tzolkin-calendar

Release 1.0.0

Release-Candidate

25.03.2021

PROJECT LINKS

tzolkin-calendar is a program and Python package (to use in your own code) that converts Gregorian dates to Maya Tzolk'in dates and vice versa. If you want to know more about the Maya calendar systems, see *Some General Links*.

Information about installing and the different ways to use tzolkin-calendar you find under **Installation and Usage**.

If you want to convert and calculate with Tzolk'in dates in your own code, see *Usage of tzolkin-calendar in Your Python Code*.

Contributing has information about how to contribute to tzolkin-calendar, by filing bug reports or feature requests, to contributing source code, to adding documentation or translations or any other way you can help tzolkin-calendar.

There are 3 interactive Jupyter Notebooks online at Binder, were you can get a fast overview of the possibilities of tzolkin-calendar. To start the interactive program, click to **Kernel -> Restart & Run All**.

- Interactive gregorian to Tzolk'in converter and vice versa: Tzolk'in Calendar Notebook
- The command line program: Tzolk'in Calendar Command Line Client
- The usage of the module tzolkin-calendar in your code: Tzolk'in Calendar Python Package

CHAPTER

ONE

SOME GENERAL LINKS

Smithsonian Museo Nacional del Indígena Americano: Viviendo El Tiempo Maya
Website of the Smithsonian National Museum of the American Indian on Mayas Living Maya Time
Online general Maya (not only Tzolk'in) calendar converter: Maya Converter of the Smithsonian NMAI and Convertidor Al Calendario Maya Smithsonian NMIA
Mayan Glyphs and Unicode: Roadmap to the SMP and the PDF Updated List of Characters for Mayan Codices

1.1 Installation

1.1.1 Prerequisites

You need Python, at least version 3.8 to be able to use tzolkin-calendar. You can download it from python.org.

To install the package, you need pip, see [Installing pip](https://pip.pypa.io/en/stable/installing/). But normally pip is installed with Python, if the version is sufficiently recent.

Windows

Download and install the latest Python from python.org. The official documentation about how to install Python on Windows: Using Python on Windows Use the options to add Python to your `PATH` and install pip, the package manager used to install Python packages like tzolkin_calendar.

Linux

Use your distribution's package management system (apt, dnf, ...) to install at least Python 3.8 and pip. If your distribution'S official packages are too old, you can install from source from python.org or search for a newer package or repository.

Mac OS X

Download and install the latest Python from python.org. The official documentation about how to install Python on OS X: Using Python on OS X

Or you can use the Homebrew package manager to install PYthon (and many other OSS packages) Homebrew Python 3.9.

1.1.2 Installation

Install the package using pip on a shell or command prompt:

```
python -m pip install tzolkin-calendar
```

Depending on your Python installation and/or OS, you may also need to call Python 3.8 (or later) using

```
python3 -m pip install tzolkin-calendar
```

or

```
python3.8 -m pip install tzolkin-calendar
```

If neither version works, the right Python executable is not in your `PATH`.

To upgrade an installed version of tzolkin-calendar to the latest version, add the argument `--upgrade`.

```
python -m pip install --upgrade tzolkin-calendar
```

More information about using pip you get at pip Quickstart.

1.2 Interactive Jupyter Notebook

You can test it online at Interactive Jupyter Notebook online at MyBinder. You need to restart the kernel first by going to the menu and selecting **Kernel**->**Restart & Run All** to get the interactive sliders and input fields.

More Information about Jupyter Notebooks at the official documentation

1.2.1 Installation

Install Jupyter Notebook and ipywidgets

```
python -m pip install notebook ipywidgets
```

If you want to be able to open the Jupyter notebook files directly, install `nbopen`.

```
python -m pip install nbopen
```

and add the extension to the list of extensions of your OS, so that you can double click the `.ipynb` files and Jupyter opens it.

On Linux:

```
python -m nbopen.install_xdg
```

On Windows:

```
python -m nbopen.install_win
```

For OS X, the installation is a bit more advanced, see nbopen.

Download the Tzolk'in calendar notebook from GitHub, using `Save As` with this link: Tzolk'in Calendar.ipynb.

Open it in Jupyter Notebook and run all cells, by going to the menu and using **Kernel -> Restart & Run All**.

You should now see something like:

Tzolk'in Calendar

For more information, visit the Github project page

Convert gregorian dates to Tzolk'in dates and search for the gregorian dates of a Tzolk'in date.

To see the Interactive Elements

Go to the menu and select **Kernel** -> **Restart & Run All** (or **Kernel** -> **Restart Kernel and Run All Cells...**)

Out[2]: The Python code of this notebook is by default hidden. To toggle on/off the raw code, click here.

Gregorian to Tzolk'in Converter

Select a date by clicking the calendar icon or writing it into the field. The Tzolk'in date of this day is displayed below the field.

Pick a Date 03/24/2021

3 Men

Search Tzolk'in Dates

Select a Tzolk'in date using the two sliders and you get a list of gregorian dates of days with this Tzolk'in date.

Day Number: 1

Day Name Imix

22.09.1957	09.06.1958	24.02.1959	11.11.1959	28.07.1960	14.04.1961	30.12.1961	16.09.1962	03.06.1963
18.02.1964	04.11.1964	22.07.1965	08.04.1966	24.12.1966	10.09.1967	27.05.1968	11.02.1969	29.10.1969
16.07.1970	02.04.1971	18.12.1971	03.09.1972	21.05.1973	05.02.1974	23.10.1974	10.07.1975	26.03.1976
11.12.1976	28.08.1977	15.05.1978	30.01.1979	17.10.1979	03.07.1980	20.03.1981	05.12.1981	22.08.1982
09.05.1983	24.01.1984	10.10.1984	27.06.1985	14.03.1986	29.11.1986	16.08.1987	02.05.1988	17.01.1989
04.10.1989	21.06.1990	08.03.1991	23.11.1991	09.08.1992	26.04.1993	11.01.1994	28.09.1994	15.06.1995
01.03.1996	16.11.1996	03.08.1997	20.04.1998	05.01.1999	22.09.1999	08.06.2000	23.02.2001	10.11.2001
28.07.2002	14.04.2003	30.12.2003	15.09.2004	02.06.2005	17.02.2006	04.11.2006	22.07.2007	07.04.2008
23.12.2008	09.09.2009	27.05.2010	11.02.2011	29.10.2011	15.07.2012	01.04.2013	17.12.2013	03.09.2014
21.05.2015	05.02.2016	22.10.2016	09.07.2017	26.03.2018	11.12.2018	28.08.2019	14.05.2020	29.01.2021
16.10.2021	03.07.2022	20.03.2023	05.12.2023	21.08.2024	08.05.2025	23.01.2026	10.10.2026	27.06.2027
13.03.2028	28.11.2028	15.08.2029	02.05.2030	17.01.2031	04.10.2031	20.06.2032	07.03.2033	22.11.2033
09.08.2034	26.04.2035	11.01.2036	27.09.2036	14.06.2037	01.03.2038	16.11.2038	03.08.2039	19.04.2040
04.01.2041	21.09.2041	08.06.2042	23.02.2043	10.11.2043	27.07.2044	13.04.2045	29.12.2045	15.09.2046
02.06.2047	17.02.2048	03.11.2048	21.07.2049	07.04.2050	23.12.2050	09.09.2051	26.05.2052	10.02.2053
28.10.2053	15.07.2054	01.04.2055	17.12.2055	02.09.2056	20.05.2057	04.02.2058	22.10.2058	09.07.2059
25.03.2060	10.12.2060	27.08.2061	14.05.2062	29.01.2063	16.10.2063	02.07.2064	19.03.2065	04.12.2065
21.08.2066	08.05.2067	23.01.2068	09.10.2068	26.06.2069	13.03.2070	28.11.2070	15.08.2071	01.05.2072
16.01.2073	03.10.2073	20.06.2074	07.03.2075	22.11.2075	08.08.2076	25.04.2077	10.01.2078	27.09.2078
14.06.2079	29.02.2080	15.11.2080	02.08.2081	19.04.2082	04.01.2083	21.09.2083	07.06.2084	22.02.2085

