Al Adhan Python API

Release 6.1.0 Alpha

Khaled Mahmoud

Jan 03, 2023

TABLE OF CONTENTS:

This Python module calculates accurate Islamic prayer times for any location. It uses the aladhan.com prayer times API and converts the API responses into Python objects. No API key or registration is required. This module simplifies the process of using the aladhan.com prayer times API in Python projects.

Note: This module is still in development and under alpha version, so it may have some bugs, and the API may have major changes in the future.

CHAPTER

ONE

INSTALLATION

You can install the latest version of the package from PyPI:

```
pip install aladhan-api
```

Or if you have an older version of the package installed, you can upgrade it using:

```
pip install aladhan-api --upgrade
```

Note: The current pip domain `aladhan-api` will be changed in the first stable release to `al-adhan` *which is already taken*.

CHAPTER

TWO

CHOOSING A LOCATION

Before using the module, you must specify a location to get prayer times for. The API does not automatically determine your location, so you must define it by using a location object. There are three types of location objects:

2.1 aladhan.Coordinates

Coordinates (latitude and longitude) are like X and Y coordinates on a map, you can see your own coordinates by clicking here.

Arguments:

- `latitude (float)`: The latitude of the location (required)
- `longitude (float)`: The longitude of the location (required)

Raises:

- `InvalidLocationException`: If the latitude or longitude is invalid.

Example:

```
import aladhan

location = aladhan.Coordinates(51.507351, -0.127758)
```

2.2 aladhan.City

City is a location object that is defined by a city name and a country, state is optional.

Arguments:

- `city (str)`: The name of the city (required)
- `country (str)`: The name or code of the country in ISO 3166-1 alpha-2 or alpha-3 format, or a CLDR short name. Examples: QA, QAT, Qatar (required)
- `state (str)`: The name of the state or region where the city is located (optional, default= `None`)

Raises:

- `InvalidLocationException`: If the country is invalid.

Example:

```
import aladhan

location = aladhan.City("London", "GB") # London, United Kingdom
# or
location = aladhan.City("London", "GB", "England") # London, England, United Kingdom
```

2.3 aladhan.Address

Address location type takes the address as a parameter, it can be a street address, a city name, a postal code or a combination of all. According to the official aladhan.com prayer times API, here is a list of valid addresses:

1420 Austin Bluffs Parkway, Colorado Springs, CO OR 25 Hampstead High Street, London, NW3 1RL, United Kingdom OR Sultanahmet Mosque, Istanbul, Turkey

Arguments:

- `address (str)`: The address (required)

Example:

```
import aladhan

location = aladhan.Address("1420 Austin Bluffs Parkway, Colorado Springs, CO")
```

CHAPTER

THREE

CLIENT CLASS

This is the main class of the library, it's used to get the prayer times for a specific location and date, and other useful methods.

3.1 aladhan.Client

Arguments:

- `default_location: City | Coordinates | Address`: The location of the client. This can be a City, Coordinates or Address object (optional, default= `None`).

Example of initializing a client with a City object:

```
>>> import aladhan
>>>
>>> location = aladhan.City("London", "GB")
>>> client = aladhan.Client(location)
```

Note: The *default_location* argument is optional. If it is not provided when the class is initialized, then the location must be provided for each method call. If *default_location* is provided both in the main class and in a method call, then the location provided in the method call will be used.

3.2 aladhan.Client.get_today_times

Returns a list of *Adhan* objects for today's prayer times.

Arguments:

- `location: City | Coordinates | Address`: Location of the prayer times (optional, default= `None`).

3.3 aladhan.Client.get_calendar_times

Returns a list of *Adhan* objects for the current month's prayer times, or you can specify the `month` and `year` arguments to get a specifc date.

Arguments:

- `location: City | Coordinates | Address`: Location of the prayer times (optional).
- `month: int`: Month number (optional, default= `None`).
- `year: int`: Year number (optional, default= `None`).

3.4 aladhan.Client.get_annual_times

Returns a list of *Adhan* objects for specifc year's prayer times (current year by default), which returns all the prayer times for the year, day by day.

Arguments:

- `location: City | Coordinates | Address`: Location of the prayer times (optional, default=``None``).
- `year: int`: Year number (optional, default= `None`).

All the get_*_times methods have the same raises:

- `aladhan.exceptions.InvalidLocationException`: Raised when the location is invalid.
- `aladhan.exceptions.RateLimitedException`: Raised when the client is rate limited by the server.
- `aladhan.exceptions.BadRequestException`: Raised when the request is invalid (e.g. invalid date).
- `aladhan.exceptions.ServerErrorException`: Raised when the server responds with unhandled error.

3.5 aladhan.Client.api_status

Get the status of the API, returns a dictionary with the *status* and *code* keys, might raise an *requests.exceptions.ConnectionError* if the website is down.

Returns:

- `dict`: A dictionary with the `status` and `code` keys.

