SENDGRID INTEGRATION WITH PYTHON

Date	10 NOV 2022
Team ID	PNT2022TMID13419
Project Name	CUSTOMER CARE REGISTRY

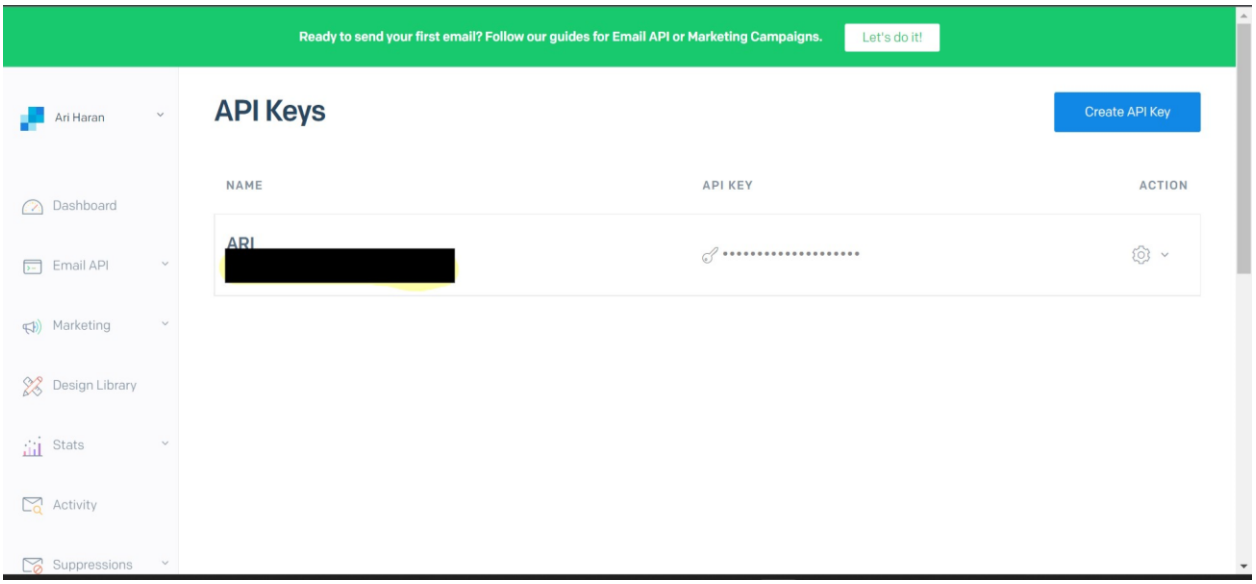
STEP 1:

REQUIREMENTS:

Python 2.6, 2.7, 3.4 or 3.5.

STEP 2:

Create an API key



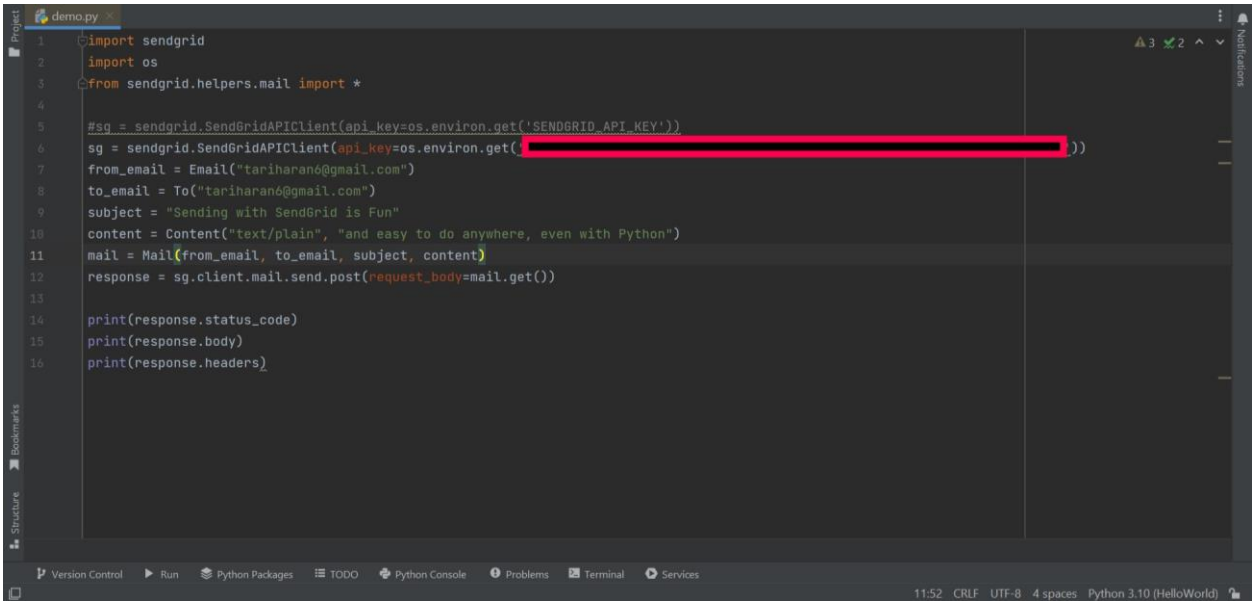
STEP 3:

INSTALL PAKAGE:

> pip install sendgrid

SETP 4:

SEND EMAIL



SENDGRID PYTHON CODE :

```
1         import os
2         from sendgrid import SendGridAPIClient 3 from
sendgrid.helpers.mail import Mail
4
5         message = Mail(
6         from_email='from_email@example.com',
7         to_emails='to@example.com',
8         subject='Sending with Twilio SendGrid is Fun',
9         html_content='<strong>and easy to do anywhere, even with
    Python</strong>')
10        try:
11        sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
12        response = sg.send(message) 13    print(response.status_code)
14          print(response.body)
15          print(response.headers)
16          except Exception as e:
17          print(e.message)
```

HTTP CLIENT PROGRAM:

```
1   """HTTP Client library"""
2   import json
3   import logging
4   from .exceptions import handle_error
5
6         try:
7         # Python 3
8         import urllib.request as urllib
9         from urllib.parse import urlencode 10    from urllib.error import HTTPError
11          except ImportError:
12          # Python 2
```

```
13          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
```

```
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
        :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
    """
    self.request_headers.update(request_headers)

def _build_client(self, name=None):
```

```
148     """Make a new Client object
149
150                             :param name: Name of the url segment
151                             :type name: string
152                             :return: A Client object
153                             """
154                             url_path = self._url_path + [name] if name else self._url_path
155                             return Client(host=self.host,
156                             version=self._version,
157                             request_headers=self.request_headers,
158                             url_path=url_path,
159                             append_slash=self.append_slash,
160                             timeout=self.timeout)
161
162         def _make_request(self, opener, request, timeout=None):
163         """Make the API call and return the response. This is separated into
164         it's own function, so we can mock it easily for testing. 165
166             :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(
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):
```