DATE:	10 NOV 2022
TEAM ID:	PNT2022TMID42344
PROJECT NAME:	CUSTOMER CARE REGISTRY

STEP 1:

REQUIREMENTS:

Python 2.6, 2.7, 3.4 or 3.5.

STEP 2:

Create an API key

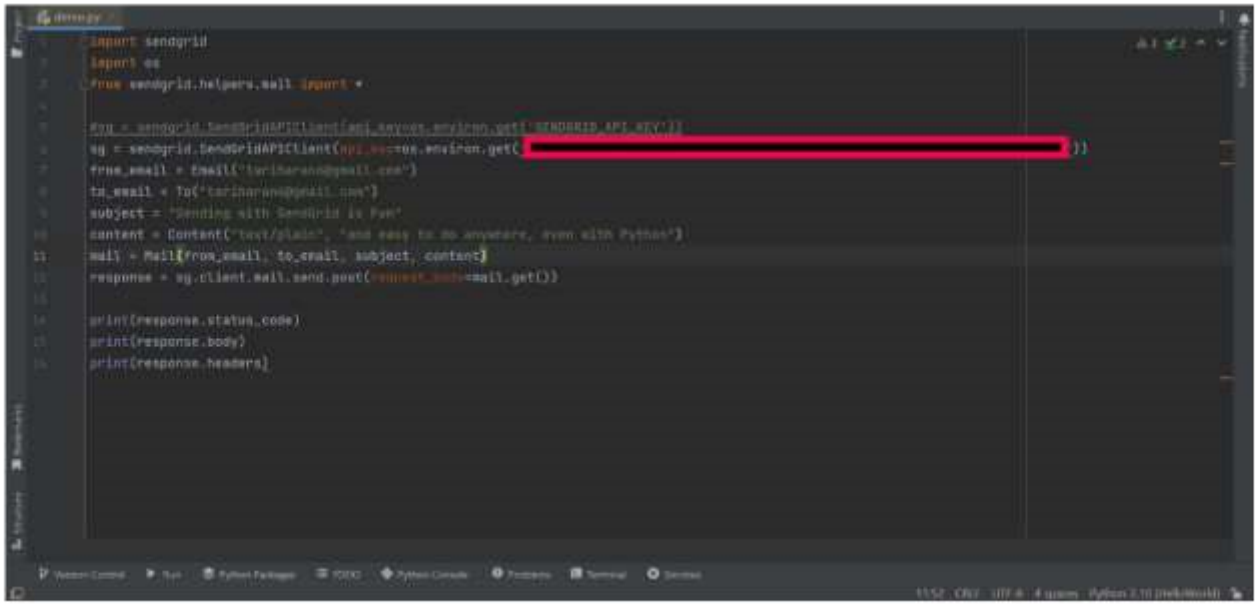
STEP 3:

INSTALL PAKAGE:

> pip install sendgrid

SETP 4:

SEND EMAIL



SENDGRID PYTHON CODE :

```
import os
from sendgrid import SendGridAPIClient 3 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>')
try:
sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
response = sg.send(message) 13    print(response.status_code)
print(response.body)
print(response.headers)  16            except Exception as e:
  print(e.message)
```

HTTP CLIENT PROGRAM:

```
"""HTTP Client library"""
import json

import logging

from .exceptions import handle_error


    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
        def status_code(self):
  34    """
  35

        """
  37

  38    return self._status_code  39
40        @property
        def body(self):
41      """
42

        """
44

45        return self._body

46
```

```
47  @property
```

```
48  def headers(
49      """
50          self):
51      """
52      :return: dict of response headers
53      """
54      return self._headers
55  @property
56  def to_dict(self):
57      """
58      :return: dict of response from the API
59      """
60      if self.body:
61          return json.loads(self.body.decode('utf-8'))
62      else:
63          return None
64
65
66
67  class Client(object):
68      """Quickly and easily access any REST or REST-like API."""
69      # These are the supported HTTP verbs
70      methods = {'delete', 'get', 'patch', 'post', 'put'}
71
72  def __init__(self,
73          host,
74          request_headers=None,
75          version=None,
76          url_path=None,
77          append_slash=False,
78          timeout=None):
79      """
80      :param host: Base URL for the api. (e.g. https://api.sendgrid.com)
81          :type host: string
          :param request_headers: A dictionary of the headers you want
```

```
        applied on all calls
        :type request_headers: dictionary
        :param version: The version number of the API.
        Subclass _build_versioned_url for custom behavior.
        Or just pass the version as part of the URL
        (e.g. client._("/v3"))
        :type version: integer
        :param url_path: A list of the url path segments
        :type url_path: list of strings
        """
        self.host = host
        self.request_headers = request_headers or {}
        self._version = version
        # _url_path keeps track of the dynamically built url
        self._url_path = url_path or []
        # APPEND SLASH set
        self.append_slash = append_slash
        self.timeout = timeout

    def _build_versioned_url(self, url):
        """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), url)

    def _build_url(self, query_params):
        """Build the final URL to be passed to urllib
```

```
:param query_params: A dictionary of all
```

