**INTEGRATION SENDGRID SERVICE**

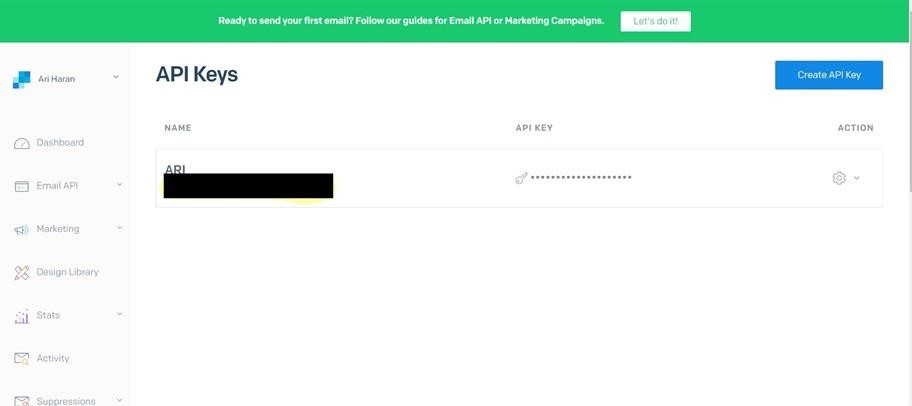
**STEP 1:**

**REQUIREMENTS:**

**Python 2.6, 2.7, 3.4 or 3.5.**

**STEP 2:**

Create an API key



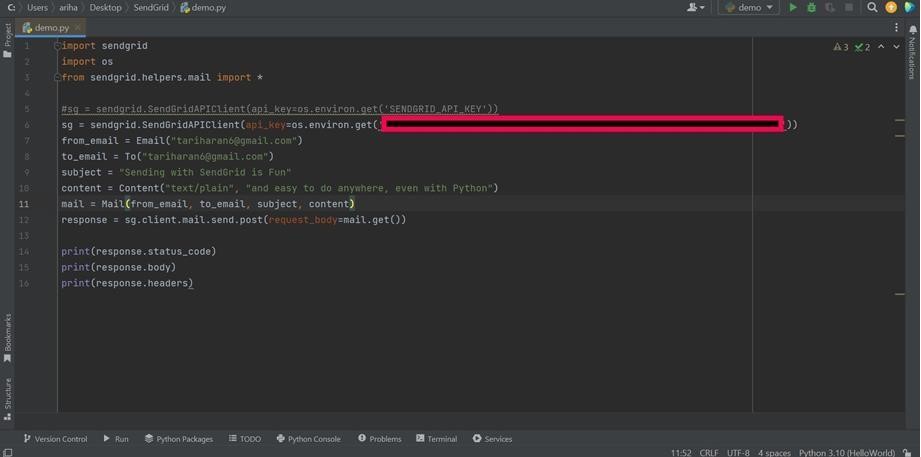
STEP 3:

INSTALL

PAKAGE: > pip install sendgrid

SETP 4:

SEND EMAIL



**SENDGRID PYTHON CODE :**

|  |
| --- |
| 1. """HTTP Client library""" 2. import json 3. import logging 4. from .exceptions import handle\_error   5   1. try: 2. # Python 3 3. import urllib.request as urllib 4. from urllib.parse import urlencode 5. from urllib.error import HTTPError 11 except ImportError:   12 # Python 2 |

|  |
| --- |
| 1. **import os** 2. **from sendgrid import SendGridAPIClient 3 from sendgrid.helpers.mail import Mail**   **4**   1. **message = Mail(** 2. **from\_email='from\_email@example.com',** 3. **to\_emails='to@example.com',** 4. **subject='Sending with Twilio SendGrid is Fun',** 5. **html\_content='<strong>and easy to do anywhere, even with**   **Python</strong>') 10 try:**   1. **sg = SendGridAPIClient(os.environ.get('SENDGRID\_API\_KEY'))** 2. **response = sg.send(message)** 3. **print(response.status\_code)** 4. **print(response.body) 15 print(response.headers) 16 except Exception as e:**   **17 print(e.message)** |

**HTTP CLIENT PROGRAM:**

|  |
| --- |
| 1. import urllib2 as urllib 2. from urllib2 import HTTPError 3. 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 3. def init (self, response): 4. """ 5. :param response: The return value from a open call 6. on a urllib.build\_opener() 7. :type response: urllib response object 8. """ 9. self.\_status\_code = response.getcode() 10. self.\_body = response.read() 11. self.\_headers = response.info()   32  33 @property |
| 1. def status\_code(self): 2. """ 3. :return: integer, status code of API call 4. """ 5. return self.\_status\_code   39   1. @property 2. def body(self): 3. """ 4. :return: response from the API 5. """ 6. return self.\_body     46  47 @property |

|  |
| --- |
| 1. def headers(self): 2. """ 3. :return: dict of response headers 4. """ 5. return self.\_headers   53   1. @property 2. def to\_dict(self): 3. """ 4. :return: dict of response from the API 5. """ 6. if self.body: 7. return json.loads(self.body.decode('utf-8')) 8. else: 9. 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 methods = {'delete', 'get', 'patch', 'post', 'put'} 70   1. def init (self, 2. host, 3. request\_headers=None, 4. version=None, 5. url\_path=None, 6. append\_slash=False, 77 timeout=None): 7. """ 8. :param host: Base URL for the api. (e.g.   https://api.sendgrid.com)   1. :type host: string 2. :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 6. (e.g. client.\_("/v3")) 7. :type version: integer 8. :param url\_path: A list of the url path segments 9. :type url\_path: list of strings 10. """ 11. self.host = host 12. self.request\_headers = request\_headers or {} 13. self.\_version = version 14. # \_url\_path keeps track of the dynamically built url 15. self.\_url\_path = url\_path or [] 16. # APPEND SLASH set 17. self.append\_slash = append\_slash 18. self.timeout = timeout   100  101 def \_build\_versioned\_url(self, url): |
| 1. """Subclass this function for your own needs. 2. Or just pass the version as part of the URL 3. (e.g. client.\_('/v3')) 4. :param url: URI portion of the full URL being requested 5. :type url: string 6. :return: string 7. """ 8. return '{}/v{}{}'.format(self.host, str(self.\_version), url)   110   1. def \_build\_url(self, query\_params): 2. """Build the final URL to be passed to urllib     113  114 :param query\_params: A dictionary of all the query |

