SENDGRID INTEGRATION WITH PYTHON CODE

TEAM ID	PNT2022TMID43147
PROJECT NAME	NUTRITION ASSISTANT APPLICATION

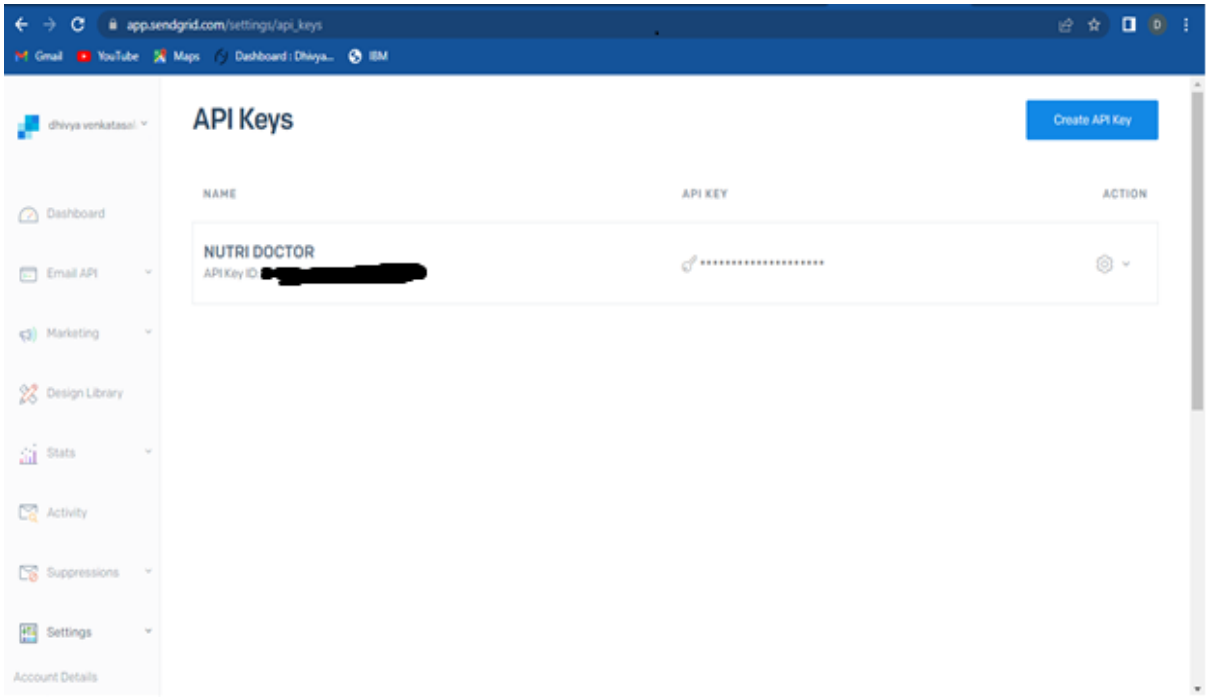
STEP 1:

REQUIREMENTS:

PYTHON 2.6,2.7,3.4 OR 3.5

STEP 2:

