wayback Documentation

Release 0.3.0b1

Contributors

Mar 16, 2021

CONTENTS

Wayback is A Python API to the Internet Archive's Wayback Machine. It gives you tools to search for and load mementos (historical copies of web pages).

The Internet Archive maintains an official "internetarchive" Python package, but it does not focus on the Wayback Machine. Instead, it is mainly concerned with the APIs and tools that manage the Internet Archive as a whole: managing items and collections. These are how e-books, audio recordings, movies, and other content in the Internet Archive are managed. It doesn't, however, provide particularly good tools for finding or loading historical captures of specific URLs (i.e. the part of the Internet Archive called the "Wayback Machine"). That's what the *wayback* package does.

CHAPTER

ONE

INSTALLATION

Wayback is meant to be used as a Python library, and is best installed via *pip* on the command line:

```
$ pip install wayback
```

CHAPTER TWO

USAGE

Search for historical mementos (archived copies) of a URL. Download metadata about the mementos and/or the memento content itself.

2.1 Tutorial

2.1.1 What is the earliest memento of nasa.gov?

Instantiate a `WaybackClient`.

```
In [1]: from wayback import WaybackClient

In [2]: client = WaybackClient()
```

Search for all Wayback's records for nasa.gov.

```
In [3]: results = client.search('nasa.gov')
```

This statement should execute fairly quickly because it doesn't actually do much work. The object we get back, `results`, is a *generator*, a "lazy" object from which we can pull results, one at a time. As we pull items out of it, it loads them as needed from the Wayback Machine in chronological order. We can see that `results` by itself is not informative:

```
In [4]: results
Out[4]: <generator object WaybackClient.search at 0x7fbebc986e50>
```

There are couple ways to pull items out of generator like `results`. One simple way is to use the built-in Python function `next()`, like so:

```
In [5]: record = next(results)
```

This takes a moment to run because, now that we've asked to see the first item in the generator, this lazy object goes to fetch a chunk of results from the Wayback Machine. Looking at the record in detail,

```
In [6]: record
Out[6]: CdxRecord(key='gov,nasa)/', timestamp=datetime.datetime(1996, 12, 31, 23, 58,␣
↪47, tzinfo=datetime.timezone.utc), url='http://www.nasa.gov/', mime_type='text/html
↪', status_code=200, digest='MGIGF4GRGGF5GKV6VNCBAXOE3OR5BTZC', length=1811, raw_url=
↪'http://web.archive.org/web/19961231235847id_/http://www.nasa.gov/', view_url=
↪'http://web.archive.org/web/19961231235847/http://www.nasa.gov/')
```

we can find our answer: Wayback's first memento of nasa.gov was in 1996. We can use dot access on `record` to access the timestamp specifically.

```
In [7]: record.timestamp
Out[7]: datetime.datetime(1996, 12, 31, 23, 58, 47, tzinfo=datetime.timezone.utc)
```

2.1.2 How many times does the word 'mars' appear on nasa.gov?

Above, we access the metadata for the oldest memento on nasa.gov, stored in the variable `record`. Starting from where we left off, we'll access the *content* of the memento and do a very simple analysis.

The Wayback Machine provides multiple *playback modes* to view the data it has captured. The `wayback.Mode.view` mode is a copy edited for human viewers on the web, and the `wayback.Mode.original` mode is the original copy of what was captured when the page was scraped. For analysis purposes, we generally want `original`. (Check the documentation of `wayback.Mode` for a few other, less commonly used modes.)

Let's download the original content using `WaybackClient`. (You could download the content directly with an HTTP library like `requests`, but `WaybackClient` adds extra tools for dealing with Wayback Machine servers.)

```
In [8]: from wayback import Mode

# `Mode.original` is the default and doesn't need to be explicitly set;
# we've set it here to show how you might choose other modes.
In [9]: response = client.get_memento(record, mode=Mode.original)

In [10]: content = response.content.decode()
```

We can use the built-in method `count` on strings to count the number of times that `'mars'` appears in the content.

```
In [11]: content.count('mars')
Out[11]: 30
```

This is case-sensitive, so to be more accurate we should convert the content to lowercase first.

```
In [12]: content.lower().count('mars')
Out[12]: 39
```

We picked up a couple additional occurrences that the original count missed.

2.2 API Documentation

The Wayback Machine exposes its data through two different mechanisms, implementing two different standards for archival data, the CDX API and the Memento API. We implement a Python client that can speak both.

class `wayback.`**`WaybackClient`**(*session=None*)

A client for retrieving data from the Internet Archive's Wayback Machine.

You can use a WaybackClient as a context manager. When exiting, it will close the session it's using (if you've passed in a custom session, make sure not to use the context manager functionality unless you want to live dangerously).

Parameters

session [`requests.Session`, optional]

search(*self*, *url*, ***, *matchType=None*, *limit=None*, *offset=None*, *fastLatest=None*, *gzip=None*, *from_date=None*, *to_date=None*, *filter_field=None*, *collapse=None*, *showResumeKey=True*, *resumeKey=None*, *page=None*, *pageSize=None*, *resolveRevisits=True*, *skip_malformed_results=True*, *previous_result=None*, ***kwargs*)

Search archive.org's CDX API for all captures of a given URL.

This will automatically page through all results for a given search.

Returns an iterator of CdxRecord objects. The StopIteration value is the total count of found captures.

Note that even URLs without wildcards may return results with different URLs. Search results are matched by url_key, which is a SURT-formatted, canonicalized URL:

- Does not differentiate between HTTP and HTTPS
- Is not case-sensitive
- Treats `www.` and `www*.` subdomains the same as no subdomain at all

Note not all CDX API parameters are supported. In particular, this does not support: *output*, *fl*, *showDupeCount*, *showSkipCount*, *lastSkipTimestamp*, *showNumPages*, *showPagedIndex*.

Parameters

url [str] The URL to query for captures of.

matchType [str, optional] Must be one of 'exact', 'prefix', 'host', or 'domain'. The default value is calculated based on the format of *url*.

limit [int, optional] Maximum number of results per page (this iterator will continue to move through all pages unless *showResumeKey=False*, though).

offset [int, optional] Skip the first N results.

fastLatest [bool, optional] Get faster results when using a negative value for *limit*. It may return a variable number of results.

gzip [bool, optional] Whether output should be gzipped.

from_date [datetime or date, optional] Only include captures after this date. Equivalent to the *from* argument in the CDX API. If it does not have a time zone, it is assumed to be in UTC.

to_date [datetime or date, optional] Only include captures before this date. Equivalent to the *to* argument in the CDX API. If it does not have a time zone, it is assumed to be in UTC.

filter_field [str, optional] A filter for any field in the results. Equivalent to the *filter* argument in the CDX API. (format: *[!]field:regex*)

collapse [str, optional] Collapse consecutive results that match on a given field. (format: *fieldname* or *fieldname:N* – N is the number of chars to match.)

showResumeKey [bool, optional] If False, don't continue to iterate through all pages of results. The default value is True

resumeKey [str, optional] Start returning results from a specified resumption point/offset. The value for this is supplied by the previous page of results when *showResumeKey* is True.

page [int, optional] If using paging start from this page number (note: paging, as opposed to the using *resumeKey* is somewhat complicated because of the interplay with indexes and index sizes).

pageSize [int, optional] The number of index blocks to examine for each page of results. Index blocks generally cover about 3,000 items, so setting *pageSize=1* might return anywhere from 0 to 3,000 results per page.

resolveRevists [bool, optional] Attempt to resolve *warc/revisit* records to their actual content type and response code. Not supported on all CDX servers. Defaults to True.

skip_malformed_results [bool, optional] If true, don't yield records that look like they have no actual memento associated with them. Some crawlers will erroneously attempt to capture bad URLs like *http://mailto:someone@domain.com* or *http://data:image/jpeg;base64,AF34...* and so on. This is a filter performed client side and is not a CDX API argument. (Default: True)

previous_result [str, optional] *For internal use.* The CDX API sometimes returns repeated results. This is used to track the previous result so we can filter out the repeats.

****kwargs** Any additional CDX API options.

Yields

version: CdxRecord A `CdxRecord` encapsulating one capture or revisit

Raises

UnexpectedResponseFormat If the CDX response was not parseable.

References

- https://github.com/internetarchive/wayback/tree/master/wayback-cdx-server

get_memento(*self*, *url*, *datetime=None*, *mode=<Mode.original: 'id_'>*, ***, *exact=True*, *exact_redirects=None*, *target_window=86400*, *follow_redirects=True*)

Fetch a memento (an archived HTTP response) from the Wayback Machine.

