

SENDGRID INTEGRATION WITH PYTHON

Date	10 Nov 2022
Team ID	PNT2022TMID48235
Project Name	CUSTOMER CARE REGISTRY

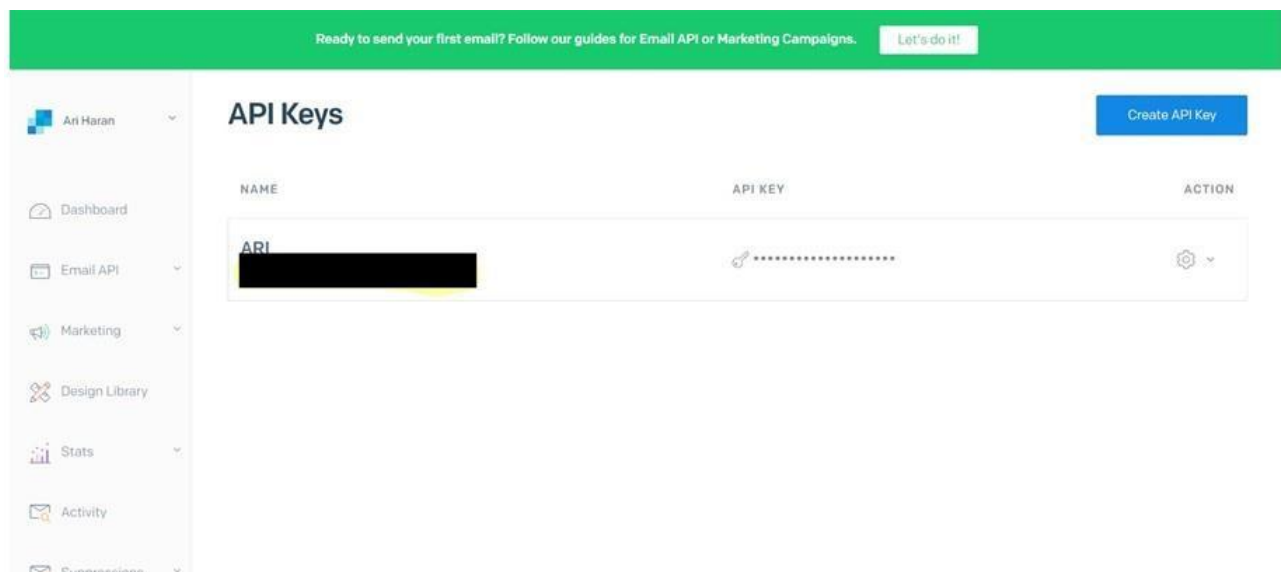
STEP 1:

REQUIREMENTS:

Python 2.6, 2.7, 3.4 or 3.5.

STEP 2:

Create an API key



STEP 3:

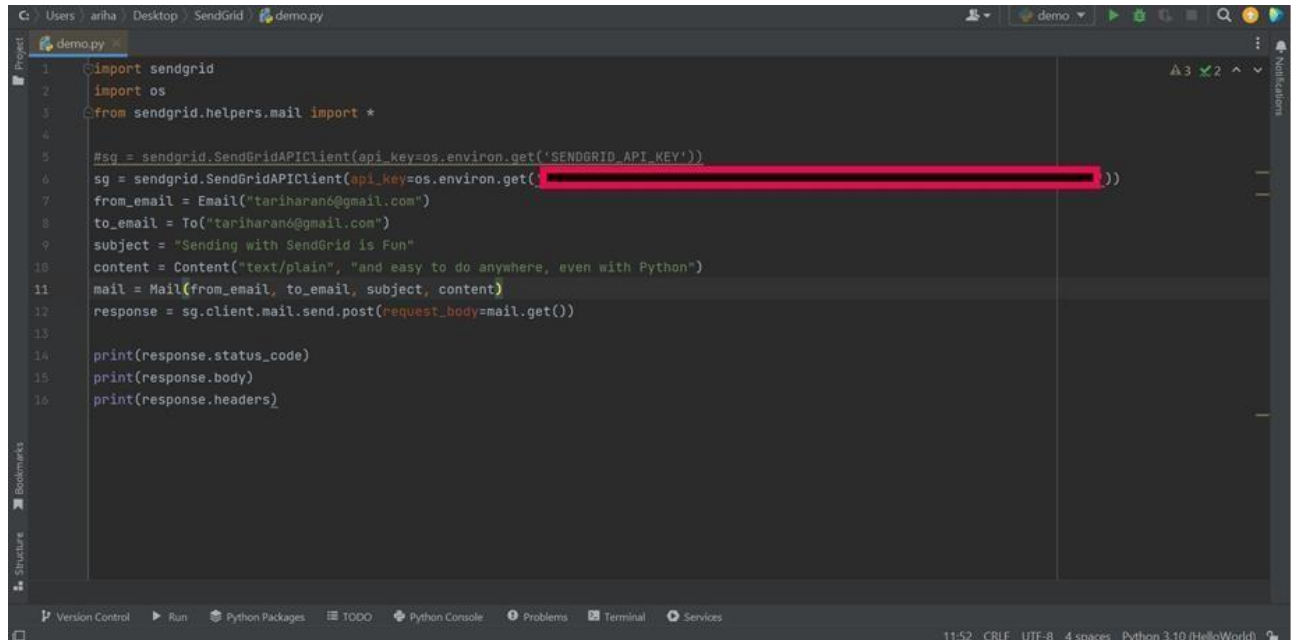
INSTALL

PACKAGE:

> pip install sendgrid

SETP 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("tariharan@gmail.com")
8 to_email = To("tariharan@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)
```

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
11 try:
12     sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
13     response = sg.send(message)
14     print(response.status_code)
15     print(response.body) 15 print(response.headers) 16 except Exception as
17 e:
18     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
9     from urllib.error import HTTPError 11 except ImportError: 12         # Python
10
11
12

```

```
import urllib2 as urllib
```

```
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
22
23         def init (self, response):
24             """
25             :param response: The return value from a
26                             open call
27                             on a urllib.build_opener()
28             :type response: urllib response object
29             """
30             self._status_code = response.getcode()
31             self._body = response.read()
32             self._headers = response.info()
33
34         @property
35         def status_code(self):
36             """
37             :return: integer, status code of API call
38             """
39             return self._status_code
40
41         @property
42         def body(self):
43             """
44             :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
```



applied on all calls :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),
110                                     url)
111
112     def _build_url(self, query_params):
```

```
112         """Build the final URL to be passed to urllib
```

```
113         :param query_params: A dictionary of all the query
114
```



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()),
130                                     True)
131
132             url = '{}?{}'.format(url, url_values)
133
134             if self._version:
135                 url = self._build_versioned_url(url)
136             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     def _build_client(self, name=None):  
147
```



```
"""Make a new Client object
```

