SENDGRID INTEGRATION WITH PYTHON

Date	18 November 2022
Team ID	PNT2022TMID50239
Project Name	Nutrition Assistant Application

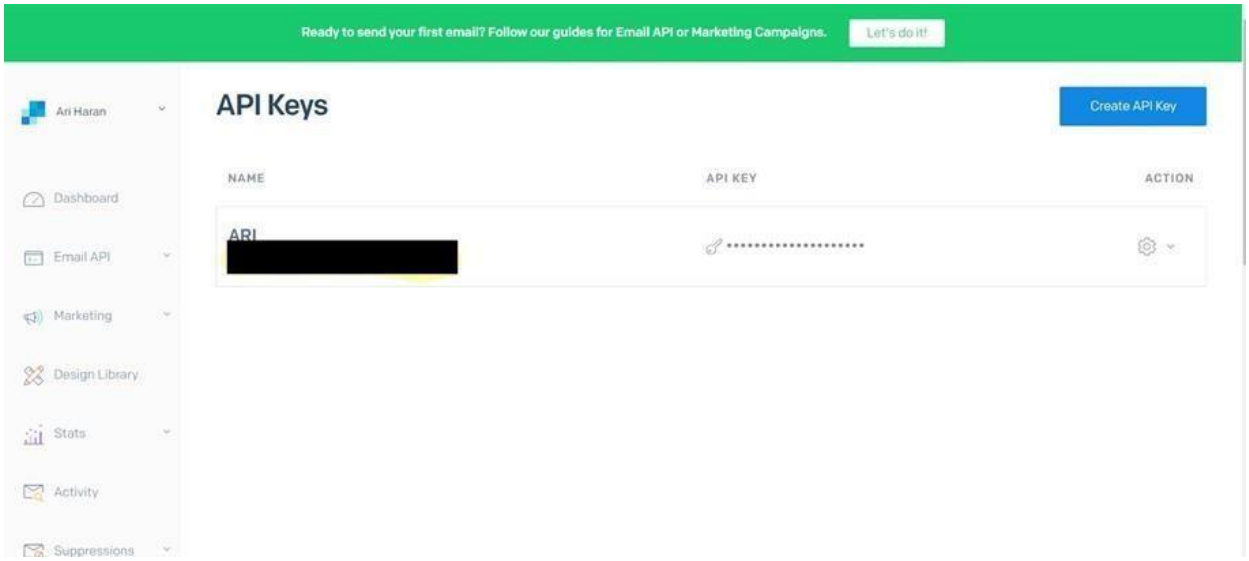
STEP 1:

REQUIREMENTS:

Python 2.6, 2.7, 3.4 or 3.5.

STEP 2:

Create an API key

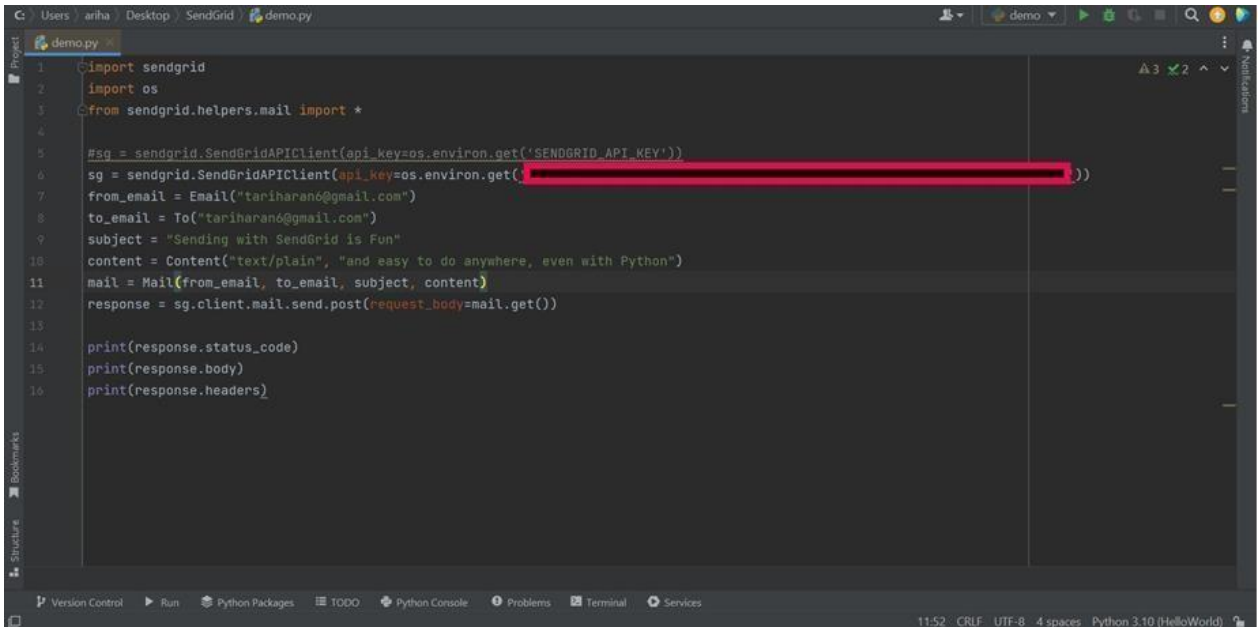


STEP 3:

INSTALL PAKAGE: > pip install sendgrid

SETP 4:

SEND EMAIL



SENDGRID PYTHON CODE :

```
"""HTTP Client library"""
import json
import logging
from .exceptions import handle_error

    try:
    # Python 3
    import urllib.request as urllib
    from urllib.parse import urlencode
    from urllib.error import HTTPError
except ImportError:
     # Python 2
```

```
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

    message = Mail(
    from_email='from_email@example.com',
    to_emails='to@example.com',
    subject='Sending with Twilio SendGrid is Fun',
    html_content='<strong>and easy to do anywhere, even with
    Python</strong>')
    try:
    sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sg.send(message)
    print(response.status_code)
    print(response.body) 15 print(response.headers) 16except Exception as
    e:
    print(e.message)
```

HTTP CLIENT PROGRAM:

```
        import urllib2 as urllib
14      from urllib2 import HTTPError
15      from urllib import urlencode
16
17  _logger = logging.getLogger( name )
18
19
20    class Response(object):
21    """Holds the response from an API call."""  22
23                            def init (self, response):
24                            """
25                            :param response: The return value from a
                              open call
26                            on a urllib.build_opener()
27                            :type response: urllib response object
28                            """
29                            self._status_code = response.getcode()
30                            self._body = response.read()
31                            self._headers = response.info()
32
33      @property
```

```
34          def status_code(self):
35          """
36          :return: integer, status code of API call
37          """
38          return self._status_code
39
40          @property
41          def body(self):
42          """
43          :return: response from the API
44          """
45          return self._body
46
47      @property
```

```
        def headers(self):
        """
        :return: dict of response headers
        """
        return self._headers

            @property
            def to_dict(self):
            """
            :return: dict of response from the API
            """
            if self.body:
            return json.loads(self.body.decode('utf-8'))
            else:
            return None


class Client(object):
"""Quickly and easily access any REST or REST-like API.""" 
 # These are the supported HTTP verbs
```

```
    methods = {'delete', 'get', 'patch', 'post', 'put'} 70
    def init (self,
    host,
    request_headers=None,
    version=None,
    url_path=None,
    append_slash=False, 77       timeout=None):
          """
          :param host: Base URL for the api. (e.g.
   https://api.sendgrid.com)
          :type host: string
          :param request_headers: A dictionary of the headers you want
                                  applied on all calls
          :type request_headers: dictionary
          :param version: The version number of the
          API.
          Subclass _build_versioned_url for custom
   behavior.
          Or just pass the version as part of the URL
          (e.g. client._("/v3"))
          :type version: integer
          :param url_path: A list of the url path
          segments
          :type url_path: list of strings
          """
          self.host = host
          self.request_headers = request_headers or {}
          self._version = version
          # _url_path keeps track of the dynamically
          built url
          self._url_path = url_path or []
          # APPEND SLASH set
```

```
                        self.append_slash = append_slash
                        self.timeout = timeout

    def _build_versioned_url(self, url):
```

```
102             """Subclass this function for your own needs.
103             Or just pass the version as part of the URL
104             (e.g. client._('/v3'))
105             :param url: URI portion of the full URL being requested
106             :type url: string
107             :return: string
108             """
109             return '{}/v{}{}'.format(self.host, str(self._version),
                url)
110
111         def _build_url(self, query_params):
112         """Build the final URL to be passed to urllib
113
114         :param query_params: A dictionary of all the query
```

