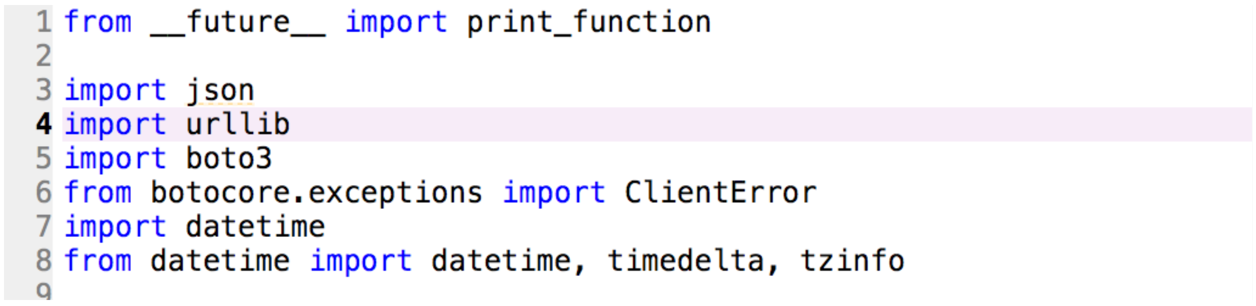
Implementation – Detailed Coding Steps

Step by step documentation of how the system is implemented in Python 2.7 is given below.

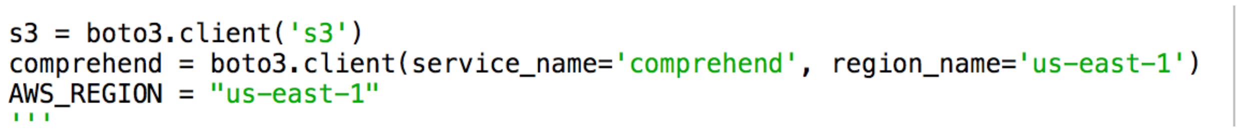
Significant portions of the python code from S3Lambda.py related to each step has been copied over below.

1.1. Step 1 – Importing the necessary packages



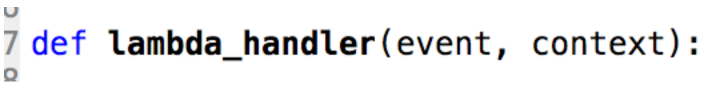
```python
from __future__ import print_function

import json
import urllib
import boto3
from botocore.exceptions import ClientError
import datetime
from datetime import datetime, timedelta, tzinfo
```

1.2. Step 2 – Defining AWS region for the application



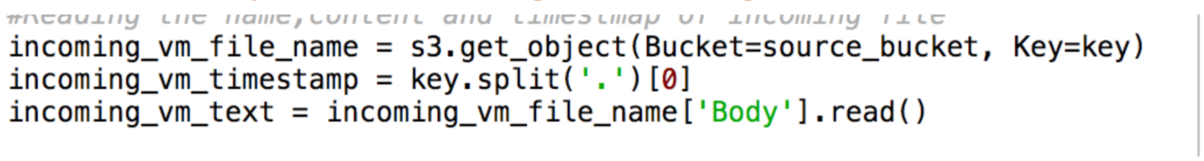
```python
s3 = boto3.client('s3')
comprehend = boto3.client(service_name='comprehend', region_name='us-east-1')
AWS_REGION = "us-east-1"
'''
```

