

---

**INTEGRATING WITH SENDGRID**

Date	15 Nov 2022
Team ID	PNT2022TMID44798
Project Name	SKILL/JOB RECOMMENDED APP

# SENDGRID INTEGRATION WITH PYTHON

STEP 1:

Requirements:  
Python 2.6, 2.7, 3.4 or 3.5.

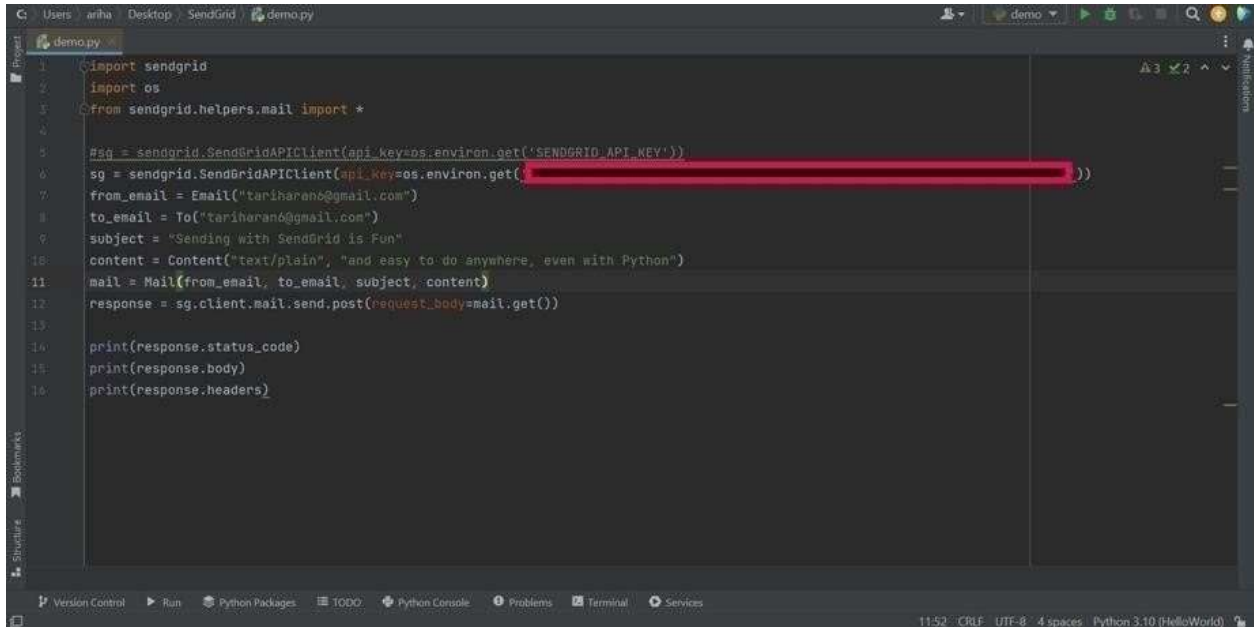
STEP 2:  
Creating an API key



STEP 3: INSTALL  
PACKAGE: > pip install sendgrid

STEP 4:

## SEND EMAIL



The image shows a Visual Studio Code editor window with a file named 'demo.py' open. The code is a Python script that uses the SendGrid API to send an email. The script includes imports for 'sendgrid' and 'os', and then uses 'sendgrid.helpers.mail' to create an email object. It sets the 'from\_email' to 'tariharano@gmail.com', the 'to\_email' to 'tariharano@gmail.com', and the 'subject' to 'Sending with SendGrid is Fun'. The 'content' is set to 'text/plain' with the body 'and easy to do anywhere, even with Python'. The script then creates a 'Mail' object and sends it using 'sg.client.mail.send.post()'. Finally, it prints the 'status\_code', 'body', and 'headers' of the response.

```
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("tariharano@gmail.com")
8 to_email = To("tariharano@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 """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
1
1 import os
2 from sendgrid import SendGridAPIClient
3 from sendgrid.helpers.mail import Mail
4
5
6     message = Mail(
7         from_email='from_email@example.com',
8
9         to_emails='to@example.com',
10
11        subject='Sending with Twilio SendGrid is Fun',
12
13        html_content='<strong>and easy to do anywhere, even with  
Python</strong>')
14
15    try:
16
17        sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
18
19        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:





```
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
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
```



```
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
```



```
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.
80             https://api.sendgrid.com)
81         :type host: string
82         :param request_headers: A dictionary of the headers you want

```

```

82             applied on all calls
83         :type request_headers: dictionary
84
85         :param version: The version number of the
86             API.
87
88         Subclass _build_versioned_url for custom
89             behavior.
90
91         Or just pass the version as part of the URL
92
93         (e.g. client._("/v3"))
94
95         :type version: integer
96
97         :param url_path: A list of the url path
98             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
104         Or just pass the version as part of the URL
105         (e.g. client._('/v3'))
106
107         :param url: URI portion of the full URL being requested
108
109         :type url: string
110
111         :return: string
```

```
109         return '{} /v{}'.format(self.host, str(self._version),
110                                   url)
111
112     def _build_url(self, query_params):
113         """Build the final URL to be passed to urllib
114
115         :param query_params: A dictionary of all the query
```

