

## SENDGRID INTEGRATION WITH PYTHON

DATE: 10 NOV 2022

TEAM ID: PNT2022TMID10739

PROJECT NAME: CUSTOMER CARE REGISTRY

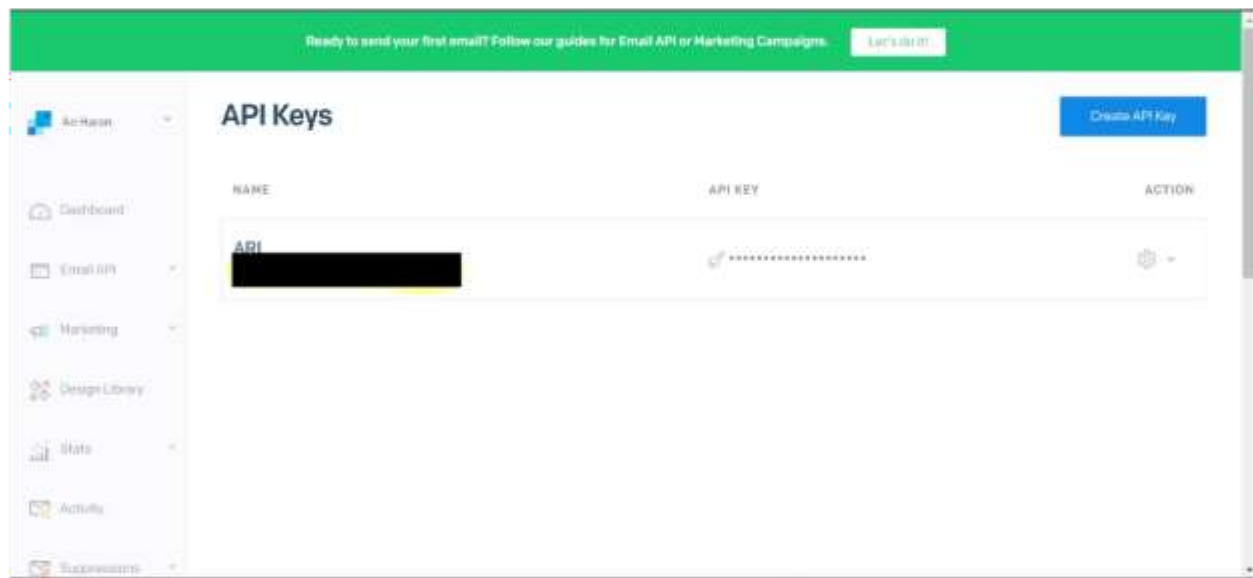
### STEP 1:

#### REQUIREMENTS:

Python 2.6, 2.7, 3.4 or 3.5.

### STEP 2:

Create an API key



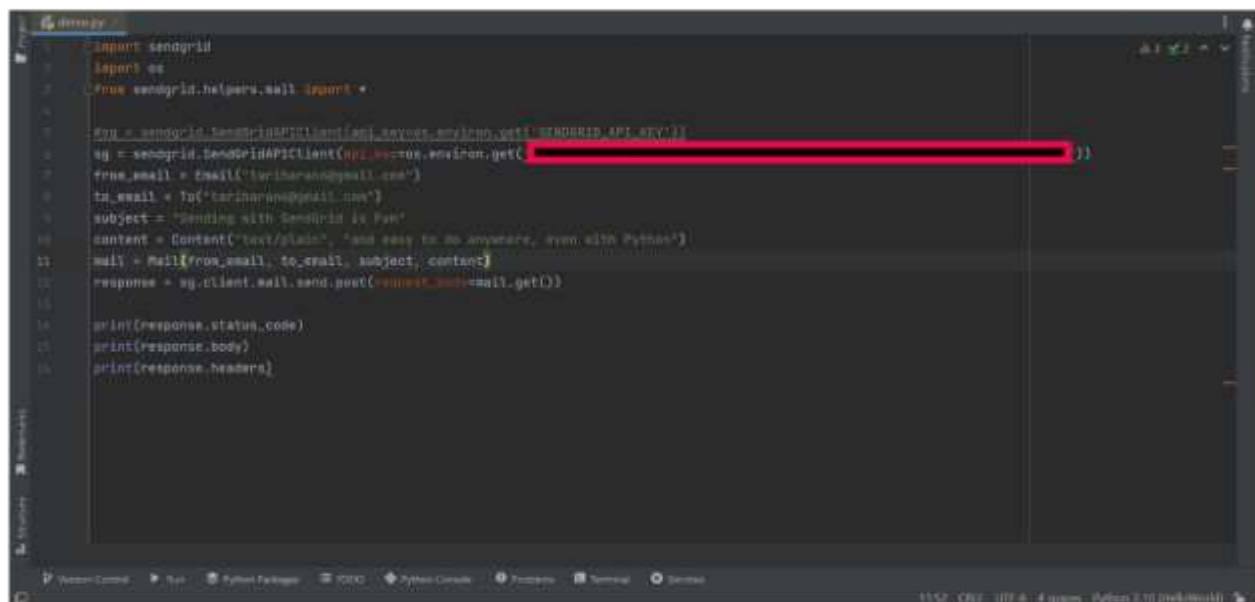
STEP 3:

INSTALL PACKAGE:

> pip install sendgrid

STEP 4:

SEND EMAIL



```
1 import sendgrid
2 import os
3 from sendgrid.helpers.mail import *
4
5 sg = sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))
6 sg = sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))
7 from_email = Email('tusharsendgrid@gmail.com')
8 to_email = To('tusharsendgrid@gmail.com')
9 subject = 'Sending with SendGrid is Fun'
10 content = Content('text/plain', 'and easy to do anywhere, even with Python')
11 mail = Mail(from_email, to_email, subject, content)
12 response = sg.client.mail.send.post(request_body=mail.get())
13
14 print(response.status_code)
15 print(response.body)
16 print(response.headers)
```

The screenshot shows a code editor with a dark theme. The code is written in Python and uses the SendGrid library to send an email. The code is as follows:

```
1 import sendgrid
2 import os
3 from sendgrid.helpers.mail import *
4
5 sg = sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))
6 sg = sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))
7 from_email = Email('tusharsendgrid@gmail.com')
8 to_email = To('tusharsendgrid@gmail.com')
9 subject = 'Sending with SendGrid is Fun'
10 content = Content('text/plain', 'and easy to do anywhere, even with Python')
11 mail = Mail(from_email, to_email, subject, content)
12 response = sg.client.mail.send.post(request_body=mail.get())
13
14 print(response.status_code)
15 print(response.body)
16 print(response.headers)
```

The code is numbered 1 through 16. The first line imports the sendgrid module. The second line imports the os module. The third line imports the Email, To, and Content classes from the sendgrid.helpers.mail module. The fourth line is a blank line. The fifth line creates a SendGridAPIClient object named sg using the API key from the environment variables. The sixth line is a duplicate of the fifth line. The seventh line creates a from\_email object. The eighth line creates a to\_email object. The ninth line sets the subject of the email. The tenth line creates a content object. The eleventh line creates a mail object. The twelfth line sends the email and stores the response in the response variable. The thirteenth line is a blank line. The fourteenth line prints the status code of the response. The fifteenth line prints the body of the response. The sixteenth line prints the headers of the response.





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
```







```

16
17 _logger =
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
import urllib2 as urllib
from urllib2 import HTTPError
from urllib import urlencode

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

```









```

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
73             host,
74             request_headers=None,
75             version=None,
76             url_path=None,
77             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 custom behavior.
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 '{}{}/{}'.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

```

:param query\_params: A dictionary of all





```

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):

```













```

148
149
150     new Client object
151
152         :param name: Name of the url segment
153         :type name: string
154         :return: A Client object
155         """ url_path = self._url_path + [name] if name else
156         self._url_path return Client(host=self.host,
157         version=self._version,
158         request_headers=self.request_headers,
159         url_path=url_path,
160         append_slash=self.append_slash,
161         timeout=self.timeout)
162
163     def _make_request(self, opener, request, timeout=None):
164         """Make the API call and return the response. This is separated into
165         it's own function, so we can mock it easily for testing. 165 :param
166         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,
177             timeout=timeout) except HTTPError as err:
178             exc = handle_error(err) exc.__cause__ = None
179             _logger.debug('{method} Response: {status}')
180

```





```

    {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
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

```







```

212
213         :return: string, version
214         """
215         self._version = args[0] return
216         self._build_client() return
217         get_version
218
219     # We have reached the end of the method chain, make the API call
220     if name in self.methods:
221         method = name.upper()
222
223         def http_request(
224             request_body=None,
225             query_params=None,
226             request_headers=None,
227             timeout=None, **_):
228             """Make the API call
229             :param timeout: HTTP request timeout. Will be propagated to urllib
230             client
231             :type timeout: float
232             :param request_headers: HTTP headers. Will be merged into
233             current client object state
234             :type request_headers: dict
235             :param query_params: HTTP query parameters
236             :type query_params: dict
237             :param request_body: HTTP request body
238             :type request_body: string or json-serializable object :param
239             kwargs:
240             :return: Response object
241             """ if
242             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

```





```

276     response = Response(
277         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
288
289     http_request
290     else:
291         # Add a segment to the URL
292         return self._(name)
293
294     def __getstate__(self):
295         return self.__dict__
296
297     def __setstate__(self, state):

```