Not all mementos can be successfully fetched (or "played back" in Wayback terms). In this case, `get_memento` can load the next-closest-in-time memento or it will raise `wayback.exceptions.MementoPlaybackError` depending on the value of the `exact` and `exact_redirects` parameters (see more details below).

Parameters

url [string or CdxRecord] URL to retrieve a memento of. This can be any of:

- A normal URL (e.g. `http://www.noaa.gov/`). When using this form, you must also specify `datetime`.
- A `CdxRecord` retrieved from `wayback.WaybackClient.search()`.
- A URL of the memento in Wayback, e.g. `http://web.archive.org/web/20180816111911id_/http://www.noaa.gov/`

datetime [datetime.datetime or datetime.date or str, optional] The time at which to retrieve a memento of `url`. If `url` is a `wayback.CdxRecord` or full memento URL, this parameter can be omitted.

mode [wayback.Mode or str, optional] The playback mode of the memento. This determines whether the content of the returned memento is exactly as originally captured (the default) or modified in some way. See `wayback.Mode` for a description of possible values.

For more details, see: http://archive-access.sourceforge.net/projects/wayback/administrator_manual.html#Archival_URL_Replay_Mode

Default: `wayback.Mode.original`

exact [boolean, optional] If false and the requested memento either doesn't exist or can't be played back, this returns the closest-in-time memento to the requested one, so long as it is within `target_window`. If there was no memento in the target window or if `exact=True`, then this will raise `wayback.exceptions.MementoPlaybackError`. Default: True

exact_redirects [boolean, optional] If false and the requested memento is a redirect whose *target* doesn't exist or can't be played back, this returns the closest-in-time memento to the intended target, so long as it is within `target_window`. If unset, this will be the same as `exact`.

target_window [int, optional] If the memento is of a redirect, allow up to this many seconds between the capture of the redirect and the capture of the redirect's target URL. This window also applies to the first memento if `exact=False` and the originally requested memento was not available. Defaults to 86,400 (24 hours).

follow_redirects [boolean, optional] If true (the default), `get_memento` will follow historical redirects to return the content that a web browser would have ultimately displayed at the requested URL and time, rather than the memento of an HTTP redirect response (i.e. a 3xx status code). That is, if `http://example.com/a` redirected to `http://example.com/b`, then this method returns the memento for `/a` when `follow_redirects=False` and the memento for `/b` when `follow_redirects=True`. Default: True

Returns

dict [requests.Response] An HTTP response with the content of the memento, including a history of any redirects involved. (For a complete history of all HTTP requests needed to obtain the memento [rather than historic redirects], check `debug_history` instead of `history`.)

class wayback.**CdxRecord**(*key*, *timestamp*, *url*, *mime_type*, *status_code*, *digest*, *length*, *raw_url*, *view_url*)

Item from iterable of results returned by `WaybackClient.search()`

These attributes contain information provided directly by CDX.

digest
Content hashed as a base 32 encoded SHA-1.

key
SURT-formatted URL

length
Size of captured content in bytes, such as `2767`. This may be inaccurate. If the record is a "revisit record", indicated by MIME type `'warc/revisit'`, the length seems to be the length of the reference, not the length of the content itself.

mime_type
MIME type of record, such as `'text/html'`, `'warc/revisit'` or `'unk'` ("unknown") if this information was not captured.

status_code
Status code returned by the server when the record was captured, such as `200`. This is may be `None` if the record is a revisit record.

timestamp
The capture time represented as a `datetime.datetime`, such as `datetime.datetime(1996, 12, 31, 23, 58, 47, tzinfo=timezone.utc)`.

url
The URL that was captured by this record, such as `'http://www.nasa.gov/'`.

And these attributes are synthesized from the information provided by CDX.

raw_url
The URL to the raw captured content, such as `'http://web.archive.org/web/19961231235847id_/http://www.nasa.gov/'`.

view_url
The URL to the public view on Wayback Machine. In this view, the links and some subresources in the document are rewritten to point to Wayback URLs. There is also a navigation panel around the content. Example URL: `'http://web.archive.org/web/19961231235847/http://www.nasa.gov/'`.

class wayback.**Memento**(**, url, timestamp, mode, memento_url, status_code, headers, encoding, raw, raw_headers, history, debug_history*)
Represents a memento (an archived HTTP response). This object is similar to a response object from the popular "Requests" package, although it has some differences designed to differentiate historical information vs. current metadata about the stored memento (for example, the `headers` attribute lists the headers recorded in the memento, and does not include additional headers that provide metadata about the Wayback Machine).

Note that, like an HTTP response, this object represents a potentially open network connection to the Wayback Machine. Reading the `content` or `text` attributes will read all the data being received and close the connection automatically, but if you do not read those properties, you must make sure to call `close()` to close to connection. Alternatively, you can use a Memento as a context manager. The connection will be closed for you when the context ends:

```
>>> with a_memento:
>>>     do_something()
>>> # Connection is automatically closed here.
```

Fields

encoding: str
The text encoding of the response, e.g. `'utf-8'`.

headers: dict
A dict representing the headers of the archived HTTP response. The keys are case-sensitive.

history: tuple[*wayback.Memento*]
A list of *wayback.Memento* objects that were redirects and were followed to produce this memento.

debug_history: tuple[str]
List of all URLs redirects followed in order to produce this memento. These are "memento URLs" – that is, they are absolute URLs to the Wayback machine like `http://web.archive.org/web/20180816111911id_/http://www.noaa.gov/`, rather than URLs of captured redirects, like `http://www.noaa.gov`. Many of the URLs in this list do not represent actual mementos.

status_code: int
The HTTP status code of the archived HTTP response.

mode: str
The playback mode used to produce the Memento.

timestamp: datetime.datetime
The time the memento was originally captured. This includes `tzinfo`, and will always be in UTC.

url: str
The URL that the memento represents, e.g. `http://www.noaa.gov`.

`memento_url: str`
The URL at which the memento was fetched from the Wayback Machine, e.g. `http://web.archive.org/web/20180816111911id_/http://www.noaa.gov/`.

`ok: bool`
Whether the response had an non-error status (i.e. < 400).

`is_redirect: bool`
Whether the response was a redirect (i.e. had a 3xx status).

`content: bytes`
The body of the archived HTTP response in bytes.

`text: str`
The body of the archived HTTP response decoded as a string.

`close`(*self*)
Close the HTTP response for this Memento. This happens automatically if you read `content` or `text`, and if you use the memento as a context manager. This method is always safe to call – it does nothing if the response has already been closed.

`classmethod parse_memento_headers`(*raw_headers*, *url='http://web.archive.org/'*)
Extract historical headers from the Memento HTTP response's headers.

Parameters

raw_headers [dict] A dict of HTTP headers from the Memento's HTTP response.

url [str, optional] The URL of the resource the headers are being parsed for. It's used when header data contains relative/incomplete URL information.

Returns

dict

`class` `wayback.`**`WaybackSession`**(*retries=6*, *backoff=2*, *timeout=None*, *user_agent=None*)
A custom session object that network pools connections and resources for requests to the Wayback Machine.

Parameters

retries [int, optional] The maximum number of retries for requests.

backoff [int or float, optional] Number of seconds from which to calculate how long to back off and wait when retrying requests. The first retry is always immediate, but subsequent retries increase by powers of 2:

seconds = backoff * 2 ^ (retry number - 1)

So if this was *4*, retries would happen after the following delays: 0 seconds, 4 seconds, 8 seconds, 16 seconds, …

timeout [int or float or tuple of (int or float, int or float), optional] A timeout to use for all requests. If not set, there will be no no explicit timeout. See the Requests docs for more: http://docs.python-requests.org/en/master/user/advanced/#timeouts

user_agent [str, optional] A custom user-agent string to use in all requests. Defaults to: *wayback/{version} (+https://github.com/edgi-govdata-archiving/wayback)*

`reset`(*self*)
Reset any network connections the session is using.

2.2.1 Utilities

`wayback.`**`memento_url_data`**`(`*memento_url*`)`

Get the original URL, time, and mode that a memento URL represents a capture of.

Returns

url [str] The URL that the memento is a capture of.

time [datetime.datetime] The time the memento was captured in the UTC timezone.

mode [str] The playback mode.

Examples

Extract original URL, time and mode.

```
>>> url = ('http://web.archive.org/web/20170813195036id_/'
...        'https://arpa-e.energy.gov/?q=engage/events-workshops')
>>> memento_url_data(url)
('https://arpa-e.energy.gov/?q=engage/events-workshops',
 datetime.datetime(2017, 8, 13, 19, 50, 36, tzinfo=timezone.utc),
 'id_')
```

`class` `wayback.`**`Mode`**`(`*value*`)`

An enum describing the playback mode of a memento. When requesting a memento (e.g. with *`wayback.WaybackClient.get_memento()`*), you can use these values to determine how the response body should be formatted.