CREATE AN API KEY

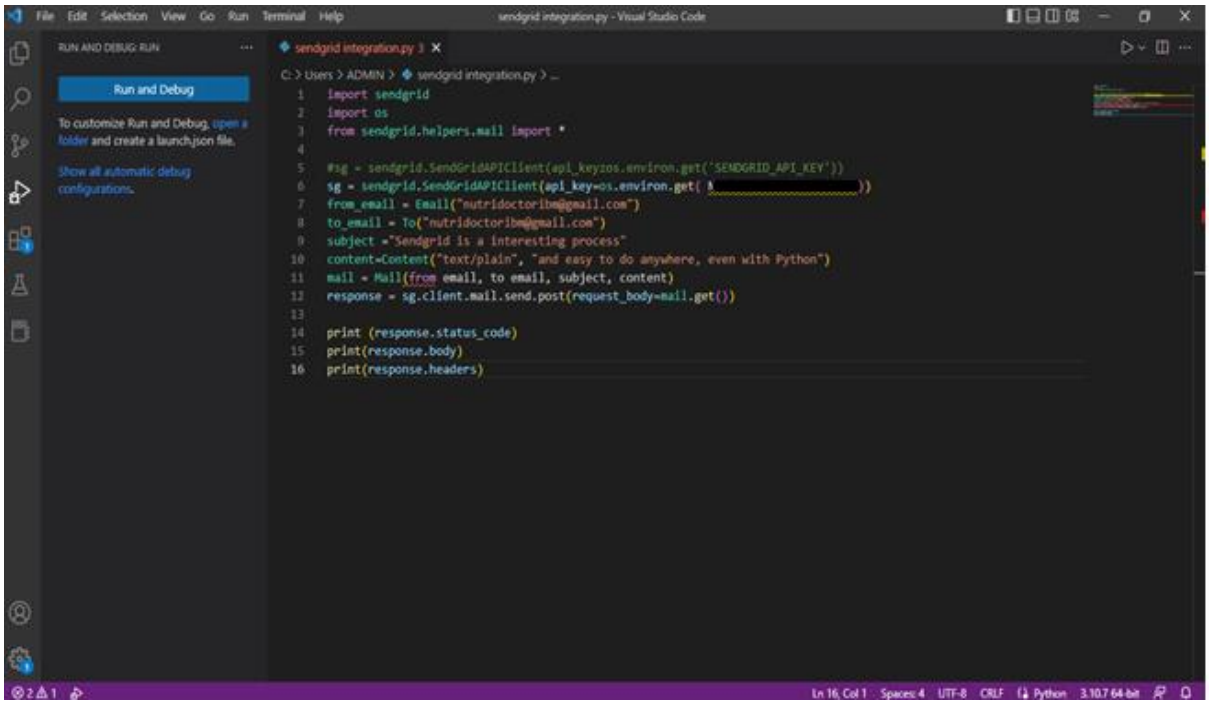


STEP 3:

INSTALL PACKAGE: > pipinstallsendgrid

STEP 4:

SEND EMAIL



SENDGRID PYTHON CODE :

```python
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)
    print(response.headers)
except Exception as e:
    print(e.message)
```

