```
# Program moonless.py
# Counts number of days in specified period when no Moon visible between
Sunrise and Sunset.
# Simplifying assumptions
# Sunrise / Sunset is when UL altitude is -0° 50.0'
# Moonrise / Moonset is when UL altitude is -0° 34.0'
# MSL, HoE 0, Standard T & P
# Always for 0° longitude
# Latitudes above 50° excluded.

# Imports
from datetime import datetime
print ("==============================")
print("Start Run", datetime.now().strftime("%H:%M:%S"))
from skyfield.api import wgs84, load, Angle
from skyfield import almanac

# ======= Constants
Lat = 45 # Range +50 to -50
Long = 0 # Don't change - no provision for time zones
print("Lat:", Lat, "Long:", Long)
StartScanDateYear = 2006    # Earliest 1900
StartScanDateMonth = 3      # 1 - 12
StartScanDateDay = 22       # Avoid > 28 in Feb, etc
EndScanDateYear = 2024      # Latest 2053
EndScanDateMonth = 10       #  1 - 12
EndScanDateDay = 9          # Avoid > 28 in Feb, etc
ShowDetails = False         # Only set True if period less than (say) 3
months
# 2006/03/22 was start of present 18 year cycle
# 2024/10/09 will be end of present 18 year cycle

# ====== Globals
MoonlessByDayCount = 0    # Count of days when no Moon visible between
Sunrise and Sunset

# ======= Skyfield preliminaries
ts = load.timescale()
eph = load('de421.bsp')
sun = eph['Sun']
moon = eph['Moon']
observer = eph['Earth'] + wgs84.latlon(Lat, Long)

# Function to print all rise/set times
def PrintDetails(sr, ss, mr, ms):
    if ShowDetails: # flag to only show with very short scans
        print("Sunrise ", sr.utc_iso(' '))
        print("Sunset  ", ss.utc_iso(' '))
        print("Moonrise", mr.utc_iso(' '))
        print("Moonset ", ms.utc_iso(' '))
        print()
# End function PrintDetails()

# Function to analyse specified day (midnight to midnight UT)
def ThisDay(t0, t1):
    global MoonlessByDayCount
    # Get times for this day
    # Sunrise. sr (etc) are UT rise times. sry (etc) are success flags.
    sr, sry = almanac.find_risings(observer, sun, t0, t1)
    # Sunset ss
    ss, ssy = almanac.find_settings(observer, sun, t0, t1)
    # Moonrise mr
    mr, mry = almanac.find_risings(observer, moon, t0, t1)
    # Moonset ms
    ms, msy = almanac.find_settings(observer, moon, t0, t1)

    # Analyse for this day
    # ==== Case 1 ==== No Moonrise detected (but could already be risen)
    # Assumes a Moonset detected. This must be so at Lats lower than 50.
    if (str(mry) == "[]"):  # Empty mry array indicates no mr this day
        # No moonrise detected
        if (ms < sr):
            # If Moonset before Sunrise, a Moonless by Day event found
            # Rare, if ever!
            MoonlessByDayCount += 1
            return  # Essential
        return # Essential

    # ==== Case 2 ==== No Moonset detected (but could already be set)
    # Assumes a valid Moonrise detected. This must be so at Lats lower than
50.
    if (str(msy) == "[]"):  # msy will be empty array if no ms this day
        # No Moonset detected
        if (mr > ss):
            # If Moonrise after Sunset, a Moonless by Day event found
            # Rare, if ever!
            MoonlessByDayCount += 1
            return  # Essential
        return # Essential

    # ==== Case 3 ==== Moonrise during daylight
    # Can't be a Moonless by Day event. Just return.
    if (mr > sr and mr < ss):
        return

    # ==== Case 4 ==== Moonset during daylight
    # Can't be a Moonless by Day event. Just return.
    if (ms > sr and ms < ss):
        return

    # ==== Case 5 ==== Moon risen throughout daylight
    # Can't be a Moonless by Day event. Just return.
    if (mr < sr and ms > ss):
        return

    # ==== Case 6 ==== Default
    # If flow reaches here, a Moonless by Day event detected
    MoonlessByDayCount += 1
    PrintDetails(sr, ss, mr, ms)
    return
# End function ThisDay()

# Main Control Loop for every day in chosen period
if (abs(Lat) > 50):
    print("Lat", Lat, "is out of bounds")
else:
    t0 = ts.ut1(StartScanDateYear, StartScanDateMonth, StartScanDateDay) #
Start of scan day
    t1 = t0 + 1                                                          #
End of this day
    print("First date scanned:", t0.utc_strftime('%Y/%m/%d
%H:%M:%S'))                                                            #
Start of Scan
    tEnd = ts.ut1(EndScanDateYear, EndScanDateMonth, EndScanDateDay, 23,
59, 59) # End of Scan
    Days = int(tEnd - t0 + 1)

    while (t1 < tEnd):
        ThisDay(t0, t1) # Analyse each day in turn
        t0 += 1
        t1 = t0 + 1

    print("Last date scanned: ", tEnd.utc_strftime('%Y/%m/%d %H:%M:%S'))
    print(MoonlessByDayCount, "Moonless by Day events detected in period
of", Days, "days.")
    print('{0:.1f}'.format(365/Days * MoonlessByDayCount), "Average
Moonless by Day events per year.")
    print("Average can be misleading for short scans.")
    print("Ideal period is around 18.6 years or multiples thereof. (6780
days)")

print("End Run", datetime.now().strftime('%Y/%m/%d %H:%M:%S'))
print ("==============================")
print()
```

```
''' end
python moonless.py
'''
```