For more details, see: http://archive-access.sourceforge.net/projects/wayback/administrator_manual.html#Archival_URL_Replay_Mode

Examples

```
>>> waybackClient.get_memento('https://noaa.gov/',
>>>                           datetime=datetime.datetime(2018, 1, 2),
>>>                           mode=wayback.Mode.view)
```

Values

`original`

Returns the HTTP response body as originally captured.

`view`

Formats the response body so it can be viewed with a web browser. URLs for links and subresources like scripts, stylesheets, images, etc. will be modified to point to the equivalent memento in the Wayback Machine so that the resulting page looks as similar as possible to how it would have appeared when originally captured. It's mainly meant for use with HTML pages. This is the playback mode you typically use when browsing the Wayback Machine with a web browser.

`javascript`

Formats the response body by updating URLs, similar to `Mode.view`, but designed for JavaScript instead of HTML.

`css`

Formats the response body by updating URLs, similar to `Mode.view`, but designed for CSS instead of HTML.

image
formats the response body similar to `Mode.view`, but designed for image files instead of HTML.

2.2.2 Exception Classes

class `wayback.exceptions.`**`WaybackException`**
Base exception class for all Wayback-specific errors.

class `wayback.exceptions.`**`UnexpectedResponseFormat`**
Raised when data returned by the Wayback Machine is formatted in an unexpected or unparseable way.

class `wayback.exceptions.`**`BlockedByRobotsError`**
Raised when a URL can't be queried in Wayback because it was blocked by a site's *robots.txt* file.

class `wayback.exceptions.`**`BlockedSiteError`**
Raised when a URL has been blocked from access or querying in Wayback. This is often because of a takedown request. (URLs that are blocked because of `robots.txt` get a `BlockedByRobotsError` instead.)

class `wayback.exceptions.`**`MementoPlaybackError`**
Raised when a Memento can't be 'played back' (loaded) by the Wayback Machine for some reason. This is a server-side issue, not a problem in parsing data from Wayback.

class `wayback.exceptions.`**`NoMementoError`**
Raised when there was no memento available for a given URL. This might mean the given URL has no mementos at all or that none that are available for playback.

This also means you should *not* try to request a memento of the same URL in a different timeframe. If there may be other mementos of the URL available, you'll get a different error.

class `wayback.exceptions.`**`RateLimitError`**(*response*)
Raised when the Wayback Machine responds with a 429 (too many requests) status code. In general, this package's built-in limits should help you avoid ever hitting this, but if you are running multiple processes in parallel, you could go overboard.

Attributes

retry_after [int, optional] Recommended number of seconds to wait before retrying. If the Wayback Machine does not include it in the HTTP response, it will be set to `None`.

class `wayback.exceptions.`**`WaybackRetryError`**(*retries*, *total_time*, *causal_error*)
Raised when a request to the Wayback Machine has been retried and failed too many times. The number of tries before this exception is raised generally depends on your *WaybackSession* settings.

Attributes

retries [int] The number of retries that were attempted.

cause [Exception] The actual, underlying error that would have caused a retry.

time [int] The total time spent across all retried requests, in seconds.

class `wayback.exceptions.`**`SessionClosedError`**
Raised when a Wayback session is used to make a request after it has been closed and disabled.

CHAPTER

THREE

RELEASE HISTORY

3.1 v0.3.0 Beta 1 (2021-03-15)

`wayback.WaybackClient.get_memento()` now raises `wayback.exceptions.NoMementoError` when the requeted URL has never been archived. It also now raises `wayback.exceptions.MementoPlaybackError` in all other cases where an error was returned by the Wayback Machine (so you should never see a `requests.exceptions.HTTPError`). However, you may still see other *network-level* errors (e.g. `ConnectionError`).