HTTP CLIENT PROGRAM

```
"""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 urllib2 as urllib
    from urllib2 import HTTPError
    from urllib import urlencode

_logger = logging.getLogger(__name__)


class Response(object):
    """Holds the response from an API call."""

    def __init__(self, response):
        """
        :param response: The return value from a open call
                         on a urllib.build_opener()
        :type response:  urllib response object
        """
        self._status_code = response.getcode()
        self._body = response.read()
        self._headers = response.info()

    @property
    def status_code(self):
        """
        :return: integer, status code of API call
        """
        return self._status_code

    @property
    def body(self):
        """
        :return: response from the API
        """
        return self._body

    @property
```

```
48      def headers(self):
49          """
50          :return: dict of response headers
51          """
52          return self._headers
53
54      @property
55      def to_dict(self):
56          """
57          :return: dict of response from the API
58          """
59          if self.body:
60              return json.loads(self.body.decode('utf-8'))
61          else:
62              return None
63
64
65  class Client(object):
66      """Quickly and easily access any REST or REST-like API."""
67
68      # These are the supported HTTP verbs
69      methods = ('delete', 'get', 'patch', 'post', 'put')
70
71      def __init__(self,
72                   host,
73                   request_headers=None,
74                   version=None,
75                   url_path=None,
76                   append_slash=False,
77                   timeout=None):
78          """
79          :param host: Base URL for the api. (e.g.
    https://api.sendgrid.com)
80          :type host:  string
81          :param request_headers: A dictionary of the headers you want
```

```
82                                  applied on all calls
83          :type request_headers: dictionary
84          :param version: The version number of the API.
85                          Subclass _build_versioned_url for custom
   behavior.
86                          Or just pass the version as part of the URL
87                          (e.g. client._("/v3"))
88          :type version: integer
89          :param url_path: A list of the url path segments
90          :type url_path: list of strings
91          """
92          self.host = host
93          self.request_headers = request_headers or {}
94          self._version = version
95          # _url_path keeps track of the dynamically built url
96          self._url_path = url_path or []
97          # APPEND SLASH set
98          self.append_slash = append_slash
99          self.timeout = timeout
100
101     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
115         :type query_params: dictionary
116         :return: string
117         """
118         url = ''
119         count = 0
120         while count < len(self._url_path):
121             url += '/{}'.format(self._url_path[count])
122             count += 1
123 
124         # add slash
125         if self.append_slash:
126             url += '/'
127 
128         if query_params:
129             url_values = urlencode(sorted(query_params.items()),
    True)
130             url = '{}?{}'.format(url, url_values)
131 
132         if self._version:
133             url = self._build_versioned_url(url)
134         else:
135             url = '{}{}'.format(self.host, url)
136         return url
137 
138     def _update_headers(self, request_headers):
139         """Update the headers for the request
140 
141         :param request_headers: headers to set for the API call
142         :type request_headers: dictionary
143         :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. This is separated into
           it's own function, so we can mock it easily for testing.

        :param opener:
        :type opener:
        :param request: url payload to request
        :type request: urllib.Request object
        :param timeout: timeout value or None
        :type timeout: float
        :return: urllib response
        """
        timeout = timeout or self.timeout
        try:
            return opener.open(request, timeout=timeout)
        except HTTPError as err:
            exc = handle_error(err)
            exc.__cause__ = None
            _logger.debug('{method} Response: {status}
```

```
    {body}'.format(
181                 method=request.get_method(),
182                 status=exc.status_code,
183                 body=exc.body))
184             raise exc
185
186     def _(self, name):
187         """Add variable values to the url.
188             (e.g. /your/api/{variable_value}/call)
189             Another example: if you have a Python reserved word,
    such as global,
190             in your url, you must use this method.
191
192         :param name: Name of the url segment
193         :type name: string
194         :return: Client object
195         """
196         return self._build_client(name)
197
198     def __getattr__(self, name):
199         """Dynamically add method calls to the url, then call a
    method.
200             (e.g. client.name.name.method())
201             You can also add a version number by using
    .version(<int>)
202
203         :param name: Name of the url segment or method call
204         :type name: string or integer if name == version
205         :return: mixed
206         """
207         if name == 'version':
208             def get_version(*args, **kwargs):
209                 """
210                     :param args: dict of settings
211                     :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:
```

```
243                     self._update_headers(request_headers)
244
245                 if request_body is None:
246                     data = None
247                 else:
248                     # Don't serialize to a JSON formatted str
249                     # if we don't have a JSON Content-Type
250                     if 'Content-Type' in self.request_headers and \
251                             self.request_headers['Content-Type'] !=
    \
252                             'application/json':
253                         data = request_body.encode('utf-8')
254                     else:
255                         self.request_headers.setdefault(
256                             'Content-Type', 'application/json')
257                         data =
    json.dumps(request_body).encode('utf-8')
258
259                 opener = urllib.build_opener()
260                 request = urllib.Request(
261                     self._build_url(query_params),
262                     headers=self.request_headers,
263                     data=data,
264                 )
265                 request.get_method = lambda: method
266
267                 _logger.debug('{method} Request: {url}'.format(
268                     method=method,
269                     url=request.get_full_url()))
270                 if request.data:
271                     _logger.debug('PAYLOAD: {data}'.format(
272                         data=request.data))
273                 _logger.debug('HEADERS: {headers}'.format(
274                     headers=request.headers))
275
```

```
                response = Response(
                    self._make_request(opener, request,
timeout=timeout)
                )

                _logger.debug('{method} Response: {status}
{body}'.format(
                    method=method,
                    status=response.status_code,
                    body=response.body))

                return response

            return http_request
        else:
            # Add a segment to the URL
            return self._(name)

    def __getstate__(self):
        return self.__dict

    def   setstate  (self, state):
```