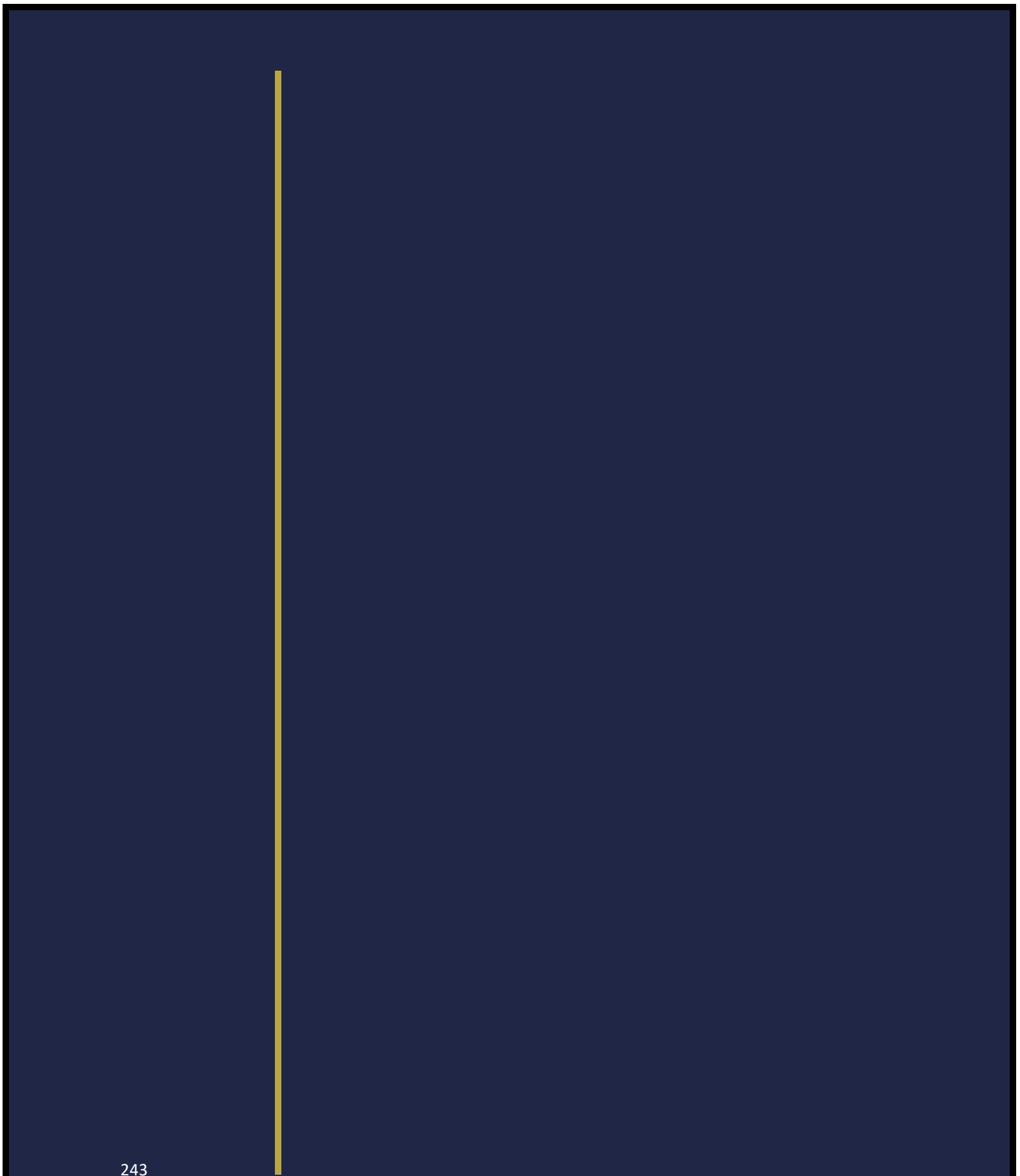
```
parameters
115         :type query_params: dictionary
116         :return: string
                117     """
                118     url     = ''
119         count = 0
120         while count < len(self._url_path):
                121     url     += '/{}'.format(self._url_path[count])
122         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 [redacted]
146
147   def _build_client(self,       name=None):
```

```
148
149         new Client object
150
151                         :param name: Name of the url segment
152                         :type name: string
153
154                         :return: A Client object
                            """  url_path = self._url_path + [name] if name else
155                         self._url_path          return     Client(host=self.host,
156                         version=self._version,
157
158                         request_headers=self.request_headers,
159
160                         url_path=url_path,
                            append_slash=self.append_slash,
                            timeout=self.timeout)
162
161
163     def _make_request(self, opener, request, timeout=None):
        """Make the API call and return the response. This is separated into
164
166     it's own function, so we can mock it easily for testing. 165  :param
167     opener:
168
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,
179         timeout=timeout)  except HTTPError as err:
180         exc = handle_error(err)  exc.__cause__ = None
            _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
```

```
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
```

```
        :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):
```