Search Tzolk'in Dates and Filter

Select a Tzolk'in date using the two sliders and you get a list of gregorian dates of days with this Tzolk'in date. In the field `Filter` you can add a part of a date to filter the output.

E.g. "2021" only shows results in the year 2021

Day Number: 1

Day Name Imix

Filter 01.01.

```
01.01.0119
01.01.0203
01.01.1290
01.01.1374
01.01.1458
01.01.1500
01.01.1542
01.01.1584
```

1.3 Tzolk'in Command Line Client

You can try the command line client in an interactive Jupyter Notebook at MyBinder: tzolkin_calendar Command Line Client

We start the command line client of `tzolkin-calendar` using `python -m`: beware of the underscore (_)

```
% python -m tzolkin_calendar
```

```
Gregorian "24.03.2021" is "3 Men" as Tzolk'in
```

As default, if no argument is given, the Tzolk'in date of the current day ('today' ins the 24th of March, 2021) is printed.

To get the version of `tzolkin_calendar`, use the argument `--version`

```
% python -m tzolkin_calendar --version
```

```
tzolkin-calendar 0.9.3
```

The argument `--help` displays a short usage text, we go through all options in the following parts.

```
% python -m tzolkin_calendar --help
```

```
usage: python -m tzolkin_calendar [-h] [--version] [-l LIST_LENGTH]
                                  [-s START_DATE] [-y]
                                  [DATE ...]

A Tzolk'in date converter and calculator.

Examples:

To get the Tzolk'in date of today:

 python -m tzolkin_calendar

To get the next and last gregorian dates with a Tzolk'in date of '8 Chuwen' you can␣
↪use either:

 python -m tzolkin_calendar 8 Chuwen
 python -m tzolkin_calendar 8/Chuwen
 python -m tzolkin_calendar 8.Chuwen
 python -m tzolkin_calendar 8-Chuwen
 python -m tzolkin_calendar 8 11
 python -m tzolkin_calendar 8/11
 python -m tzolkin_calendar 8.11
 python -m tzolkin_calendar 8-11

To get the Tzolk'in date of the 16th april 2016, use one of these date formats:

    python -m tzolkin_calendar 16.04.2016
    python -m tzolkin_calendar 16-04-2016
    python -m tzolkin_calendar 16 04 2016
    python -m tzolkin_calendar 2016.04.16
    python -m tzolkin_calendar 2016-04-16
    python -m tzolkin_calendar 2016/04/16
    python -m tzolkin_calendar 2016 04 16
```

(continues on next page)

(continued from previous page)

```
    python -m tzolkin_calendar 04/16/2016

positional arguments:
  DATE                  The date to parse and convert. Either a Tzolk'in date or a␣
↪gregorian date can be given. The default is the date of today.

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -l LIST_LENGTH, --list LIST_LENGTH
                        Display a list of dates with the given Tzolk'in date instead␣
↪of a single one. The length of the list is LIST_LENGTH.
  -s START_DATE, --start START_DATE
                        The start date to begin the search for the dates with the␣
↪same Tzolk'in date. The same formatting rules apply as for the main argument DATE.
  -y, --year            Print all dates of a Tzolk'in year.

See website https://github.com/Release-Candidate/tzolkin_calendar for a detailed␣
↪description.
```

1.3.1 Converting Gregorian Dates to Tzolk'in Dates

To get the Tzolk'in date of a Gregorian date use the Gregorian date as the main argument to `tzolkin_calendar`.

E.g. to get the Tzolk'in date of the 18th of May, 1974, which is "12 Akbal"

```
% python -m tzolkin_calendar 18.05.1974
```

```
Gregorian "18.05.1974" is "12 Akbal" as Tzolk'in
```

Many date format conventions are supported, any of these work (and result in the 18th of May, 1974):
DD.MM.YYYY - 18.05.1974
DD-MM-YYYY - 18-05-1974
DD MM YYYY - 18 05 1974
YYYY.MM.DD - 1974.05.18
YYYY-MM-DD - 1974-05-18
YYYY/MM/DD - 1974/05/18
YYYY MM DD - 1974 05 18
MM/DD/YYYY - 05/18/1974

```
% python -m tzolkin_calendar 05/18/1974
```

```
Gregorian "05/18/1974" is "12 Akbal" as Tzolk'in
```

1.3.2 Searching Tzolk'in Dates

To search for Gregorian Dates to a given Tzolk'in date, input the Tzolk'in date to search for.

As default the search is started today (the 24th of March, 2021). So, we search for "13 Lamat"

```
% python -m tzolkin_calendar 13 Lamat
```

```
Tzolk'in date "13 Lamat" next date is "24.08.2021", last date has been "07.12.2020"
```

The next gregorian date with a Tzolk'in date of "12 Lamat" after today (the 24th of March 2021) is the 24th od August, 2021, the last gregorian date before today has been the 7th of December 2020.