1.3. Step 3 – Defining AWS Lambda function



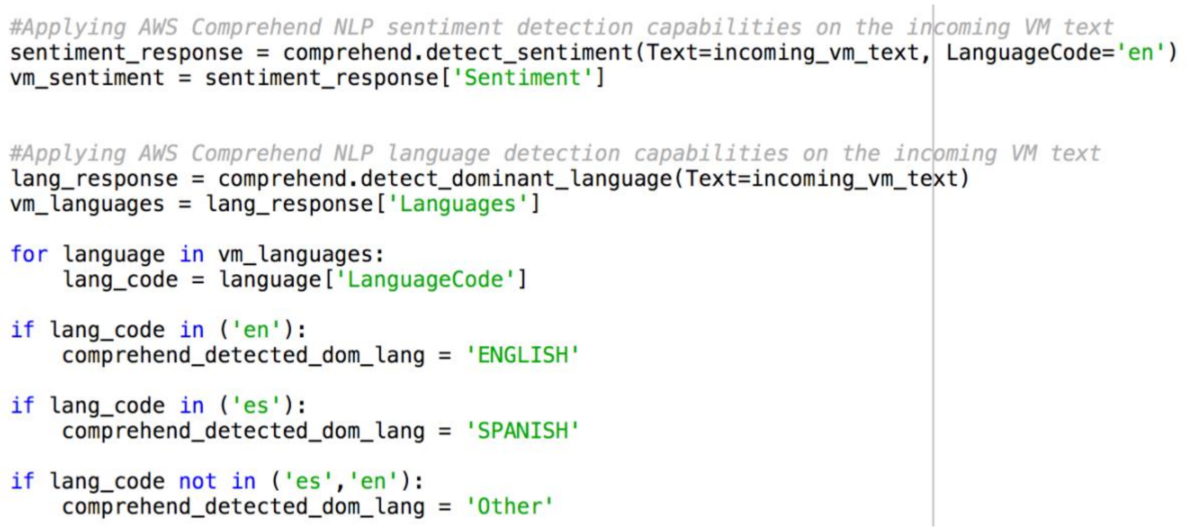
```python
def lambda_handler(event, context):
```

1.4. Step 4 – Fetching incoming file details



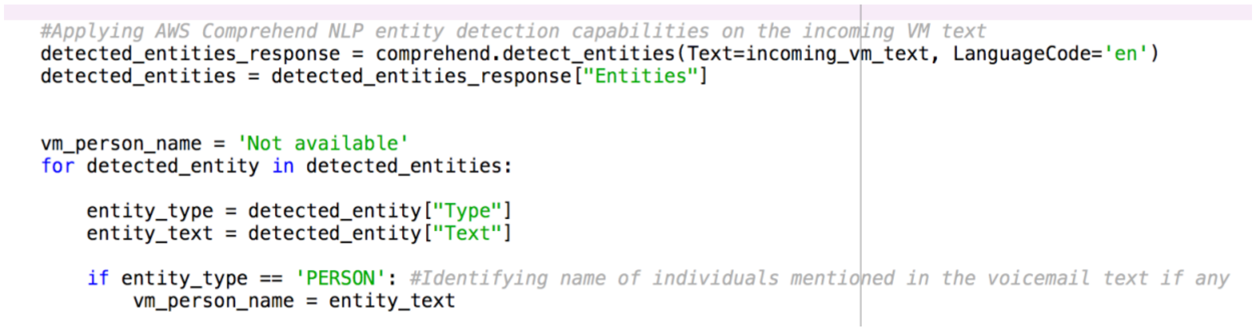
```python
#Reading the name,content and timestmap of incoming file
incoming_vm_file_name = s3.get_object(Bucket=source_bucket, Key=key)
incoming_vm_timestamp = key.split('.')[0]
incoming_vm_text = incoming_vm_file_name['Body'].read()
```

1.5. Step 5 – API call to AWS Comprehend for language detection



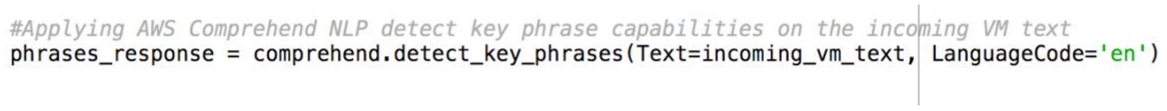
```
#Applying AWS Comprehend NLP sentiment detection capabilities on the incoming VM text
sentiment_response = comprehend.detect_sentiment(Text=incoming_vm_text, LanguageCode='en')
vm_sentiment = sentiment_response['Sentiment']


#Applying AWS Comprehend NLP language detection capabilities on the incoming VM text
lang_response = comprehend.detect_dominant_language(Text=incoming_vm_text)
vm_languages = lang_response['Languages']

for language in vm_languages:
    lang_code = language['LanguageCode']

if lang_code in ('en'):
    comprehend_detected_dom_lang = 'ENGLISH'

if lang_code in ('es'):
    comprehend_detected_dom_lang = 'SPANISH'

if lang_code not in ('es','en'):
    comprehend_detected_dom_lang = 'Other'
```

