

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



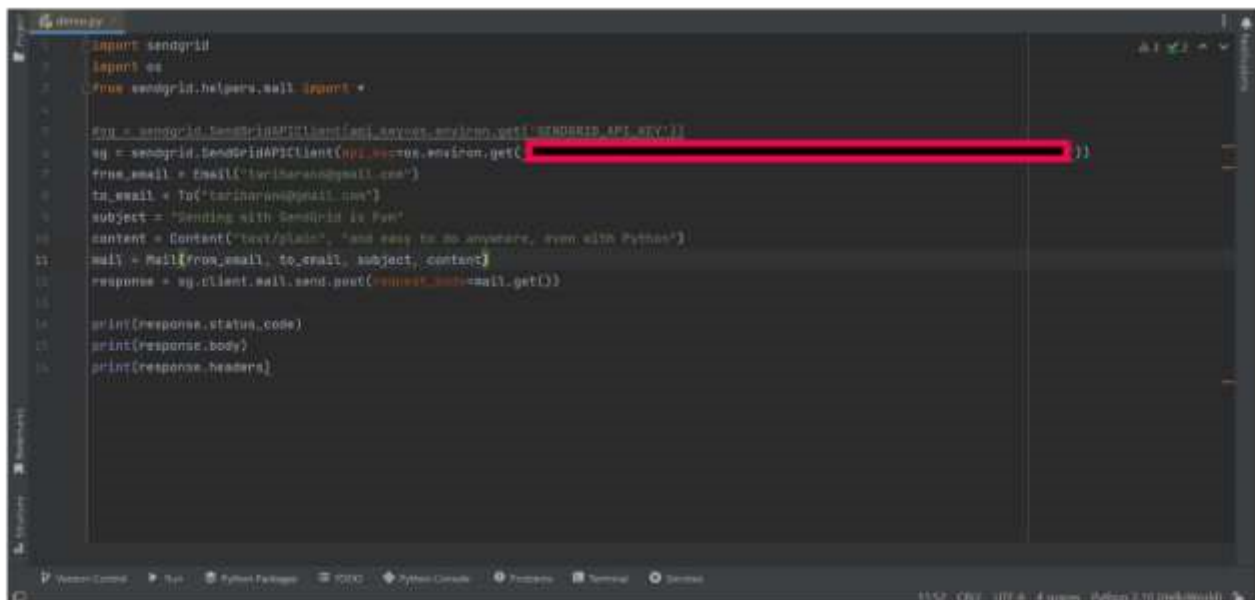
STEP 3:

INSTALL PACKAGE:

> pip install sendgrid

STEP 4:

SEND EMAIL



```
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('carinarand@gmail.com')
8 to_email = To('carinarand@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)
```

The screenshot shows a code editor with a dark theme. The code is written in Python and uses the SendGrid library to send an email. A red rectangle highlights the line `sg = sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))`. The editor has a sidebar on the left with icons for Explorer, Search, and Run and Debug. The bottom status bar shows the file path `C:\Users\carinarand\AppData\Local\Programs\Python\Python38-64\Scripts\python.exe` and the file name `send_email.py`.

SENDGRID PYTHON CODE :

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

```
1 """HTTP Client library"""
2 import json
3
4 import logging
5
6 from .exceptions import handle_error
7
8 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 =
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
34     def status_code(self):
35         """
36         :return: integer, status code of API call
37
38         return self._status_code 39
39
40     @property
41     def body(self):
42         """
43         :return: response from the API
44
45         return self._body

```



```

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

```



```

100
    82
    83
    84
    85
    86
    87
    88
    89 90
    91
    92
110  93
    94
    95
113  96
114  97
    98
    99

    101
    def
    _build_versioned_url(self
    """Subclass this function for
    Or just pass the version as p
    (e.g. client._('/v3'))
    :param url: URI portion of th
    :type url: string
    :return: string
    """
    return '{}{}/v{}'.format(self.h
    s): d to
    urllib
111 def _build_url(self, query_param
112 """Build the final URL to be passe

```

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
 , url): your
 own needs. art
 of the URL
 e full URL being requested
 :param url: URI portion of th
 :type url: string
 :return: string
 """
 return '{}{}/v{}'.format(self.h
 s): d to
 urllib
 def _build_url(self, query_param
 """Build the final URL to be passe

: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         :param request_headers: headers to set for the API call
141         :type request_headers: dictionary
142         :return: dictionary
143         """
144
145     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         :return: A Client object
154         """ url_path = self._url_path + [name] if name else
155         self._url_path      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
163     def _make_request(self, opener, request, timeout=None):
164         """Make the API call and return the response. This is separated into
165         it's own function, so we can mock it easily for testing. 165 :param
166         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,
177             timeout=timeout) except HTTPError as err:
178             exc = handle_error(err) exc.__cause__ = None
179             _logger.debug('{method} Response: {status}

```



```

191
186 def
187
188
189
190
191
192
193
194
195
196 return
197
198 def
199
200
201
202
203
204
205
206
207
208
209
210
211
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
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000


```

{body}'.format(
 181 method=request.get_method(),
 182 status=exc.status_code,
 183 body=exc.body))
 184 raise exc 185
 186 def
 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,
 191 you must use this
 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
 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
 202 .version(<int>) 203
 204 :param name: Name of the url segment or method call
 205 :type name: string or integer if name == version
 206 :return: mixed
 207 if
 208 def
 209 get_version(*args, **kwargs): 209
 210 :param args: dict of
 211 :param kwargs:
 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
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 580
 581
 582
 583
 584
 585
 586
 587
 588
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 780
 781
 782
 783
 784
 785
 786
 787
 788
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 880
 881
 882
 883
 884
 885
 886
 887
 888
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 1000


```

212
213
214         :return: string, version
215         """
216         self._version = args[0] return
217         self._build_client() return
218         get_version
219
220     # We have reached the end of the method chain, make the API call
221     if name in self.methods: method = name.upper()
222
223     def http_request(
224         request_body=None,
225         query_params=None,
226         request_headers=None,
227         timeout=None, **_):
228         """Make the API call
229         :param timeout: HTTP request timeout. Will be propagated to urllib client
230         :type timeout: float
231         :param request_headers: HTTP headers. Will be merged into current
232         client object state
233         :type request_headers: dict
234         :param query_params: HTTP query parameters
235         :type query_params: dict
236         :param request_body: HTTP request body
237         :type request_body: string or json-serializable object :param kwargs:
238         :return: Response object
239         """ if
240         request_headers:
241
242

```

```
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
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(
                data=request.data))

```

```
273 _logger.debug('HEADERS: {headers}'.format( 274 headers=request.headers)) 275
```



```

276     response = Response(
277         self._make_request(opener, request, timeout=timeout)
278     )
279
280     _logger.debug('{method} Response: {status} {body}'.format(
281         method=method,
282         status=response.status_code,
283         body=response.body))
284
285     return response
286
287     return
288
http_request
else:
289     # Add a segment to the URL
290     return self._(name)
291
292     def __getstate__(self):
293         return self.__dict__
294
295     def __setstate__(self, state):

```