We again can use many formats to pass as Tzolk'in dates: DD NNNN - 13 Lamat
DD/NNNN - 13/Lamat
DD.NNNN - 13.Lamat
DD-NNNN - 13-Lamat

Instead of the name, we can also use the number of the day name (between 1 and 20), so instead of "Lamat" we could use the number 8. The valid formats are again (with or without leading zeroes). DD NN - 13 8 DD/NN - 13/8 DD.NN - 13.8 DD-NN - 13-8

We can also search starting at other days than today, so lets start the search at the 18th of May 1974, this is the argument to `--start`

```
% python -m tzolkin_calendar 13 Lamat --start 18.05.1974
```

```
Tzolk'in date "13 Lamat" next date is "31.08.1974", last date has been "14.12.1973"
```

Now the search returned the 31th of August, 1974 as the next and the 14th of December 1974 as the last Gregorian date with the same Tzolk'in date.

We can also search for more than one date in the future and the past, by using the argument `--list`, which is the number of Gregorian dates to return. Lets search for 5 Gregorian dates with a Tzolk'in date of "13 Lamat", starting at the 18th of May, 1974.

```
% python -m tzolkin_calendar 13 Lamat --start 18.05.1974 --list 5
```

```
Tzolk'in date "13 Lamat"
 next dates are ['31.08.1974', '18.05.1975', '02.02.1976', '19.10.1976', '06.07.1977']
 last dates have been ['14.12.1973', '29.03.1973', '12.07.1972', '26.10.1971', '08.02.
↪1971']
```

So we're getting 5 Gregorian dates after and before the 18th of May, 1974.

Without an `--start` argument, we start the search today (the 24th of March, 2021).

```
% python -m tzolkin_calendar 13 Lamat --list 5
```

```
Tzolk'in date "13 Lamat"
 next dates are ['24.08.2021', '11.05.2022', '26.01.2023', '13.10.2023', '29.06.2024']
 last dates have been ['07.12.2020', '22.03.2020', '06.07.2019', '19.10.2018', '01.02.
↪2018']
```

We can make the list as long as we want, but if the list would be too long, we ran out of the valid calendar days.

```
% python -m tzolkin_calendar 13 Lamat --list 10000
```

```
Traceback (most recent call last):
 ...
  File "./tzolkin_calendar/calculate.py", line 432, in lastTzolkin
    return starting + day_diff_delta
OverflowError: date value out of range
```

1.3.3 Print all Tzolk'in Dates in a Tzolk'in Year

To get a list of all 260 Tzolk'in dates in a Tzolk'in year, we use the argument `--year`:

```
% python -m tzolkin_calendar --year
```

```
 1 Imix 2 Ik 3 Akbal 4 Kan 5 Chikchan 6 Kimi 7 Manik 8 Lamat 9 Muluk 10 Ok 11 Chuwen␣
↪12 Eb 13 Ben
 1 Ix 2 Men 3 Kib 4 Kaban 5 Etznab 6 Kawak 7 Ajaw 8 Imix 9 Ik 10 Akbal 11 Kan 12␣
↪Chikchan 13 Kimi
 1 Manik 2 Lamat 3 Muluk 4 Ok 5 Chuwen 6 Eb 7 Ben 8 Ix 9 Men 10 Kib 11 Kaban 12␣
↪Etznab 13 Kawak
 1 Ajaw 2 Imix 3 Ik 4 Akbal 5 Kan 6 Chikchan 7 Kimi 8 Manik 9 Lamat 10 Muluk 11 Ok 12␣
↪Chuwen 13 Eb
 1 Ben 2 Ix 3 Men 4 Kib 5 Kaban 6 Etznab 7 Kawak 8 Ajaw 9 Imix 10 Ik 11 Akbal 12 Kan␣
↪13 Chikchan
 1 Kimi 2 Manik 3 Lamat 4 Muluk 5 Ok 6 Chuwen 7 Eb 8 Ben 9 Ix 10 Men 11 Kib 12 Kaban␣
↪13 Etznab
 1 Kawak 2 Ajaw 3 Imix 4 Ik 5 Akbal 6 Kan 7 Chikchan 8 Kimi 9 Manik 10 Lamat 11 Muluk␣
↪12 Ok 13 Chuwen
 1 Eb 2 Ben 3 Ix 4 Men 5 Kib 6 Kaban 7 Etznab 8 Kawak 9 Ajaw 10 Imix 11 Ik 12 Akbal␣
↪13 Kan
 1 Chikchan 2 Kimi 3 Manik 4 Lamat 5 Muluk 6 Ok 7 Chuwen 8 Eb 9 Ben 10 Ix 11 Men 12␣
↪Kib 13 Kaban
 1 Etznab 2 Kawak 3 Ajaw 4 Imix 5 Ik 6 Akbal 7 Kan 8 Chikchan 9 Kimi 10 Manik 11␣
↪Lamat 12 Muluk 13 Ok
 1 Chuwen 2 Eb 3 Ben 4 Ix 5 Men 6 Kib 7 Kaban 8 Etznab 9 Kawak 10 Ajaw 11 Imix 12 Ik␣
↪13 Akbal
 1 Kan 2 Chikchan 3 Kimi 4 Manik 5 Lamat 6 Muluk 7 Ok 8 Chuwen 9 Eb 10 Ben 11 Ix 12␣
↪Men 13 Kib
 1 Kaban 2 Etznab 3 Kawak 4 Ajaw 5 Imix 6 Ik 7 Akbal 8 Kan 9 Chikchan 10 Kimi 11␣
↪Manik 12 Lamat 13 Muluk
 1 Ok 2 Chuwen 3 Eb 4 Ben 5 Ix 6 Men 7 Kib 8 Kaban 9 Etznab 10 Kawak 11 Ajaw 12 Imix␣
↪13 Ik
 1 Akbal 2 Kan 3 Chikchan 4 Kimi 5 Manik 6 Lamat 7 Muluk 8 Ok 9 Chuwen 10 Eb 11 Ben␣
↪12 Ix 13 Men
 1 Kib 2 Kaban 3 Etznab 4 Kawak 5 Ajaw 6 Imix 7 Ik 8 Akbal 9 Kan 10 Chikchan 11 Kimi␣
↪12 Manik 13 Lamat
 1 Muluk 2 Ok 3 Chuwen 4 Eb 5 Ben 6 Ix 7 Men 8 Kib 9 Kaban 10 Etznab 11 Kawak 12 Ajaw␣
↪13 Imix
 1 Ik 2 Akbal 3 Kan 4 Chikchan 5 Kimi 6 Manik 7 Lamat 8 Muluk 9 Ok 10 Chuwen 11 Eb 12␣
↪Ben 13 Ix
 1 Men 2 Kib 3 Kaban 4 Etznab 5 Kawak 6 Ajaw 7 Imix 8 Ik 9 Akbal 10 Kan 11 Chikchan␣
↪12 Kimi 13 Manik
 1 Lamat 2 Muluk 3 Ok 4 Chuwen 5 Eb 6 Ben 7 Ix 8 Men 9 Kib 10 Kaban 11 Etznab 12␣
↪Kawak 13 Ajaw
```