1.6. Step 6 – API call to AWS Comprehend for entity detection



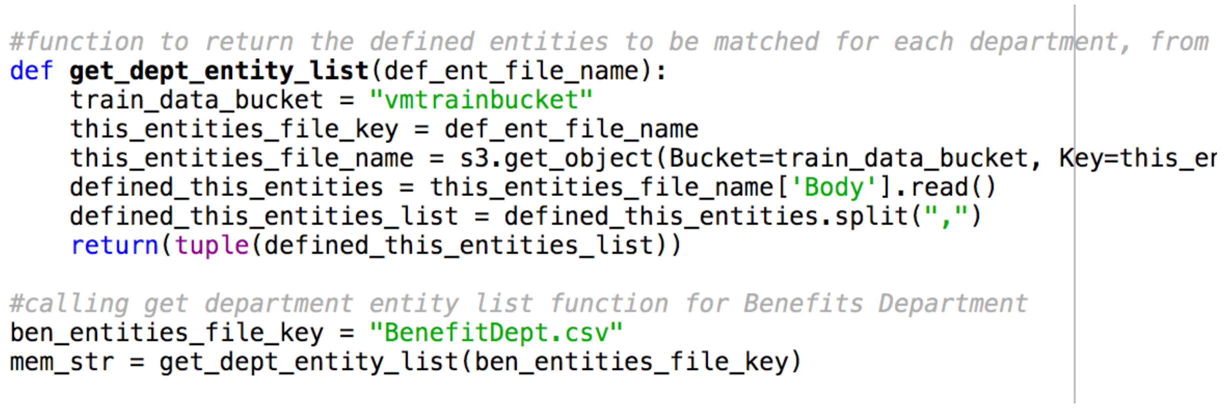
```
#Applying AWS Comprehend NLP entity detection capabilities on the incoming VM text
detected_entities_response = comprehend.detect_entities(Text=incoming_vm_text, LanguageCode='en')
detected_entities = detected_entities_response["Entities"]


vm_person_name = 'Not available'
for detected_entity in detected_entities:

    entity_type = detected_entity["Type"]
    entity_text = detected_entity["Text"]

    if entity_type == 'PERSON': #Identifying name of individuals mentioned in the voicemail text if any
        vm_person_name = entity_text
```

1.7. Step 7 – API call to AWS Comprehend for key phrases detection

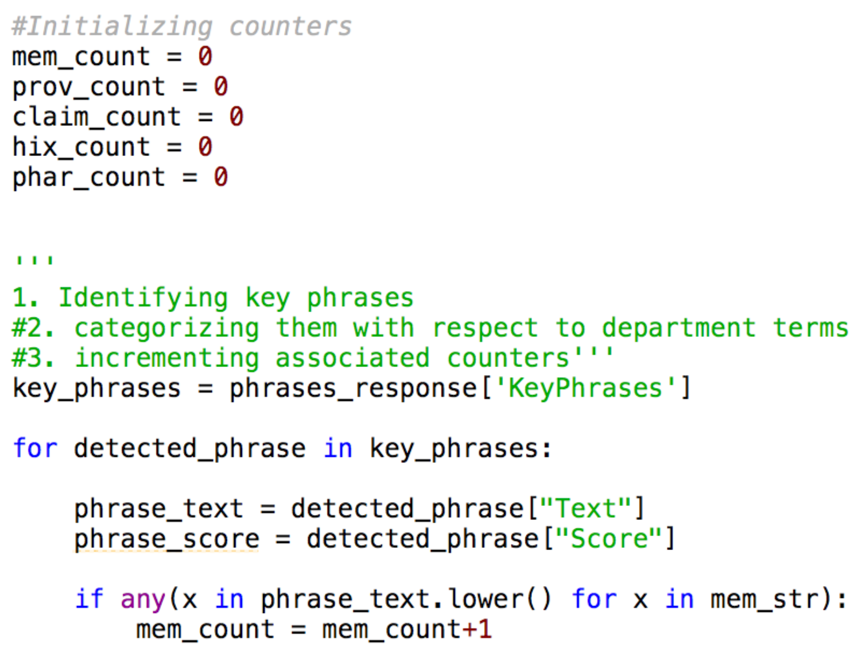


```
#Applying AWS Comprehend NLP detect key phrase capabilities on the incoming VM text
phrases_response = comprehend.detect_key_phrases(Text=incoming_vm_text, LanguageCode='en')
```

