SENDGRID INTEGRATION WITH PYTHON

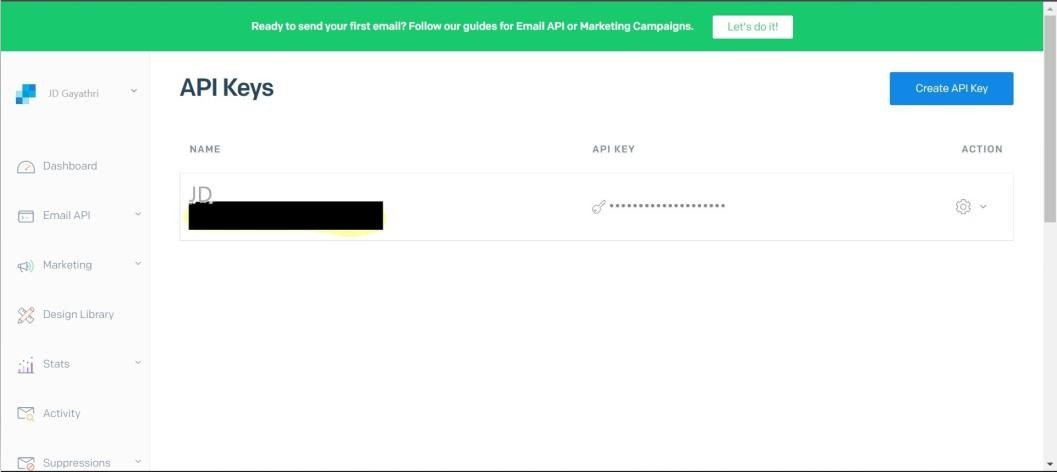
|  |  |
| --- | --- |
| Team ID | PNT2022TMID31494 |
| Project Name | CUSTOMER CARE REGISTRY |

# STEP 1: REQUIREMENTS:

Python 2.6, 2.7, 3.4 or 3.5.