(continues on next page)

(continued from previous page)

```
Gregorian "24.03.2021" is "3 Men" as Tzolk'in
```

1.4 License

tzolkin-calendar is licensed under the MIT license:

```
MIT License

Copyright (c) 2021 Roland Csaszar

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

1.5 Usage of tzolkin-calendar in Your Python Code

You can try the tzolkin_calendar package interactively using the online Jupyter Notebook at MyBinder: tzolkin_calendar module

1.5.1 Import the Module

To use the Tzolk'in date package, import `tzolkin_calender` (with an underscore _).

A short check to see if it is working, is to print the version of `tzolkin-calendar`, that's the constant `tzolkin_calendar.VERSION`.

```
# Import the package.
import tzolkin_calendar

# Check, if it is working.
tzolkin_calendar.VERSION
```

```
'0.9.3'
```

1.5.2 The Tzolk'in Date Class `Tzolkin`

The Tzolk'in date class resides in the module `tzolkin_calendar.tzolkin`, and is named `Tzolkin`.

So, import that module:

```
# Import the module tzolkin that contains `Tzolkin`.
from tzolkin_calendar import tzolkin
```

Convert Gregorian Dates to Tzolk'in Dates

To get the Tzolk'in date of today, call the static method `fromToday`. This returns a `Tzolkin` instance holding the Tzolk'in date of today ('today' is the 22nd of March, 2021, with a Tzolk'in date of '1 Ben')

```
tzolkin.Tzolkin.fromToday()
```

```
'3 Men'
```

You can generate a `Tzolkin` instance from any gregorian date using the `datetime.date` class or a date string.

So, first import the datetime module:

```
import datetime
```

And then use the 3 possibilities to set the Tzolk'in date from a gregorian date. Or, in other words, to convert a gregorian date to a Tzolk'in date.

1. from a datetime.date instance using `Tzolkin.fromDate`. We use the method `fromisoformat` to set the gregorian date to the 22nd of March 2021 with the date string '2021-03-22'. The Tzolk'in date of the 22nd of March 2021 is '1 Ben'.

```
gregorian_date = datetime.date.fromisoformat("2021-03-22")

tzolkin.Tzolkin.fromDate(gregorian_date)
```

```
'1 Ben'
```

2. from an ISO date string using `Tzolkin.fromIsoFormat`. We set the Tzolk'in date to the 22nd of March 2021 with the ISO date string '2021-03-22'. The Tzolk'in date of the 22nd of March 2021 is '1 Ben'.

```
tzolkin.Tzolkin.fromIsoFormat("2021-03-22")
```

```
'1 Ben'
```

3. from an arbitrary date string using `Tzolkin.fromDateString`. We set the Tzolk'in date to the 22nd of March 2021 with the date string '22=03*2021' and the format string `fmt` '%d=%m*%Y'. The Tzolk'in date of the 22nd of March 2021 is '1 Ben'.

```
tzolkin.Tzolkin.fromDateString("22=03*2021", fmt="%d=%m*%Y")
```

```
'1 Ben'
```

Set Tzolk'in Dates

You can set the `Tzolkin` instance to a Tzolk'in Date using it's constructor. The constructor takes the Tzolk'in day number (between 1 and 13 including 1 and 13) and either a Tzolk'in day name or the number of the Tzolk'in day name (between 1 and 20 , including 1 and 20).

To get a dictionary of Tzolk'in day names and numbers, look at `tzolkin.day_names`.

```
tzolkin.day_names
```

```
{1: 'Imix', 2: 'Ik', 3: 'Akbal', 4: 'Kan', 5: 'Chikchan', 6: 'Kimi', 7: 'Manik',
 8: 'Lamat', 9: 'Muluk', 10: 'Ok', 11: 'Chuwen', 12: 'Eb', 13: 'Ben', 14: 'Ix',
 15: 'Men', 16: 'Kib', 17: 'Kaban', 18: 'Etznab', 19: 'Kawak', 20: 'Ajaw'}
```

If we want to set a Tzolk'in day of '8 Kaban', we can either pass the day number 8 and day name Kaban to the constructor, or the day number 8 and the day name number 17.

```
tzolkin.Tzolkin(number=8, name_str="Kaban")
```

```
'8 Kaban'
```

```
tzolkin.Tzolkin(number=8, name_number=17)
```

```
'8 Kaban'
```

If we pass an invalid number (not in [1, 13]) or name to the constructor, we get a `TzolkinException`.

```
tzolkin.Tzolkin(number=53, name_number=17)

    TzolkinException: number 53 is not a valid Tzolkin day number, not between 1 and␣
↪13 (including 1 and 13)

tzolkin.Tzolkin(number=3, name_str="Hugo")

    TzolkinException: string "Hugo" is not a valid Tzolkin day name, one of: dict_
↪values(['Imix', 'Ik', 'Akbal', 'Kan', 'Chikchan', 'Kimi', 'Manik', 'Lamat', 'Muluk',
↪ 'Ok', 'Chuwen', 'Eb', 'Ben', 'Ix', 'Men', 'Kib', 'Kaban', 'Etznab', 'Kawak', 'Ajaw
↪'])

tzolkin.Tzolkin(number=3, name_number=-5)

    TzolkinException: -5 is not a valid Tzolkin day name number, it must be between 1␣
↪and 20 (including 1 and 20)
```

These Tzolk'in day numbers and names can be accessed using the methods `getDayNumber`, `getDayName` and `getDayNameNumber`.

```
# Set the Tzolk'in date to '12 Kimi'.
tzolkin_date = tzolkin.Tzolkin(number=12, name_str="Kimi")

tzolkin_date.getDayNumber()
```

```
12
```

```
tzolkin_date.getDayName()
```

```
'Kimi'
```

```
tzolkin_date.getDayNameNumber()
```

```
6
```

To get the number of Tzolk'in day in the Tzolk'in year of 260 days, there is `getTzolkinYearDay`. For example '12 Kimi' is the 246. day (of 260 days) of the Tzolk'in year

```
tzolkin_date.getTzolkinYearDay()
```

```
246
```

To parse a Tzolk'in day name that isn't exactly like the ones in `tzolkin.day_names` there is the method `Tzolkin.parseTzolkinName`, that ignores upper- and lowercase and all non-alphanumeric and non-ascii characters.

