youtube_transcript_api._api.CouldNotRetrieveTranscript

youtube_transcript_api._api.YouTubeTranscriptApi.CouldNotRetrieveTranscript: Could not get the transcript for the video https://www.youtube.com/watch?v=T! This usually happens if one of the following things is the case: - subtitles have been disabled by the uploader - none of the language codes you provided are valid - none of the languages you provided are supported by the video - the video is no longer available. If none of these things is the case, please create an issue at https://github.com/jdepoix/youtube-transcript-api/issues

Traceback *(most recent call last)*

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/youtube_transcript_api/_api.py*", **line** *93*, **in** **`get_transcript`**

```
        :type proxies: {'http': str, 'https': str} -
http://docs.python-requests.org/en/master/user/
advanced/#proxies
        :return: a list of dictionaries containing the
'text', 'start' and 'duration' keys
        :rtype: [{'text': str, 'start': float, 'end':
float}]
        """
        try:
            return
_TranscriptParser(_TranscriptFetcher(video_id,
languages, proxies).fetch()).parse()
        except Exception:
            raise
YouTubeTranscriptApi.CouldNotRetrieveTranscript(video_
id)


class _TranscriptFetcher():
```

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/youtube_transcript_api/_api.py*", **line** *153*, **in** **`parse`**

```
            {
                'text': re.sub(self.HTML_TAG_REGEX,
'', unescape(xml_element.text)),
                'start':
```

```
float(xml_element.attrib['start']),
                'duration':
float(xml_element.attrib['dur']),
            }
            for xml_element in
ElementTree.fromstring(self.plain_data)
            if xml_element.text is not None
        ]
```

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/xml/etree/ElementTree.py*"**, line** *1315*, **in XML**

```
    Returns an Element instance.

    """
    if not parser:
        parser = XMLParser(target=TreeBuilder())
    parser.feed(text)
    return parser.close()


def XMLID(text, parser=None):
    """Parse XML document from string constant for its
IDs.
```

- During handling of the above exception, another exception occurred:

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/app.py*"**, line** *2463*, **in __call__**

```
    def __call__(self, environ, start_response):
        """The WSGI server calls the Flask application
object as the
        WSGI application. This calls :meth:`wsgi_app`
which can be
        wrapped to applying middleware."""
        return self.wsgi_app(environ, start_response)

    def __repr__(self):
        return "<%s %r>" % (self.__class__.__name__,
self.name)
```

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/app.py*", **line** *2449*, **in** **wsgi_app**

```
            try:
                ctx.push()
                response =
self.full_dispatch_request()
            except Exception as e:
                error = e
                response = self.handle_exception(e)
            except:  # noqa: B001
                error = sys.exc_info()[1]
                raise
            return response(environ, start_response)
        finally:
```

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/app.py*", **line** *1866*, **in** **handle_exception**

```
            # if we want to repropagate the exception,
we can attempt to
            # raise it with the whole traceback in
case we can do that
            # (the function was actually called from
the except part)
            # otherwise, we just raise the error again
            if exc_value is e:
                reraise(exc_type, exc_value, tb)
            else:
                raise e

        self.log_exception((exc_type, exc_value, tb))
        server_error = InternalServerError()
```

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/_compat.py*", **line** *39*, **in** **reraise**

```
    import collections.abc as collections_abc

    def reraise(tp, value, tb=None):
```

```
            if value.__traceback__ is not tb:
                raise value.with_traceback(tb)
            raise value

        implements_to_string = _identity

    else:
        iterkeys = lambda d: d.iterkeys()
```

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/app.py*", **line** *2446*, **in** **`wsgi_app`**

```
            ctx = self.request_context(environ)
            error = None
            try:
                try:
                    ctx.push()
                    response =
    self.full_dispatch_request()
                except Exception as e:
                    error = e
                    response = self.handle_exception(e)
                except:  # noqa: B001
                    error = sys.exc_info()[1]
```