# STEP 2:

Create an API key

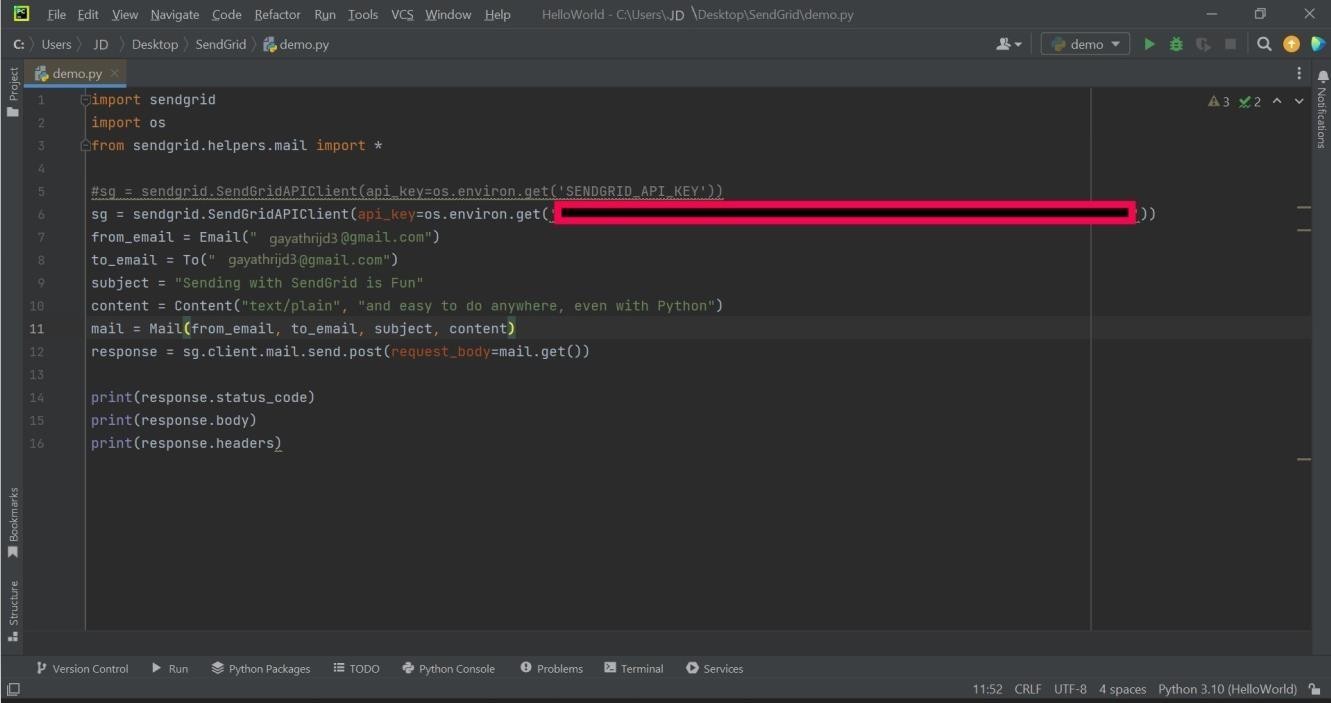


# STEP

**3**: INSTALLPACKAGE: > pip install sendgíid

# STEP 4:

SEND EMAIL



SENDGRID PYTHON CODE :

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**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>')**

1. **try:**
2. **sg = SendGridAPIClient(os.environ.get('SENDGRID\_API\_KEY'))**
3. **response = sg.send(message)**
4. **print(response.status\_code)**
5. **print(response.body)**
6. **print(response.headers)**
7. **except Exception as e:**
8. **print(e.message)**

HTTP CLIENT PROGRAM:

1

2

3

4

5

6

7

8

9

10

"""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

1. except ImportError:
2. # Python 2
3. import urllib2 as urllib
4. from urllib2 import HTTPError
5. from urllib import urlencode 16

17 \_logger = logging.getLogger( name ) 18

19

1. class Response(object):
2. """Holds the response from an API call.""" 22

23 def init (self, response):

24 """

1. :param response: The return value from a open call
2. on a urllib.build\_opener()
3. :type response: urllib response object 28 """
4. self.\_status\_code = response.getcode()
5. self.\_body = response.read()
6. self.\_headers = response.info() 32

33 @property

34

35

36

37

38

39

40

41

42

43

44

45

46

47

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

1. @property
2. def to\_dict(self):

56 """

57 :return: dict of response from the API 58 """

1. if self.body:
2. return json.loads(self.body.decode('utf-8'))
3. else:
4. return None 63

64

1. class Client(object):
2. """Quickly and easily access any REST or REST-like API.""" 67

68 # These are the supported HTTP verbs

69

70

71

72

73

74

75

76

77

78

79

methods = {'delete', 'get', 'patch', 'post', 'put'}

def init (self,

host, request\_headers=None, version=None, url\_path=None, append\_slash=False, timeout=None):

"""

:param host: Base URL for the api. (e.g.

https://api.sendgrid.com)

80

81

:type host:

string

:param request\_headers: A dictionary of the headers you want

1. applied on all calls
2. :type request\_headers: dictionary
3. :param version: The version number of the API.
4. Subclass \_build\_versioned\_url for custom behavior.
5. Or just pass the version as part of the URL

87 (e.g. client.\_("/v3"))

1. :type version: integer
2. :param url\_path: A list of the url path segments
3. :type url\_path: list of strings

91 """

1. self.host = host
2. self.request\_headers = request\_headers or {}
3. self.\_version = version
4. # \_url\_path keeps track of the dynamically built url
5. self.\_url\_path = url\_path or []
6. # APPEND SLASH set
7. self.append\_slash = append\_slash
8. self.timeout = timeout 100

101 def \_build\_versioned\_url(self, url):

102

103

104

105

106

107

108

109

url) 110

111

112

113

114

"""Subclass this function for your own needs. Or just pass the version as part of the URL (e.g. client.\_('/v3'))