1




```

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
155         else self._url_path
156         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
165         separated into
166         it's own function, so we can mock it easily for testing.
167
168         :param opener:
169         :type opener:
170         :param request: url payload to request
171         :type request: urllib.Request object
172         :param timeout: timeout value or None
173         :type timeout: float
174         :return: urllib response
175         """
176         timeout = timeout or self.timeout
177         try:
178             return opener.open(request, timeout=timeout)
179         except HTTPError
180         as err:
181             exc = handle_error(err)

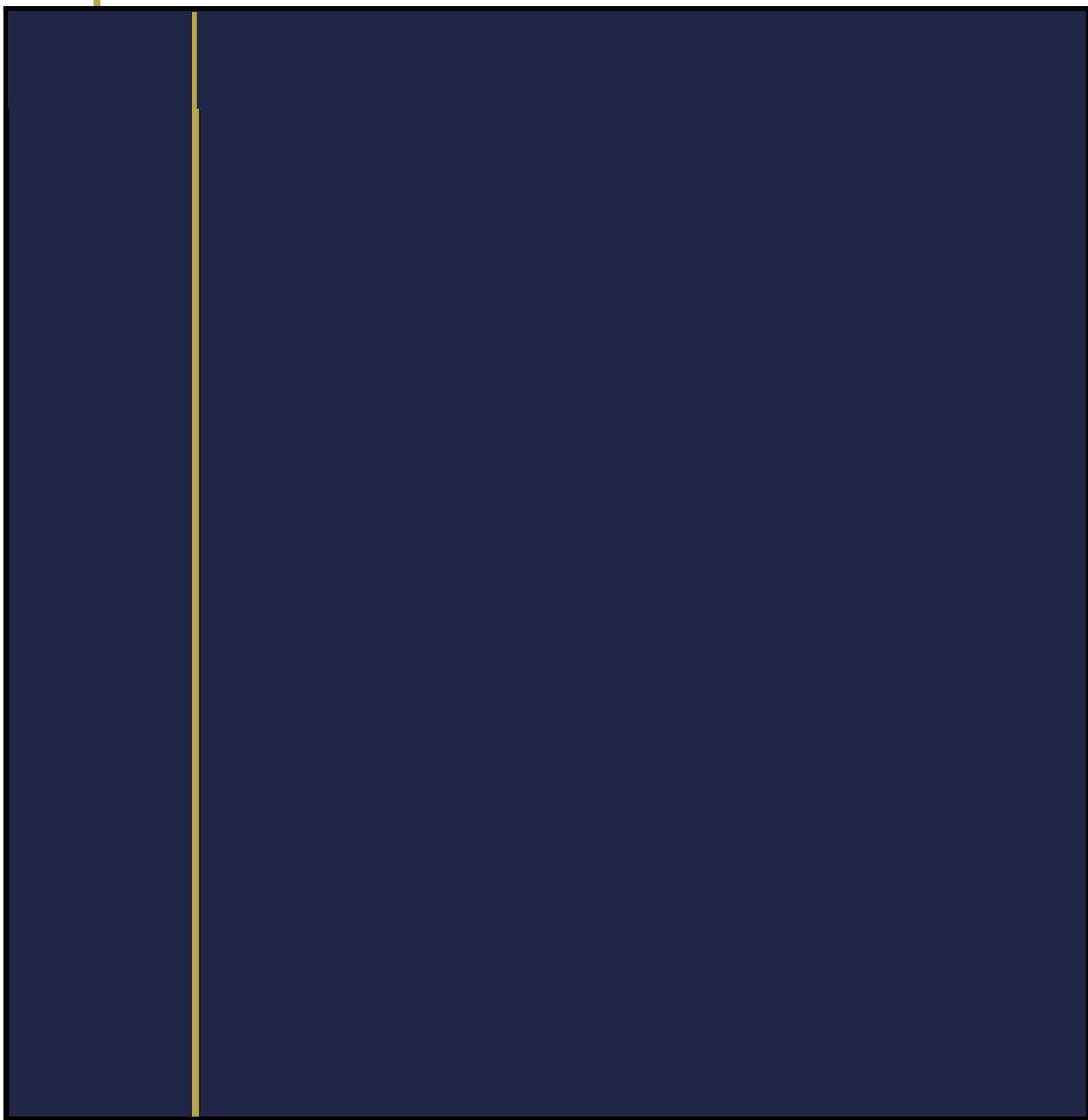
```

```
179         exc.cause_ = None
180         _logger.debug('{method} Response: {status}
```

1



```
{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,
190         such as global,
191         in your url, you must use this method.
192
193         :param name: Name of the url segment
194         :type name: string
195         :return: Client object
196         """
197         return self._build_client(name)
198
199     def getattr (self, name):
200         """Dynamically add method calls to the url, then
201         call a method.
202
203         (e.g. client.name.name.method())
204
205         You can also add a version number by using
206         .version(<int>)
207
208         :param name: Name of the url segment or method
209         call
210
211         :type name: string or integer if name ==
212         version
213
214         :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
```

```
''' '''
```



```
238             :type request_body: string or json-serializable
                object
239             :param kwargs:
240             :return: Response object
241             """
242             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))


```

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

```