3.2 v0.3.0 Alpha 3 (2020-11-05)

Fixes a bug in the new `wayback.Memento` type where header parsing would fail for mementos with schemeless `Location` headers. (#61)

3.3 v0.3.0 Alpha 2 (2020-11-04)

Fixes a bug in the new `wayback.Memento` type where header parsing would fail for mementos with path-based `Location` headers. (#60)

3.4 v0.3.0 Alpha 1 (2020-10-20)

Breaking Changes:

This release focuses on `wayback.WaybackClient.get_memento()` and makes major, breaking changes to its parameters and return type. They're all improvements, though, we promise!

get_memento() Parameters

The parameters in `wayback.WaybackClient.get_memento()` have been re-organized. The method signature is now:

```
def get_memento(self,
                url,                          # Accepts new types of values.
                datetime=None,                # New parameter.
                mode=Mode.original,           # New parameter.
                *,                            # Everything below is keyword-only.
                exact=True,
```

(continues on next page)

(continued from previous page)

```
                  exact_redirects=None,
                  target_window=24 * 60 * 60,
                  follow_redirects=True)       # New parameter.
```

- All parameters except `url` (the first parameter) from v0.2.x must now be specified with keywords, and cannot be specified positionally.

  If you previously used keywords, your code will be fine and no changes are necessary:

  ```
  # This still works great!
  client.get_memento('http://web.archive.org/web/20180816111911id_/http://www.noaa.
  ↪gov/',
                     exact=False,
                     exact_redirects=False,
                     target_window=3600)
  ```

  However, positional parameters like the following will now cause problems, and you should switch to the above keyword form:

  ```
  # This will now cause you some trouble :(
  client.get_memento('http://web.archive.org/web/20180816111911id_/http://www.noaa.
  ↪gov/',
                     False,
                     False,
                     3600)
  ```

- The `url` parameter can now be a normal, non-Wayback URL or a *`wayback.CdxRecord`*, and new `datetime` and `mode` parameters have been added.

  Previously, if you wanted to get a memento of what `http://www.noaa.gov/` looked like on August 1, 2018, you would have had to construct a complex string to pass to `get_memento()`:

  ```
  client.get_memento('http://web.archive.org/web/20180801000000id_/http://www.noaa.
  ↪gov/')
  ```

  Now you can pass the URL and time you want as separate parameters:

  ```
  client.get_memento('http://www.noaa.gov/', datetime.datetime(2018, 8, 1))
  ```

  If the `datetime` parameter does not specify a timezone, it will be treated as UTC (*not* local time).

  You can also pass a *`wayback.CdxRecord`* that you received from *`wayback.WaybackClient.search()`* instead of a URL and time:

  ```
  for record in client.search('http://www.noaa.gov/'):
      client.get_memento(record)
  ```

  Finally, you can now specify the *playback mode* of a memento using the `mode` parameter:

  ```
  client.get_memento('http://www.noaa.gov/',
                     datetime=datetime.datetime(2018, 8, 1),
                     mode=wayback.Mode.view)
  ```

  The default mode is *`wayback.Mode.original`*, which returns the exact HTTP response body as was originally archived. Other modes reformat the response body so it's more friendly for browsing by changing the URLs of links, images, etc. and by adding informational content to the page about the memento you are viewing. They are the modes typically used when you view the Wayback Machine in a web browser.

Don't worry, though — complete Wayback URLs are still supported. This code still works fine:

```
client.get_memento('http://web.archive.org/web/20180801000000id_/http://www.noaa.
↪gov/')
```

- A new `follow_redirects` parameter specifies whether to follow *historical* redirects (i.e. redirects that happened when the requested memento was captured). It defaults to `True`, which matches the old behavior of this method.

get_memento() Returns a Memento Object

`get_memento()` no longer returns a response object from the Requests package. Instead it returns a specialized *`wayback.Memento`* object, which is similar, but provides more useful information about the Memento than just the HTTP response from Wayback. For example, `memento.url` is the original URL the memento is a capture of (e.g. `http://www.noaa.gov/`) rather than the Wayback URL (e.g. `http://web.archive.org/web/20180816111911id_/http://www.noaa.gov/`). You can still get the full Wayback URL from `memento.memento_url`.