parameters

```
            :type query_params: dictionary
            :return: string
            """
            url = ''
            count = 0
            while count < len(self._url_path):
            url += '/{}'.format(self._url_path[count])
            count += 1

            # add slash
            if self.append_slash:
            url += '/'

            if query_params:
            url_values = urlencode(sorted(query_params.items()), True)
            url = '{}?{}'.format(url, url_values)

            if self._version:
            url = self._build_versioned_url(url)
            else:
            url = '{}{}'.format(self.host, url)
            return url

        def _update_headers(self, request_headers):
        """Update the headers for the request

        :param request_headers: headers to set for the API call
        :type request_headers: dictionary
        :return: dictionary
```

```
144         """
145         self.request_headers.update(request_headers)
146 
147     def _build_client(self, name=None):
```

```
        """Make a new Client object

        :param name: Name of the url segment
        :type name: string
        :return: A Client object
        """
        url_path = self._url_path + [name] if name else
    self._url_path
        return Client(host=self.host,
                      version=self._version,
                      request_headers=self.request_headers,
                      url_path=url_path,
                      append_slash=self.append_slash,
                      timeout=self.timeout)

            def _make_request(self, opener, request,
            timeout=None):
            """Make the API call and return the response.    is
            This separated into                                testing.
            it's own function, so we can mock it easily for

          :param opener:
```

```
167             :type opener:
168             :param request: url payload to request
169             :type request: urllib.Request object
170             :param timeout: timeout value or None
171             :type timeout: float
172             :return: urllib response
173             """
174             timeout = timeout or self.timeout
175             try:
176             return opener.open(request, timeout=timeout)
177             except HTTPError as err:
178             exc = handle_error(err)
179             exc. cause ___= None
180             _logger.debug('{method} Response: {status}
```

```
{body}'.format(
```

```
                method=request.get_method(),
                status=exc.status_code,
                body=exc.body))
                raise exc

            def _(self, name):
            """Add variable values to the url.
            (e.g. /your/api/{variable_value}/call)
            Another example: if you have a Python reserved word, such as
            global,
            in your url, you must use this method.

    :param name: Name of the url segment
    :type name: string
    :return: Client object
    """
    return self._build_client(name) 197

            def getattr (self, name):
            """Dynamically add method calls to the url, then call a method.
            (e.g. client.name.name.method())
            You can also add a version number by using
 .version(<int>)

                :param name: Name of the url segment or method call
                :type name: string or integer if name == version
                :return: mixed
                """
                if name == 'version':
                def get_version(*args, **kwargs):
                """
                :param args: dict of settings
                :param kwargs: unused
```

```
                :return: string, version
                """
                self._version = args[0]
                return self._build_client()
            return get_version

        # We have reached the end of the method chain, make the
        API call
        if name in self.methods:
        method = name.upper()

                def http_request(
                request_body=None,
                query_params=None,
                request_headers=None,
                timeout=None,
                **_):
                """Make the API call
                :param timeout: HTTP request timeout. Will be
                propagated to
                urllib client
            :type timeout: float
```

```
                    :param request_headers: HTTP headers. Will be
   merged into
                    current client object state
                    :type request_headers: dict
                    :param query_params: HTTP query parameters
                    :type query_params: dict
                    :param request_body: HTTP request body
                    :type request_body: string or json-serializable
   object
                    :param kwargs:
                    :return: Response object
                    """
                    if request_headers:
```

```
                    self._update_headers(request_headers)

                if request_body is None:
                    data = None
                else:
                    # Don't serialize to a JSON formatted str
                    # if we don't have a JSON Content-Type
                    if 'Content-Type' in self.request_headers and \
                            self.request_headers['Content-Type'] !=
    \
                            'application/json':
                        data = request_body.encode('utf-8')
                    else:
                        self.request_headers.setdefault(
                            'Content-Type', 'application/json')
                        data =
    json.dumps(request_body).encode('utf-8')

                opener = urllib.build_opener()
                request = urllib.Request(
                    self._build_url(query_params),
                    headers=self.request_headers,
                    data=data,
                )
                request.get_method = lambda: method

                _logger.debug('{method} Request: {url}'.format(
                    method=method,
                    url=request.get_full_url()))
                if request.data:
                    _logger.debug('PAYLOAD: {data}'.format(
                        data=request.data))
                _logger.debug('HEADERS: {headers}'.format(
                    headers=request.headers))

```

```
                    response = Response(
                    self._make_request(opener, request,
   timeout=timeout)
278                 )
279
280                     _logger.debug('{method} Response: {status}
   {body}'.format(
281                     method=method,
282                     status=response.status_code,
283                     body=response.body))
284
285                 return response
286
287   return http_request 288
      else:
289             # Add a segment to the URL
290             return self._(name)
291
292         def getstate (self):
293         return self. dict
294
295       def setstate (self, state):
```