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/app.py*", **line** *1951*, **in** **`full_dispatch_request`**

```
                request_started.send(self)
                rv = self.preprocess_request()
                if rv is None:
                    rv = self.dispatch_request()
            except Exception as e:
                rv = self.handle_user_exception(e)
            return self.finalize_request(rv)

        def finalize_request(self, rv,
    from_error_handler=False):
            """Given the return value from a view function
    this finalizes
            the request by converting it into a response
    and invoking the
```

- **File** *"/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/app.py"*, **line** *1820*, **in** **`handle_user_exception`**

```
            return self.handle_http_exception(e)

        handler = self._find_error_handler(e)

        if handler is None:
            reraise(exc_type, exc_value, tb)
        return handler(e)

    def handle_exception(self, e):
        """Handle an exception that did not have an
error handler
        associated with it, or that was raised from an
error handler.
```

- **File** *"/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/_compat.py"*, **line** *39*, **in** **`reraise`**

```
    import collections.abc as collections_abc

    def reraise(tp, value, tb=None):
        if value.__traceback__ is not tb:
            raise value.with_traceback(tb)
        raise value

    implements_to_string = _identity

else:
    iterkeys = lambda d: d.iterkeys()
```

- **File** *"/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/app.py"*, **line** *1949*, **in** **`full_dispatch_request`**

```
self.try_trigger_before_first_request_functions()
        try:
            request_started.send(self)
            rv = self.preprocess_request()
```

```
            if rv is None:
                rv = self.dispatch_request()
        except Exception as e:
            rv = self.handle_user_exception(e)
        return self.finalize_request(rv)

    def finalize_request(self, rv,
from_error_handler=False):
```

- **File** "*/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/flask/app.py*", **line** *1935*, **in** **`dispatch_request`**

```
            getattr(rule, "provide_automatic_options",
False)
            and req.method == "OPTIONS"
        ):
            return
self.make_default_options_response()
        # otherwise dispatch to the handler for that
endpoint
        return self.view_functions[rule.endpoint]
(**req.view_args)

    def full_dispatch_request(self):
        """Dispatches the request and on top of that
performs request
        pre and postprocessing as well as HTTP
exception catching and
        error handling.
```

- **File** "*/Volumes/124GB USB/INSIGHT/MyFlask/flaskexample/views.py*", **line** *44*, **in** **`mainpage`**

```
    video_ids = request.form['video_id']

    words=[]
    transcriptsmultiple={}
    for video_id in video_ids:
        transcript=
YouTubeTranscriptApi.get_transcript(video_id)
        list1 = [t['text'] for t in transcript]
        my_list=[]
        for item in list1:
```

```
                if item.find('[')==-1:
                    my_list.append(item)
```

- **File** *"/Users/MobileHome/anaconda3/envs/insight/lib/python3.7/site-packages/youtube_transcript_api/_api.py"*, **line** *95*, **in** **`get_transcript`**

```
        :rtype: [{'text': str, 'start': float, 'end': float}]
        """
        try:
            return _TranscriptParser(_TranscriptFetcher(video_id, languages, proxies).fetch()).parse()
        except Exception:
            raise YouTubeTranscriptApi.CouldNotRetrieveTranscript(video_id)


class _TranscriptFetcher():
    WATCH_URL = 'https://www.youtube.com/watch?v={video_id}'
    API_BASE_URL = 'https://www.youtube.com/api/{api_url}'
```

youtube_transcript_api._api.YouTubeTranscriptApi.CouldNotRetrieveTranscript: Could not get the transcript for the video https://www.youtube.com/watch?v=T! This usually happens if one of the following things is the case: - subtitles have been disabled by the uploader - none of the language codes you provided are valid - none of the languages you provided are supported by the video - the video is no longer available. If none of these things is the case, please create an issue at https://github.com/jdepoix/youtube-transcript-api/issues

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Brought to you by **DON'T PANIC**, your friendly Werkzeug powered traceback interpreter.

Console Locked

The console is locked and needs to be unlocked by entering the PIN. You can find the PIN printed out on the standard output of your shell that runs the server.

PIN: