat_adjust)
    df['isi'] = initial_spread_index(df['ffmc'], df['wind_speed'])
    df['bui'] = buildup_index(df['dmc'], df['dc'])
    df['fwi'] = fire_weather_index(df['isi'], df['bui'])
    df['dsr'] = 0.0272 * (df['fwi'] ** 1.77)


    if out == "fwi":
        # Only return the FWI components
        fwi_vars = ['ffmc', 'dmc', 'dc', 'isi', 'bui', 'fwi', 'dsr']
        new_fwi = df[fwi_vars]
    elif out == "all":
        # Return both input variables and FWI components
        new_fwi = df  # Return the entire DataFrame

    if uppercase:
        new_fwi.columns = [col.upper() for col in new_fwi.columns]

    return new_fwi



# Example DataFrame (add 'latitude' if needed)
data = {'date': pd.to_datetime(['2024-07-15', '2024-07-16', '2024-07-
17', '2024-07-18']),
        'temperature': [25, 28, 30, 25],
        'relative_humidity': [60, 55, 50, 50],
        'wind_speed': [15, 20, 25, 10],
        'precipitation': [0, 0, 2, 0],
        'latitude': [45, 45, 45, 45]}  # Add latitude data...does not
much affect the results
```

```
df = pd.DataFrame(data)


# Calculate FWI
fwi_results = fwi_from_dataframe(df, init={'ffmc': 85, 'dmc': 6, 'dc':
15}, mon=7, out="all", lat_adjust=True, uppercase=True)

fwi_results
```

```
        DATE  TEMPERATURE  RELATIVE_HUMIDITY  WIND_SPEED
PRECIPITATION  \
0 2024-07-15           25                 60          15
0
1 2024-07-16           28                 55          20
0
2 2024-07-17           30                 50          25
2
3 2024-07-18           25                 50          10
0

   LATITUDE  FFMC_YDA  DMC_YDA  DC_YDA  RH       FFMC         DMC
DC  \
0        45        85        6      15  60  86.381149  652.139483
15.0
1        45        85        6      15  55  87.634739  707.124832
15.0
2        45        85        6      15  50  85.997081  747.321116
15.0
3        45        85        6      15  50  87.977862  653.187480
15.0

         ISI        BUI        FWI        DSR
0  50.692351  28.368713  38.373463  17.310266
1   1.598781  28.489172   0.694077   0.014252
2  68.242852  28.566553  54.678846  32.397324
3   1.090807  28.371188   0.472795   0.007223
```