|  |
| --- |
| parameters   1. :type query\_params: dictionary 2. :return: string 3. """ 4. url = '' 5. count = 0 6. while count < len(self.\_url\_path): 7. url += '/{}'.format(self.\_url\_path[count]) 8. count += 1   123   1. # add slash 2. if self.append\_slash: 3. url += '/'   127   1. if query\_params: 2. url\_values = urlencode(sorted(query\_params.items()),   True)   1. url = '{}?{}'.format(url, url\_values)   131   1. if self.\_version: 2. url = self.\_build\_versioned\_url(url) 3. else: 4. url = '{}{}'.format(self.host, url) 5. return url   137   1. def \_update\_headers(self, request\_headers): 2. """Update the headers for the request   140   1. :param request\_headers: headers to set for the API call 2. :type request\_headers: dictionary 3. :return: dictionary 4. """ 5. self.request\_headers.update(request\_headers)   146  147 def \_build\_client(self, name=None): |

|  |
| --- |
| 148 """Make a new Client object  149   1. :param name: Name of the url segment 2. :type name: string 3. :return: A Client object 4. """ 5. url\_path = self.\_url\_path + [name] if name else self.\_url\_path 6. return Client(host=self.host,      1. version=self.\_version, 2. request\_headers=self.request\_headers, 3. url\_path=url\_path, 4. append\_slash=self.append\_slash, 5. timeout=self.timeout)   161   1. def \_make\_request(self, opener, request, timeout=None): 2. """Make the API call and return the response. This is   separated into   1. it's own function, so we can mock it easily for testing.   165  166 :param opener: 167 :type opener:   1. :param request: url payload to request 2. :type request: urllib.Request object 3. :param timeout: timeout value or None 4. :type timeout: float 5. :return: urllib response 6. """ 7. timeout = timeout or self.timeout 8. try: 9. return opener.open(request, timeout=timeout) 10. except HTTPError as err: 11. exc = handle\_error(err) 12. exc. cause = None 13. \_logger.debug('{method} Response: {status} |

|  |
| --- |
| {body}'.format(   1. method=request.get\_method(), 2. status=exc.status\_code, 3. body=exc.body)) 4. raise exc   185   1. def \_(self, name): 2. """Add variable values to the url. 3. (e.g. /your/api/{variable\_value}/call) 4. Another example: if you have a Python reserved word, such as global, 5. in your url, you must use this method.   191   1. :param name: Name of the url segment 2. :type name: string 3. :return: Client object 4. """ 5. 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 4. """ 5. if name == 'version': 6. def get\_version(\*args, \*\*kwargs): 7. """ 8. :param args: dict of settings 9. :param kwargs: unused |

1. :return: string, version
2. """
3. self.\_version = args[0]

|  |
| --- |
| 1. return self.\_build\_client() 2. return get\_version   217   1. # We have reached the end of the method chain, make the API call 2. if name in self.methods: 3. method = name.upper()   221   1. def http\_request( 2. request\_body=None, 3. query\_params=None, 4. request\_headers=None, 5. timeout=None, 6. \*\*\_): 7. """Make the API call 8. :param timeout: HTTP request timeout. Will be propagated to 9. urllib client |
| 1. :type timeout: float 2. :param request\_headers: HTTP headers. Will be merged into 3. current client object state 4. :type request\_headers: dict 5. :param query\_params: HTTP query parameters 6. :type query\_params: dict 7. :param request\_body: HTTP request body 8. :type request\_body: string or json-serializable     object   1. :param kwargs: 2. :return: Response object 3. """ 4. if request\_headers: |

243 self.\_update\_headers(request\_headers)

244

245 if request\_body is None:

|  |
| --- |
| 246 data = None 247 else:   1. # Don't serialize to a JSON formatted str 2. # if we don't have a JSON Content-Type 3. if 'Content-Type' in self.request\_headers and \ 4. 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), |
| 1. headers=self.request\_headers, 2. data=data, 3. ) 4. request.get\_method = lambda: method     266   1. \_logger.debug('{method} Request: {url}'.format( 2. method=method, 3. url=request.get\_full\_url())) 270 if request.data: 4. \_logger.debug('PAYLOAD: {data}'.format( 5. data=request.data)) 6. \_logger.debug('HEADERS: {headers}'.format( 7. headers=request.headers))   275 |

1. response = Response(
2. self.\_make\_request(opener, request, timeout=timeout)

|  |
| --- |
| 278 )    279   1. \_logger.debug('{method} Response: {status}   {body}'.format(   1. method=method, 2. status=response.status\_code, 3. body=response.body))   284  285 return response  286  287 return http\_request 288 else:   1. # Add a segment to the URL 2. return self.\_(name)   291   1. def getstate (self): 2. return self. dict   294 |
| 295 def setstate (self, state): |