```
day_number = tzolkin.Tzolkin.parseTzolkinName("EtZ`nAB")
if day_number != 0:
    tzolkin_name = tzolkin_calendar.day_names[day_number]
tzolkin_name
```

```
'Etznab'
```

All 260 Tzolk'in days of a Tzolk'in year we can get as a list of strings from the static method `getTzolkinCalendar`.

```
tzolkin.Tzolkin.getTzolkinCalendar()
```

```
['1 Imix', '2 Ik', '3 Akbal', '4 Kan', '5 Chikchan', '6 Kimi', '7 Manik', '8 Lamat',
↪'9 Muluk', '10 Ok', '11 Chuwen', '12 Eb', '13 Ben',
 '1 Ix', '2 Men', '3 Kib', '4 Kaban', '5 Etznab', '6 Kawak', '7 Ajaw', '8 Imix', '9 Ik
↪', '10 Akbal', '11 Kan', '12 Chikchan', '13 Kimi',
 '1 Manik', '2 Lamat', '3 Muluk', '4 Ok', '5 Chuwen', '6 Eb', '7 Ben', '8 Ix', '9 Men
↪', '10 Kib', '11 Kaban', '12 Etznab', '13 Kawak',
 '1 Ajaw', '2 Imix', '3 Ik', '4 Akbal', '5 Kan', '6 Chikchan', '7 Kimi', '8 Manik',
↪'9 Lamat', '10 Muluk', '11 Ok', '12 Chuwen', '13 Eb',
 '1 Ben', '2 Ix', '3 Men', '4 Kib', '5 Kaban', '6 Etznab', '7 Kawak', '8 Ajaw', '9␣
↪Imix', '10 Ik', '11 Akbal', '12 Kan', '13 Chikchan',
 '1 Kimi', '2 Manik', '3 Lamat', '4 Muluk', '5 Ok', '6 Chuwen', '7 Eb', '8 Ben', '9 Ix
↪', '10 Men', '11 Kib', '12 Kaban', '13 Etznab',
 '1 Kawak', '2 Ajaw', '3 Imix', '4 Ik', '5 Akbal', '6 Kan', '7 Chikchan', '8 Kimi',
↪'9 Manik', '10 Lamat', '11 Muluk', '12 Ok', '13 Chuwen',
 '1 Eb', '2 Ben', '3 Ix', '4 Men', '5 Kib', '6 Kaban', '7 Etznab', '8 Kawak', '9 Ajaw
↪', '10 Imix', '11 Ik', '12 Akbal', '13 Kan',
 '1 Chikchan', '2 Kimi', '3 Manik', '4 Lamat', '5 Muluk', '6 Ok', '7 Chuwen', '8 Eb',
↪'9 Ben', '10 Ix', '11 Men', '12 Kib', '13 Kaban',
 '1 Etznab', '2 Kawak', '3 Ajaw', '4 Imix', '5 Ik', '6 Akbal', '7 Kan', '8 Chikchan',
↪'9 Kimi', '10 Manik', '11 Lamat', '12 Muluk', '13 Ok',
 '1 Chuwen', '2 Eb', '3 Ben', '4 Ix', '5 Men', '6 Kib', '7 Kaban', '8 Etznab', '9␣
↪Kawak', '10 Ajaw', '11 Imix', '12 Ik', '13 Akbal',
 '1 Kan', '2 Chikchan', '3 Kimi', '4 Manik', '5 Lamat', '6 Muluk', '7 Ok', '8 Chuwen',
↪ '9 Eb', '10 Ben', '11 Ix', '12 Men', '13 Kib',
 '1 Kaban', '2 Etznab', '3 Kawak', '4 Ajaw', '5 Imix', '6 Ik', '7 Akbal', '8 Kan', '9␣
↪Chikchan', '10 Kimi', '11 Manik', '12 Lamat', '13 Muluk',
```

(continues on next page)

(continued from previous page)

```
 '1 Ok', '2 Chuwen', '3 Eb', '4 Ben', '5 Ix', '6 Men', '7 Kib', '8 Kaban', '9 Etznab',
↪ '10 Kawak', '11 Ajaw', '12 Imix', '13 Ik',
 '1 Akbal', '2 Kan', '3 Chikchan', '4 Kimi', '5 Manik', '6 Lamat', '7 Muluk', '8 Ok',
↪'9 Chuwen', '10 Eb', '11 Ben', '12 Ix', '13 Men',
 '1 Kib', '2 Kaban', '3 Etznab', '4 Kawak', '5 Ajaw', '6 Imix', '7 Ik', '8 Akbal', '9␣
↪Kan', '10 Chikchan', '11 Kimi', '12 Manik', '13 Lamat',
 '1 Muluk', '2 Ok', '3 Chuwen', '4 Eb', '5 Ben', '6 Ix', '7 Men', '8 Kib', '9 Kaban',
↪'10 Etznab', '11 Kawak', '12 Ajaw', '13 Imix',
 '1 Ik', '2 Akbal', '3 Kan', '4 Chikchan', '5 Kimi', '6 Manik', '7 Lamat', '8 Muluk',
↪'9 Ok', '10 Chuwen', '11 Eb', '12 Ben', '13 Ix',
 '1 Men', '2 Kib', '3 Kaban', '4 Etznab', '5 Kawak', '6 Ajaw', '7 Imix', '8 Ik', '9␣
↪Akbal', '10 Kan', '11 Chikchan', '12 Kimi', '13 Manik',
 '1 Lamat', '2 Muluk', '3 Ok', '4 Chuwen', '5 Eb', '6 Ben', '7 Ix', '8 Men', '9 Kib',
↪'10 Kaban', '11 Etznab', '12 Kawak', '13 Ajaw']
```

Search Gregorian Dates to a given Tzolk'in Date

We can search for the next (forward in time) or last (backwards in time) day with the same Tzolk'in date using the methods `getNextDate` and `getLastDate`. Both methods return a `datetime.date` object.

When searching for the next gregorian date that has the Tzolk'in date '7 Kawak', we get the 28th of March, 2021 - because we started searching 'today', which is the 22nd of March 2021.

```
# Set the Tzolk'in date to search for to '7 Kawak'.
tzolkin_date = tzolkin.Tzolkin(number=7, name_str="Kawak")

tzolkin_date.getNextDate()
```

```
datetime.date(2021, 3, 28)
```

When searching for the last gregorian date that has the Tzolk'in date '7 Kawak', we get the 11th of July, 2020 - because we started searching 'today', which is the 22nd of March 2021.

