Date	13 NOV 2022
Team ID	PNT2022TMID12243
Project Name	NUTRITION ASSISTANT APPLICATION

SENDGRID PYTHON CODE:

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

HTTP CLIENT PROGRAM

```
"""HTTP Client library"""
 import json
 import logging
from .exceptions import handle_error
# Python 3
import urllib.request as urllib
from urllib.parse import urlencode
from urllib.error import HTTPError
 except ImportError:
# Python 2
try:
 _logger = logging.getLogger (_name_)
 class Response (object):
import urllib2 as urllib
from urllib2 import HTTPError
from urllib import urlencode
"""Holds the response from an API call."""
def init (self, response) :
:param response: The return value from a open call
on a urllib.build_opener ()
urllib response object
:type response:
self._status_code = response.getcode ()
self._body = response.read()
self._headers = response.info()
```

```
@property

def status_code (self):

1111#

:return: integer, status code of API call

11 || #

return self._status_code

@property

def body (self):

1111#

:return: response from the API

1111#

return self._body

@property

def headers (self): :

return: dict of response headers

“””

return self._headers

 @property

 def to_dict (self):

 “”” :return: dict of response from the API

“””

 if self.body:

 return json.loads (self.body.decode('utf-8'))

else:

 return None 65
 class Client (object):

"""

Quickly and easily access any REST or REST-like API.

"""

 # These are the supported HTTP verbs

 methods 'delete', 'get', 'patch', 'post', 'put'} = init (self, def

 “””

 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) :type

host: string :param request_headers:

A dictionary of the headers you want


behavior.

url)

applied on all calls

:type request_headers: dictionary

:param version: The version number of the API. Subclass _build_versioned_url for custom

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),

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

:type query_params: dictionary :return: string

url = !!

count = 0

while count < len (self._url_path):

url+'/'.format(self._url_path[count])

count += 1

# add slash

True)

if self.append_slash:

url += '/'

if query params:

url_values = urlencode(sorted (query_params.items()),

url = '{}?{}'.format(url, url_values)

if self._version:
```

```
else:

url = self._build_versioned_url (url)

url = '{}{}'.format(self.host, url) return url


def _update_headers (self, request_headers): """Update the headers for the request

:param request_headers: headers to set for the API call :type request_headers: dictionary

:return: dictionary

self.request_headers.update (request_headers)

def _build_client (self, name=None):

"Make a new Client object

:param name: Name of the url segment

:type name: string

:return: A Client object

url_path

=

self._url_path

self._url_path + [name] if name else

return Client (host=self.host,

version=self._version,

request_headers-self.request_headers,

url path-url_path,

append_slash=self.append slash, timeout=self.timeout)

def _make_request (self, opener, request, timeout=None): """Make the API call and return the
response. This is

separated into

it's own function, so we can mock it easily for testing.
```

```
:param opener:

:type opener:

:param request: url payload to request :type request: urllib.Request object

:param timeout: timeout value or None :type timeout: float

:return: urllib response

www

timeout timeout or self.timeout

try:

return opener.open (request, timeout=timeout)

except HTTPError as err:

exc= handle e_error(err)

exc. cause = None

_logger.debug('(method) Response: (status)


(body)'.format(

method=request.get_method (),

status=exc.status_code,

body-exc.body))

raise exc

def _(self, name):

"""Add variable values to the url.

(e.g. /your/api/(variable_value)/call)

Another example: if you have a Python reserved word,

such as global,

def
```

method.

in your url, you must use this method.

:param name: Name of the url segment :type name: string

:return: Client object

return self._build_client (name)

_getattr(self, name):

"""Dynamically add method calls to the url, then call a

 (e.g. client.name.name.method())

You can also add a version number by using

.version (<int>)

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

if name == 'version':

def get_version (*args, **kwargs):

www

:param args: dict of settings :param kwargs: unused