You can check out the full API docs for *`wayback.Memento`*, but here's a quick guide to what's available:

```
memento = client.get_memento('http://www.noaa.gov/home',
                             datetime(2018, 8, 16, 11, 19, 11),
                             exact=False)

# These values were previously not available except by parsing
# `memento.url`. The old `memento.url` is now `memento.memento_url`.
memento.url == 'http://www.noaa.gov/'
memento.timestamp == datetime(2018, 8, 29, 8, 8, 49, tzinfo=timezone.utc)
memento.mode == 'id_'

# Used to be `memento.url`:
memento.memento_url == 'http://web.archive.org/web/20180816111911id_/http://www.noaa.
↪gov/'

# Used to be a list of `Response` objects, now a *tuple* of Mementos. It
# Still lists only the redirects that are actual Mementos and not part of
# Wayback's internal machinery:
memento.history == (Memento<url='http://noaa.gov/home'>,)

# Used to be a list of `Response` objects, now a *tuple* of URL strings:
memento.debug_history == ('http://web.archive.org/web/20180816111911id_/http://noaa.
↪gov/home',
                          'http://web.archive.org/web/20180829092926id_/http://noaa.
↪gov/home',
                          'http://web.archive.org/web/20180829092926id_/http://noaa.
↪gov/')

# Headers now only lists headers from the original, archived response, not
# additional headers from the Wayback Machine itself. (If there's
# important information you needed in the headers, file an issue and let
# us know! We'd like to surface that kind of information as attributes on
# the Memento now.
memento.headers = {'header_name': 'header_value',
                   'another_header': 'another_value',
                   'and': 'so on'}

# Same as before:
memento.status_code
```

(continues on next page)

(continued from previous page)

```
memento.ok
memento.is_redirect
memento.encoding
memento.content
memento.text
```

Under the hood, *Wayback* still uses Requests for HTTP requests, but we expect to change that soon to ensure this package is thread-safe.

Other Breaking Changes

Finally, `wayback.memento_url_data()` now returns 3 values instead of 2. The last value is a string representing the playback mode (see above description of the new `mode` parameter on `wayback.WaybackClient.get_memento()` for more about playback modes).

3.5 v0.2.5 (2020-10-19)

This release fixes a bug where the `target_window` parameter for `wayback.WaybackClient.get_memento()` did not work correctly if the memento you were redirected to was off by more than a day from the requested time. See #53 for more.

3.6 v0.2.4 (2020-09-07)

This release is focused on improved error handling.

Breaking Changes:

- The timestamps in `CdxRecord` objects returned by `wayback.WaybackClient.search()` now include timezone information. (They are always in the UTC timezone.)

Updates:

- The `history` attribute of a memento now only includes redirects that were mementos (i.e. redirects that would have been seen when browsing the recorded site at the time it was recorded). Other redirects involved in working with the memento API are still available in `debug_history`, which includes all redirects, whether or not they were mementos.
- Wayback's CDX search API sometimes returns repeated, identical results. These are now filtered out, so repeat search results will not be yielded from `wayback.WaybackClient.search()`.
- `wayback.exceptions.RateLimitError` will now be raised as an exception any time you breach the Wayback Machine's rate limits. This would previously have been `wayback.exceptions.WaybackException`, `wayback.exceptions.MementoPlaybackError`, or regular HTTP responses, depending on the method you called. It has a `retry_after` property that indicates how many seconds you should wait before trying again (if the server sent that information, otherwise it will be `None`).
- `wayback.exceptions.BlockedSiteError` will now be raised any time you search for a URL or request a memento that has been blocked from access (for example, in situations where the Internet Archive has received a takedown notice).

3.7 v0.2.3 (2020-03-25)

This release downgrades the minimum Python version to 3.6! You can now use Wayback in places like Google Colab.

The `from_date` and `to_date` arguments for *`wayback.WaybackClient.search()`* can now be `datetime.date` instances in addition to `datetime.datetime`.

Huge thanks to @edsu for implementing both of these!

3.8 v0.2.2 (2020-02-13)

When errors were raised or redirects were involved in `WaybackClient.get_memento()`, it was previously possible for connections to be left hanging open. Wayback now works harder to make sure connections aren't left open.

This release also updates the default user agent string to include the repo URL. It now looks like: `wayback/0.2.2 (+https://github.com/edgi-govdata-archiving/wayback)`

3.9 v0.2.1 (2019-12-01)

All custom exceptions raised publicly and used internally are now exposed via a new module, `wayback.exceptions`.

3.10 v0.2.0 (2019-11-26)

Initial release of this project. See v0.1 below for information about a separate project with the same name that has since been removed from PyPI.

3.11 v0.1

This version number is reserved because it was the last published release of a separate Python project also named `wayback` that has since been deleted from the Python Package Index and subsequently superseded by this one. That project, which focused on the Wayback Machine's timemap API, was maintained by Jeff Goettsch (username `jgoettsch` on the Python Package Index). Its source code is still available on BitBucket at https://bitbucket.org/jgoettsch/py-wayback/.

INDEX