```
tzolkin_date.getLastDate()
```

```
datetime.date(2020, 7, 11)
```

Both methods, `getNextDate` and `getLastDate` take an optional argument `start_date`, which is the gregorian date to start the search. If no `start_date` is given, 'today' is used as the start date.

So now we search again for '7 Kawak' in both directions, but this time we start at the 10th of July, 2020.

```
# Set the Tzolk'in date to search for to '7 Kawak'.
tzolkin_date = tzolkin.Tzolkin(number=7, name_str="Kawak")

# Set the start date of the search to the 10th of July, 2020.
start_search = datetime.date.fromisoformat("2020-07-10")

start_search.isoformat()
```

```
'2020-07-10'
```

For the next day with a Tzolk'in date of '7 Kawak' we now get the 11th of July, 2020.

```
tzolkin_date.getNextDate(start_date=start_search)
```

```
datetime.date(2020, 7, 11)
```

For the last day before our start date with a Tzolk'in date of '7 Kawak' we now get the 25th of October, 2019.

```
tzolkin_date.getLastDate(start_date=start_search)
```

```
datetime.date(2019, 10, 25)
```

To get a list of `datetime.date` dates with the same Tzolk'in date, we can use the methods `getNextDateList` and `getLastDateList`.

Again, we can set the argument `start_date` to a gregorian date to start the search or not set it to start the search today. The number of elements in the returned list is set using the parameter `list_size`, which defaults to 50.

```
# Set the Tzolk'in date to search for to '7 Kawak'.
tzolkin_date = tzolkin.Tzolkin(number=7, name_str="Kawak")
```

Let's start the search for dates with a Tzolk'in date of ' 7 Kawak' today, the 22nd of March 2021, and set the list size to 9 elements:

```
tzolkin_date.getNextDateList(list_size=9)
```

```
[datetime.date(2021, 3, 28),
 datetime.date(2021, 12, 13),
 datetime.date(2022, 8, 30),
 datetime.date(2023, 5, 17),
 datetime.date(2024, 2, 1),
 datetime.date(2024, 10, 18),
 datetime.date(2025, 7, 5),
 datetime.date(2026, 3, 22),
 datetime.date(2026, 12, 7)]
```

Now start searching for '7 Kawak' on the 29th of March, 2021 and set the returned list size to 5.

```
# Set the start date of the search to the 29th of March, 2021.
start_search = datetime.date.fromisoformat("2021-03-29")

tzolkin_date.getLastDateList(start_date=start_search, list_size=5)
```

```
[datetime.date(2021, 3, 28),
 datetime.date(2020, 7, 11),
 datetime.date(2019, 10, 25),
 datetime.date(2019, 2, 7),
 datetime.date(2018, 5, 23)]
```

Calculations using Tzolk'in Dates

There are 4 methods to get the difference in days between two Tzolk'in dates and to add (or subtract) days from a Tzolk'in date: `addDays`, `addTimedelta`, `getDayDiff` and `getDayTimedelta`.

Lets start with a Tzolk'in date of '6 Muluk'.

```
tzolkin.Tzolkin(number=6, name_str="Muluk")
```

```
'6 Muluk'
```

Add 6 days to it, and we get a Tzolk'in date of '12 Men'.

```
tzolkin.Tzolkin(number=6, name_str="Muluk").addDays(6)
```

```
'12 Men'
```

Instead of using ints, we can also add and subtract `datetime.timedelta` objects. Now subtract 6 days from '12 Men' - we get '6 Muluk'.

```
to_subtract = datetime.timedelta(days=-6)

tzolkin.Tzolkin(number=12, name_str="Men").addTimedelta(to_subtract)
```

```
'6 Muluk'
```

To get the difference between two Tzolk'in dates there exist the Methods `getDayDiff` and `getDayTimedelta`.

Lets calculate the difference in days between '6 Muluk' and '12 Men'.

```
# Set start_tzolkin to '6 Muluk'.
start_tzolkin = tzolkin.Tzolkin(number=6, name_str="Muluk")

# Set end_tzolkin to '12 Men'.
end_tzolkin = tzolkin.Tzolkin(number=12, name_str="Men")

start_tzolkin.getDayDiff(end_tzolkin)
```

```
6
```

And using `getDayTimedelta`, which returns a `datetime.timedelta` object.

```
start_tzolkin.getDayTimedelta(end_tzolkin)
```

```
datetime.timedelta(days=6)
```

What happens, if we calculate the difference between '12 Men' and '6 Muluk'?

```
end_tzolkin.getDayDiff(start_tzolkin)
```

```
254
```

We get 254 days, not -6. That's because the difference is always calculated forward in time. If you want to get negative days or the shortest possible time difference, subtract 260 from the the result (the number of days in a Tzolk'in year). As soon as the difference in days is greater than 130, to minimum time distance in days 'is negative'.

```
day_diff = end_tzolkin.getDayDiff(start_tzolkin)

if day_diff > 130:
    day_diff = 260 - day_diff
day_diff
```

```
6
```

Or you can use the minimum of `result` and `|result - 260|` that is `abs(result - 260)`.

```
day_diff = end_tzolkin.getDayDiff(start_tzolkin)
shortest_diff = min(day_diff, abs(day_diff - 260))
shortest_diff
```

```
6
```

1.6 tzolkin_calendar Package

This is the source code reference documentation, autogenerated from tzolkin-calendar's source code.

1.6.1 Submodules

1.6.2 tzolkin_calendar.tzolkin module

1.6.3 tzolkin_calendar.calculate module

This modules holds functions needed to calculate with Tzolkin calendar dates and convert them to and from gregorian dates.

Example:

```
>>> import datetime
>>> import tzolkin_calendar.calculate
>>> tzolkin_calendar.calculate.gregorian2tzolkin(datetime.datetime.strptime("23.05.
↪2014", "%d.%m.%Y"))
2 Etznab ()
```

```
>>> import datetime
>>> import tzolkin_calendar.calculate
>>> tzolkin_calendar.calculate.nextTzolkin(tzolkin_calendar.TzolkinDate(name=3,␣
↪number=5))
datetime.date(2021, 4, 21)
```

```
>>> import datetime
>>> import tzolkin_calendar.calculate
>>> tzolkin_calendar.calculate.lastTzolkin(tzolkin_calendar.TzolkinDate(name=17,␣
↪number=3))
datetime.date(2021, 1, 5)
```

```
>>> import datetime
>>> import tzolkin_calendar.calculate
>>> tzolkin_calendar.calculate.tzolkin2gregorian(tzolkin_calendar.TzolkinDate(name="2
↪", number="7"),start=datetime.date.today())
[datetime.date(2021, 10, 15), datetime.date(2022, 7, 2), datetime.date(2023, 3, 19),␣
↪datetime.date(2023, 12, 4),
```