:param url: URI portion of the full URL being requested

:type url: string

:return: string """

return '{}/v{}{}'.format(self.host, str(self.\_version),

def \_build\_url(self, query\_params):

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

:param query\_params: A dictionary of all the query

parameters

1. :type query\_params: dictionary
2. :return: string

117 """

1. url = ''
2. count = 0
3. while count < len(self.\_url\_path):
4. url += '/{}'.format(self.\_url\_path[count])
5. count += 1
6. # add slash
7. if self.append\_slash:

126 url += '/'

127

1. if query\_params:
2. url\_values = urlencode(sorted(query\_params.items()), True)
3. url = '{}?{}'.format(url, url\_values) 131
4. if self.\_version:
5. url = self.\_build\_versioned\_url(url)
6. else:
7. url = '{}{}'.format(self.host, url)
8. return url 137
9. def \_update\_headers(self, request\_headers):
10. """Update the headers for the request 140
11. :param request\_headers: headers to set for the API call
12. :type request\_headers: dictionary
13. :return: dictionary

144 """

145 self.request\_headers.update(request\_headers) 146

147 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  separated 164 | """Make the API call and return the response. This into  it's own function, so we can mock it easily for | is  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(

1. method=request.get\_method(),
2. status=exc.status\_code,
3. body=exc.body))
4. raise exc
5. def \_(self, name):
6. """Add variable values to the url.
7. (e.g. /your/api/{variable\_value}/call)
8. Another example: if you have a Python reserved word, such as global,
9. in your url, you must use this method. 191
10. :param name: Name of the url segment
11. :type name: string
12. :return: Client object

195 """

196 return self.\_build\_client(name) 197

1. def getattr (self, name):
2. """Dynamically add method calls to the url, then call a method.
3. (e.g. client.name.name.method())
4. You can also add a version number by using

.version(<int>) 202

1. :param name: Name of the url segment or method call
2. :type name: string or integer if name == version
3. :return: mixed

206 """

1. if name == 'version':
2. def get\_version(\*args, \*\*kwargs):

209 """

1. :param args: dict of settings
2. :param kwargs: unused
3. :return: string, version

213 """

1. self.\_version = args[0]
2. return self.\_build\_client()
3. return get\_version 217
4. # We have reached the end of the method chain, make the API call
5. if name in self.methods:
6. method = name.upper() 221
7. def http\_request(
8. request\_body=None,
9. query\_params=None,
10. request\_headers=None,
11. timeout=None,

227 \*\*\_):

1. """Make the API call
2. :param timeout: HTTP request timeout. Will be propagated to
3. urllib client

merged into 233

234

235

236

237

238

object 239

240

241

242

:type timeout: float

:param request\_headers: HTTP headers. Will be

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

: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 \ |

1. self.request\_headers['Content-Type'] !=

\

1. 'application/json':
2. data = request\_body.encode('utf-8')
3. else:
4. self.request\_headers.setdefault(
5. 'Content-Type', 'application/json')
6. data =

json.dumps(request\_body).encode('utf-8') 258

1. opener = urllib.build\_opener()
2. request = urllib.Request(
3. self.\_build\_url(query\_params),

262

263

264

265

266

267

268

269

270

271

272

273

274

275

headers=self.request\_headers,

data=data,

)

request.get\_method = lambda: method

\_logger.debug('{method} Request: {url}'.format( method=method, url=request.get\_full\_url()))

if request.data:

\_logger.debug('PAYLOAD: {data}'.format( data=request.data))

\_logger.debug('HEADERS: {headers}'.format(

headers=request.headers))

276

277

timeout=timeout) 278

279

280

{body}'.format( 281

282

283

284

285

286

response = Response( self.\_make\_request(opener, request,

)

\_logger.debug('{method} Response: {status}

method=method, status=response.status\_code, body=response.body))

return response

1. return http\_request
2. else:
3. # Add a segment to the URL
4. return self.\_(name) 291
5. def getstate (self):
6. return self. dict 294

295

def setstate (self, state):