3.6 Table of today's prayer times example

Let's get today's prayer times for Doha, Qatar, and print them as a table:

```
import aladhan

location = aladhan.City("Doha", "QA") # Doha, Qatar
client = aladhan.Client(location)

adhans = client.get_today_times()

print("Today's Prayer Times for Doha, Qatar")
```

(continues on next page)

(continued from previous page)

```
print("=====================================")
for adhan in adhans:
    print("{: <15} | {: <15}".format(adhan.get_en_name(), adhan.readable_timing(show_
↪date=False)))
```

Output:

```
Today's Prayer Times for Doha, Qatar
=====================================
Fajr            | 04:51 AM
Dhuhr           | 12:30 PM
Asr             | 03:45 PM
Maghrib         | 06:20 PM
Isha            | 07:40 PM
```

CHAPTER

FOUR

ADHAN OBJECT

Let's get some `Adhan` objects:

```
import aladhan

# initialize the client
location = aladhan.City("Doha", "QA")
client = aladhan.Client(location)

# get the today's adhan times using aladhan.Client.get_today_times
# this will return 5 Adhan objects
times = client.get_today_times()
```

Here is a full list of all the attributes of the `Adhan` object:

4.1 aladhan.Adhan.readable_timing

Returns the prayer time in a human readable format.

Arguments:

- `show_time (bool)`: to show the time or not (optional, default= `True`)
- `show_date (bool)`: to show the date in the YYYY/MM/DD format or not (optional, default= `True`)
- `_24h (bool)` display the time in 24h format (optional, default= `False`)
- `arabic (bool)` display the time in Arabic (change English numbers to Arabic and PM/AM to /) (optional, default= `False`)

Example:

```
>>> some_adhan = calendar[2]
>>>
>>> print(some_adhan.readable_timing())
>>> # 2021/07/03 11:00 AM
>>>
>>> print(some_adhan.readable_timing(show_time=False))
>>> # 2021/07/03
>>>
>>> print(some_adhan.readable_timing(show_date=False))
>>> # 11:00 AM
>>>
```

(continues on next page)

(continued from previous page)

```
>>> print(some_adhan.readable_timing(_24h=True))
>>> # 2021/07/03 11:00
```

4.2 aladhan.Adhan.wait

Wait until the salah time has passed, and do a callback if provided.

Arguments:

- `callback (callable)`: a function to call when the salah time has passed (optional, default= `None`)
- `threaded_wait (bool)`: to wait in a separate thread or not, this is useful if you want to do other things while waiting and having a callback function (optional, default= `False`)
- `*args`: arguments to pass to the callback function (optional)
- `**kwargs`: keyword arguments to pass to the callback function (optional)

Returns:

- `None`
- `threading.Thread`: if `threaded_wait` is `True`

4.3 aladhan.Adhan.is_passed

Check if the salah time has passed or not.

Returns:

- `bool`: `True` if the salah time has passed, `False` otherwise.

4.4 aladhan.Adhan.is_hijri

Check if the salah is hijri () or not.

Returns:

- `bool`: `True` if the salah is hijri, `False` otherwise.

4.5 aladhan.Adhan.is_secret

Check if the salah is secret () or not.

Returns:

- `bool`: `True` if the salah is secret, `False` otherwise.

4.6 aladhan.Adhan.rakat

Get the number of rakat of the salah.

Returns:

- `int`: the number of rakat of the salah.

4.7 aladhan.Adhan.get_en_name

Returns the English name of the salah.

Returns:

- `str`: the English name of the salah.

4.8 aladhan.Adhan.get_ar_name

Returns the Arabic name of the salah.

Arguments:

- `tashkeel (bool)`: to add tashkeel to the Arabic name or not (optional, default= `False`)
- `include_al (bool)`: to include the definite article () or not (e.g. instead of) (optional, default= `True`)

Returns:

- `str`: the Arabic name of the salah.

4.9 aladhan.Adhan.sunnan_al_rawatib

Returns the sunnan al rawatib rakat number before and after the salah in a dict.

Returns:

- `dict`: a dict with the keys `before` and `after` and the values are the rakat numbers.

CHAPTER

FIVE

EXCEPTION HANDLING

You can handle exceptions by importing them directly from the `aladhan` module, this page describes what each exception is for, and when it's raised.

5.1 aladhan.BadRequestException

This exception is raised when the API returns a 400 bad request error, this happens when the API is unable to process the request, for example, some of the parameters are missing, or invalid.

```
import aladhan

location = aladhan.City("ABC", "EFG") # Invalid city and country
client = aladhan.Client(location)

print(client.get_today_times()) # Raises BadRequestException
```

5.2 aladhan.RateLimitedException

This exception is raised when the API returns a 429 Too Many Requests error, this happens when the API is being used too much, and the rate limit is reached.

According to the API, on this source:

```
For the AlAdhan API in each region, each IP is allowed approximately 14 requests per␣
↪second.
```

This could happen if you're calling the API too fast.

5.3 aladhan.ServerErrorException

This exception is raised when the API returns a 500 Internal Server Error error, this means that the API got a non-handled error, and it's unable to process the request.

Which is very common in the API due to unhandled errors.

5.4 aladhan.InvalidLocationException

Exception raised when a invalid location is provided, e.g invalid coordinates.

```
>>> import aladhan
>>>
>>> location = aladhan.Coordinates(0, 1000)
Traceback (most recent call last):
    File "...", line 1, in <module>
    File "...\location_types.py", line 10, in __init__
        raise InvalidLocationException("Invalid coordinates")
aladhan.exceptions.InvalidLocationException: Invalid coordinates
```