```
>>> import datetime
>>> import tzolkin_calendar.calculate
>>> tzolkin_calendar.calculate.tzolkin2gregorian(tzolkin_calendar.TzolkinDate(name="2
↪", number="7"), forward=False, start=datetime.date.today())
[datetime.date(2021, 10, 15), datetime.date(2021, 1, 28), datetime.date(2020, 5, 13),␣
↪datetime.date(2019, 8, 27),
```

calculateTzolkinName (*start_name: int*, *to_add: int*) → int

Return the Tzolkin name *to_add* days after *start_name*. Add or subtracts the given integer to the index of the Tzolkin name and return the index of the new name. Adds *to_add* to the name index *start_name* and takes the value modulo 20. If the result would be 0, return 20 instead.

Parameters

- **start_name** (*int*) – The index of the name to add days to.
- **to_add** (*int*) – The number of days to add to the Tzolkin name.

Returns The index of the resulting Tzolkin name, *to_add* days after *start_name*.

Return type int

calculateTzolkinNumber (*start_number: int*, *to_add: int*) → int

Return the Tzolkin number *to_add* days after *start_number*. Add or subtracts the given integer to the Tzolkin number and return the new number. Adds *to_add* to the number *start_name* and takes the value modulo 13. If the result would be 0, return 13 instead.

Parameters

- **start_number** (*int*) – The number to add the days to.
- **to_add** (*int*) – The number of days to add to the Tzolkin number.

Returns The resulting number *to_add* days after *start_number*.

Return type int

getTzolkinDay (*tzolkin: tzolkin_calendar.TzolkinDate*) → int

Return the day number in the Tzolkin year, in the interval [1,260] (including both 1 and 260). That is, the index of the given Tzolkin date in the 260 day Tzolkin year. *1 Imix* yields 1 (the first day of the year), *13 Ajaw* yields 260, the last day of the Tzolkin year. If the given date *tzolkin* does not exist, *0* is returned.

Parameters **tzolkin** (*TzolkinDate*) – The Tzolkin date to get the day in the year of.

Returns

The day of the given date in the Tzolkin year, a positive integer between and including 1 and 260. If the given date does not exist, *0* is returned.

Return type int

getTzolkinDiff (*start: tzolkin_calendar.TzolkinDate*, *end: tzolkin_calendar.TzolkinDate*) → int

Return the difference in days between the two given Tzolkin dates. No negative differences are returned, but the number of days to reach the *end* date if starting from *start*. If *start* is earlier than *end* the difference is *start* - *end*. If *end* is before *start*, 260 - *start* + *end* (same as 260 - (*end* - *start*)) is returned.

Example

getTzolkinDiff returns 12 for *start* = 4 Manik and *end* = 3 Kawak

```
>>> getTzolkinDiff(
        start=tzolkin_calendar.TzolkinDate(number=4, name=7),
        end=tzolkin_calendar.TzolkinDate(number=3, name=19),
    ) == 12
```

getTzolkinDiff returns 250 for *start* = 8 Chuwen and *end* = 11 Imix

```
>>> getTzolkinDiff(
        start=tzolkin_calendar.TzolkinDate(number=8, name=11),
        end=tzolkin_calendar.TzolkinDate(number=11, name=1),
    ) == 250
```

Parameters

- **start** (*TzolkinDate*) – The Tzolkin date to start the calculation from.
- **end** (*TzolkinDate*) – The Tzolkin date to calculate the time difference in days to.

Returns The number of days between the two given dates. Never negative (0 if *start* and *end* are the same day).

Return type int

gregorian2tzolkin(*date: datetime.date*) → tzolkin_calendar.TzolkinDate

Return the Tzolkin date of the given gregorian date.

Parameters **date** (*datetime.date*) – The gregorian date to convert to Tzolkin.

Returns The Tzolkin date of the given day *date*.

Return type TzolkinDate

lastTzolkin(*tzolkin: tzolkin_calendar.TzolkinDate*, *starting: datetime.date = datetime.date(2021, 3, 25)*) → datetime.date

Return the last gregorian date before *starting*, that has a Tzolkin date of *tzolkin*. Search backwards in time for a day with the Tzolkin date *tzolkin*.

Parameters

- **tzolkin** (*TzolkinDate*) – The Tzolkin date to search for.
- **starting** (*datetime.date, optional*) – The date to start the search. Defaults to *datetime.date.today()*.

Returns

The last gregorian date with the given Tzolkin date *tzolkin* before *starting*.

Return type datetime.date

makeLookUpTable() → Dict[int, tzolkin_calendar.TzolkinDate]

Return a dictionary holding all *TzolkinDate* instances of a tzolkin year. The tzolkin year consists of all combinations of *day_names* and *ay_numbers*, *day_numbers* are the numbers from 1 to 13 and *day_names* the names from 'Imix' to 'Ajaw'. So a Tzolkin year is: 1 Imix, 2 Ik', 3 Ak'b'al, ... and finishes at 12 Kawak and finally 13 Ajaw.

Returns

The dictionary of all tzolkin date combinations in a tzolkin year (of 260 days).

Return type Dict[int, TzolkinDate]

nextTzolkin(*tzolkin: tzolkin_calendar.TzolkinDate*, *starting: datetime.date = datetime.date(2021, 3, 25)*) → datetime.date

Return the next gregorian date after *starting*, that has a Tzolkin date of *tzolkin*. Search forward in time for a day with Tzolkin date *tzolkin*.

Parameters

- **tzolkin** (*TzolkinDate*) – The Tzolkin date to search for.
- **starting** (*datetime.date, optional*) – The date to start the search. Defaults to *datetime.date.today()*.

Returns

The next gregorian date with the given Tzolkin date *tzolkin* after *starting*.

Return type datetime.date

parseTzolkinName(*name_str: str*) → int

Parse the given string to get a Tzolkin day name. Ignores lower- and uppercase, ignores all non-alphanumberic characters.

Returns 0 if no name has been found

Parameters **name_str** (*str*) – The string to parse to get a Tzolkin day name.

Returns The number of the found Tzolkin day name. 0 on errors.

Return type int

tzolkin2gregorian(*tzolkin: tzolkin_calendar.TzolkinDate*, *start: datetime.date*, *num_results: int = 100*, *forward: bool = True*) → List[datetime.date]

Return a list of dates having the same Tzolkin date as the given date *tzolkin*.

If *num_results* is smaller than 1, an empty list is returned.

Parameters

- **tzolkin** (*TzolkinDate*) – The Tzolkin date to search for.
- **start** (*datetime.date*) – The gregorian date to start the search from.
- **num_results** (*int, optional*) – The number of results to return. If this is < 1, an empty list is returned. Defaults to 100.
- **forward** (*bool, optional*) – The direction in time to search. Either forward (if
- **is True) or backwards** (*forward*) –

Returns

The list of gregorian dates having the same Tzolkin date as *tzolkin*. The number of elements of this list is *num_results*.

Return type List[datetime.date]

1.6.4 tzolkin_calendar.main module

1.7 Contributing

Any help is welcome!

If you encounter a problem using tzolkin-calendar, a task it not as easy as you'd like it to be or you'd like something added to it: open an issue at GitHub.

1.7.1 Report Issues (Bugs and Feature Requests)

File a bug report at Github.

Add a feature request at Github.

1.7.2 Forking the Repository

If you'd like to contribute directly, e.g. better the documentation, add another language or write some source code: fork tzolkin-calendar by clicking the `Fork`-button in the upper right corner of the GitHub project website. Check out your fork of tzolkin-calendar using the URL from the `Code`-button of your fork on Github. The URL should be something like **github.com/YOUR_USERNAME/tzolkin-calendar.git**.

Details about how to fork a repository on Github are here.

1.7.3 Setting the Development Environment

All needed packages to develop tzolkin-calendar are installed in a virtual environment using `pipenv`, so your system-wide Python installation isn't affected by it.

First, install pipenv if you don't already have it installed:

```
python -m pip install --upgrade pipenv
```

and install all needed packages to develop tzolkin-calender:

```
cd tzolkin-calendar
python -m pipenv install --dev
```

That command installs all packages in `Pipfile`/`Pipfile.lock` in the directory `tzolkin-calender`, the root directory of tzolkin-calendar.

More information about pipenv can be found at Pipenv.

Make your changes, push them to your forked repository and make a pull-request (e.g. using the *Pull request*-button above and right of GitHubs source file view).

See [GitHub on Pull-Requests](https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/proposing-changes-to-your-work-with-pull-requests)

1.7.4 Github Documentation on Collaborating with Issues and Pull Requests

See GitHub's documentation about how to contribute for details: Contributing at Github.

1.7.5 Common Tasks Developing tzolkin-calendar

Jupyter Notebooks

The 3 Jupyter Notebooks are located in the project root directory *tzolkin_calendar*, named `Tzolk'in Calendar.ipynb`, `Tzolk'in Command Line.ipynb` and `Tzolk'in Calender Python Module.ipynb`

Changing and Generating Documentation

All files to generate the Sphinx documentation for Read The Docs are located in the directory `tzolkin_calendar/tzolkin_calendar/doc/source`.

- conf.py ... the Sphinx documentation
- *.rst ... the reStructuredText source file to generate the HTML documentation.

After changing any of these files, you need to run `sphinx-build` using the `Makefile` or the `make` script.

On Windows use the batch file `make.bat` with the argument `html`:

```
cd tzolkin_calendar\tzolkin_calendar\doc\
make html
```

Anywhere else use `make` with the argument `html`:

```
cd tzolkin_calendar/tzolkin_calendar/doc/
make html
```

After that, the new HTML documentation should have been generated in `tzolkin_calendar/tzolkin_calendar/doc/html` and you can open `tzolkin_calendar/tzolkin_calendar/doc/html/index.html` in a browser to see it.

GitHub Documentation

The Markdown documentation for GitHub are the files `README.md` and `CHANGELOG.md` in the project root directory *tzolkin_calendar*.

Python Source Code

The Python source code is located in the directory `tzolkin_calendar/tzolkin_calendar/`.

- __main__.py ... Just a wrapper to call `main()` in the file `main.py`
- __init__.py ... Some constants, like `VERSION`, which holds the package's version string.
- tzolkin.py ... The Tzolk'in date class `Tzolkin`, the main interface of the package
- calculate.py ... Function that do the actual date calculations are are used by the Tzolk'in date class.
- main.py ... Main entry point of the command line client, when the module is executed instead of imported.
- commandline.py ... The command line parsing for the command line client.

See also tzolkin_calendar.

Tests

All test code is located in the directory `tzolkin-calendar/tests/`. Pytest is used as test runner.

- test_tzolkin.py ... Tests of the Tzolk'in date class `Tzolkin`
- test_calender.py ... Tests of the Tzolk'in date calculation functions in `calculate.py`
- test_main.py ... Tests of the command line client, files `main.py` and `commandline.py`
- __init__.py ... some program or external site needs that(?)

To run the tests, go to the root directory `tzolkin-calendar` (not `tzolkin_calendar`) and call Pytest.

```
pytest --no-cov
```

which runs the tests without coverage analysis.

To see statistics of Hypothesis, add the argument `--hypothesis-show-statistics`

```
pytest --hypothesis-show-statistics --no-cov
```

To speed up the execution, use more than one process, the argument to `-n` is the number processes to use.

```
pytest --hypothesis-show-statistics --no-cov -n 24
```

uses 24 processes to run the tests.

There are two scripts, `run_test.bat` and `run_tests.sh` that you can use to run the tests.

```
run_tests
```

or

```
./run_tests.sh
```

Local Source Code Linters

To check the Python sources and tests using static code checkers and fix import order and the formatting, call the script `run_local_linters.sh` or `run_local_linters.bat`

```
run_local_linters
```

or

```
./run_local_linters.sh
```

GitHub Workflows/Actions

The GitHub Workflows/Actions are located in `tzolkin-calendar/.github/workflows/`

- bandit.yml ... Run Bandit, static code checker
- black.yml ... Run Black, Python code formatter
- create_pip.yml ... Create the tzolkin-calendar pip package and upload it to PyPI
- flake8.yml ... Run Flake8, static code checker
- linux.yml ... Run the command line client under Linux, from src and the package
- linux_test.yml ... Run the tests under Linux, from src and the package
- osx.yml ... Run the command line client under OS X, from src and the package
- osx_test.yml ... Run the tests under OS X, from src and the package
- pycodestyle.yml ... Run PyCodeStyle, static code checker
- pydocstyle.yml ... Run PyDocStyle, static code checker
- pyflakes.yml ... Run PyFlakes, static code checker
- windows.yml ... Run the command line client under Windows, from src and the package
- windows_test.yml ... Run the tests under Windows, from src and the package

1.8 Index

This is a placeholder to include genindex.html in the toctree.

1.9 Python Module Index

This is a placeholder to include py-modindex.html in the toctree.

PYTHON MODULE INDEX

t

INDEX