```
108
```

```
"""
```



---

---



---

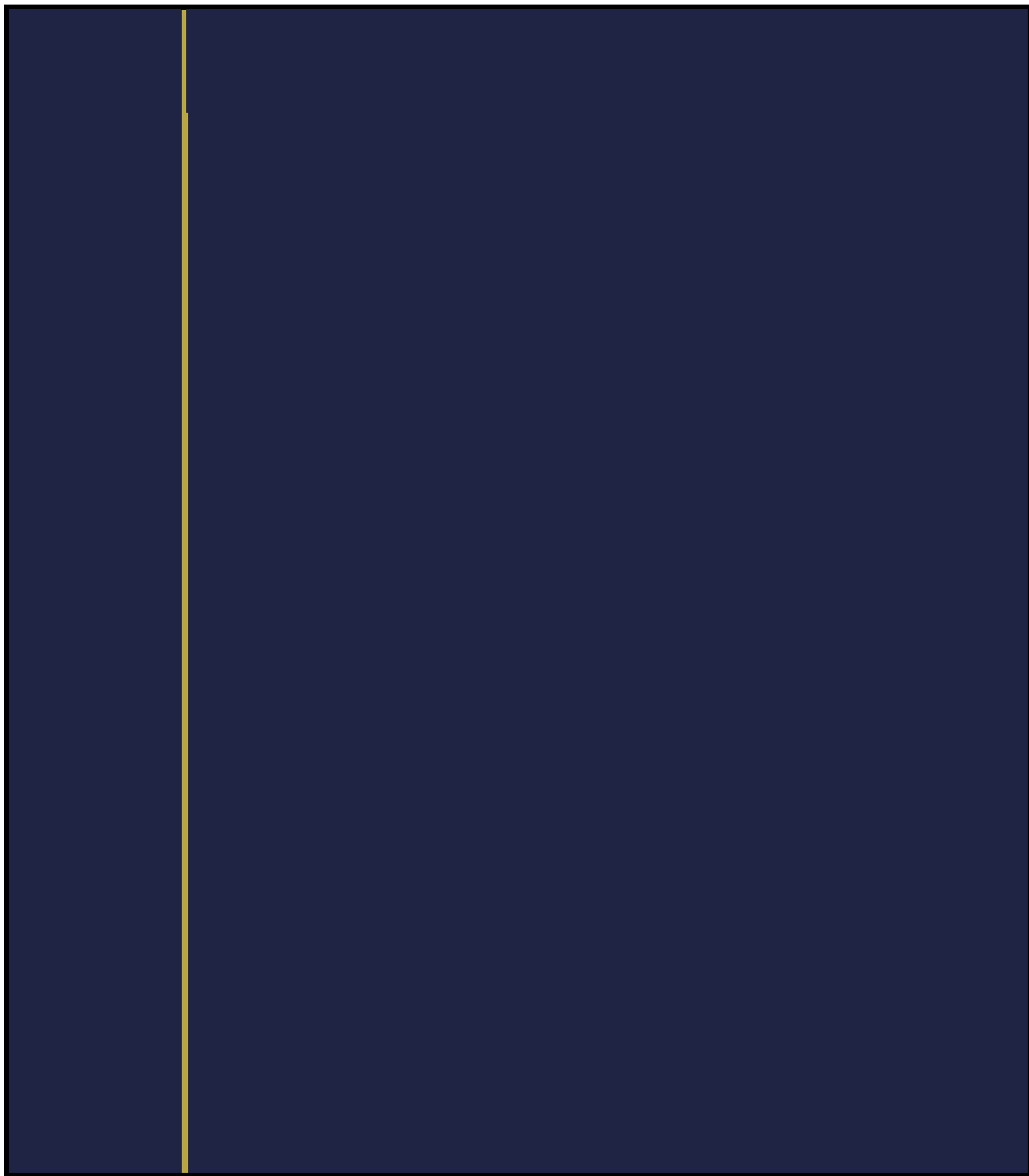
pa rameters





---

---



```
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     def _build_client(self, name=None):  
147
```



```

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,
163                         timeout=None):
164             """Make the API call and return the response.
165             This separated into
166             it's own function, so we can mock it easily for
167             is
168             testing.
169
170             :param opener:
171             :type opener:
172
173             :param request: url payload to request
174
175             :type request: urllib.Request object
176
177             :param timeout: timeout value or None
178
179             :type timeout: float
180
181             :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}
```

---

---

```
212         :return: string , version
213         """
```

```

214         self._version = args[0]

215         return self._build_client()         return
216     get_version

217
218         # We have reached the end of the method chain, make
the API call if name in self.methods:
219         method = name.upper()

220
221         def http_request(
222             request_body=None,
223             query_params=None,     request_headers=None,
224             timeout=None,
225             **_):
226             """Make the API call
227             :param timeout: HTTP request timeout. Will be
propagated to urllib client
228 229             :type timeout: float

230
231

```



```
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

239         :param kwargs:
240
241         :return: Response object
242
243         """
244
245         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 =
258                 json.dumps(request_body).encode('utf-8')
259
260         opener = urllib.build_opener()
261         request = urllib.Request(
262             self._build_url(query_params),
263             headers=self.request_headers,
264             data=data,
265         )
266         request.get_method = lambda: method
267
268         _logger.debug('[method] Request: {url}'.format(
269             method=method,
270             url=request.get_full_url()))
271         if request.data:
272             _logger.debug('PAYLOAD: {data}'.format(
273                 data=request.data))
274         _logger.debug('HEADERS: {headers}'.format(
275             headers=request.headers))
```

```

276         response = Response(
277             self, make_request(opener, request,
278                 timeout=timeout)
279         )
280         logger.debug('{method} Response: {status}
281             {body}'.format(
282                 method=method,
283                 status=response.status_code,
284                 body=response.body))
285         return response
286
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):

```