Get The Remaining Dates In A Year Using Datetime

2025-11-11

Table of contents

Today I Learned added on 2025-11-13, learned on 2025-11-11; edited on 2026-06-14.

Suppose you want to see how many dates there are remaining in the current year or, better yet, want a `list` of all of the remaining dates in the current year.

The Python package `datetime`, which is Python's built-in date and time manipulation package (https://docs.python.org/3/library/datetime.html), allows one to do this rather easily.

```
import datetime


# out: datetime.date(2025, 11, 13)
today = datetime.date.today()

# out: datetime.date(2025, 12, 31)
end_of_year = today.replace(month=12, day=31)

# out: 49
remaining_days_in_year = (end_of_year - today).days + 1

# out: ['2025-11-13', '2025-11-14', ..., '2025-12-31']
remaining_dates_in_year = [today +
 ↪  datetime.timedelta(days=d).strftime("%Y-%m-%d") for d in
 ↪  range(remaining_days_in_year)]
```

I became interested in this originally when I wanted to create text files for my notes for the remainder of the year. I use files entitled `Notes_For_YYYY-DD-MM` containing a template (e.g. Todos, Daily Plan, etc…) and the task boiled down to getting the remaining ISO-8601 formatted dates in the current year. The full program can be found below:

Notes File Generator

```
"""
Script for getting the remaining dates (formatted as
ISO-8601 strings) in the current year and creating
daily note files from a template.
"""

from datetime import date, timedelta
import os
import sys


def get_remaining_dates(year: int) -> list[str]:
```

```
    """
    Get all remaining dates in the specified year as
    ISO-8601 strings.

    Parameters
    ----------
    year : int
        The year to get remaining dates for.

    Returns
    -------
    list[str]
        List of date strings in YYYY-MM-DD format.
    """
    today = date.today()
    end_of_year = date(year, 12, 31)
    days_remaining = (end_of_year - today).days + 1
    return [(today + timedelta(days=i)).strftime("%Y-%m-%d") for i in
     ↪  range(days_remaining)]


def read_template(filename: str) -> str:
    """
    Read the notes template file.

    Parameters
    ----------
    filename : str
        Path to the template file.

    Returns
    -------
    str
        Template content.
    """
    try:
        with open(filename, "r") as f:
            return f.read()
    except FileNotFoundError:
        print(f"Error: Template file '{filename}' not found.")
        sys.exit(1)
    except IOError as e:
        print(f"Error reading template file: {e}")
        sys.exit(1)


def create_note_files(dates: list[str], template: str, prefix: str =
 ↪  "Notes_For_") -> int:
    """
    Create note files for each date if they don't
    already exist.
```

```
    Parameters
    ----------
    dates : list[str]
        List of date strings to create files for.
    template : str
        Template content to write to each file.
    prefix : str
        Filename prefix for note files.

    Returns
    -------
    int
        Number of files created.
    """
    created_count = 0
    for d in dates:
        filename = f"{prefix}{d}.txt"
        if not os.path.exists(filename):
            with open(filename, "w") as f:
                f.write(template)
            created_count += 1

    return created_count


if __name__ == "__main__":
    current_year = date.today().year
    remaining_dates = get_remaining_dates(current_year)
    template = read_template("NOTES.txt")
    created = create_note_files(remaining_dates, template)
    print(f"Created {created} new note files for the remaining days of
     ↪  {current_year}.")
```