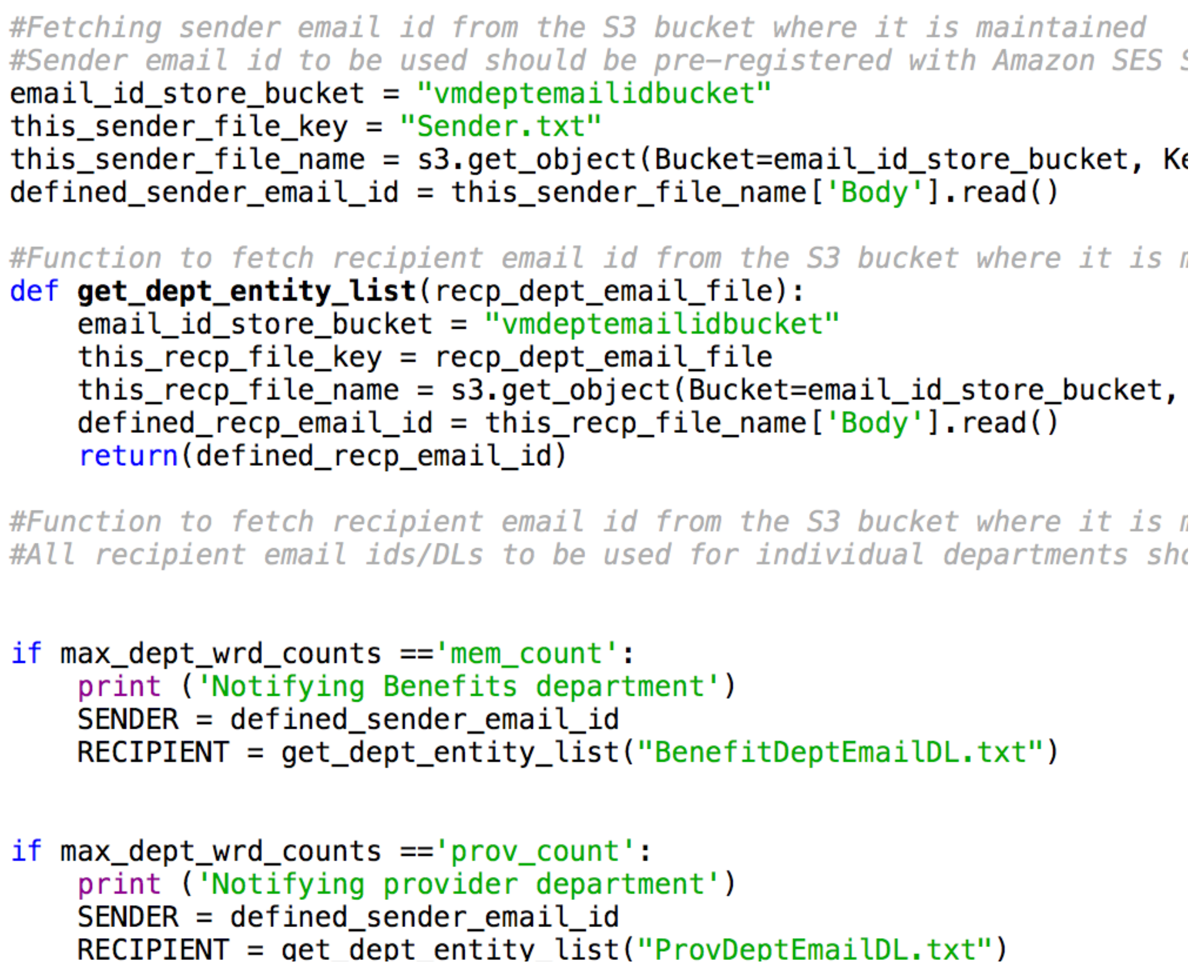
1.8. Step 8 – Identifying the department to be notified based on dept data from S3, Fetching sender and recipient email ids from S3



```
#function to return the defined entities to be matched for each department, from
def get_dept_entity_list(def_ent_file_name):
    train_data_bucket = "vmtrainbucket"
    this_entities_file_key = def_ent_file_name
    this_entities_file_name = s3.get_object(Bucket=train_data_bucket, Key=this_er
    defined_this_entities = this_entities_file_name['Body'].read()
    defined_this_entities_list = defined_this_entities.split(",")
    return(tuple(defined_this_entities_list))

#calling get department entity list function for Benefits Department
ben_entities_file_key = "BenefitDept.csv"
mem_str = get_dept_entity_list(ben_entities_file_key)
```

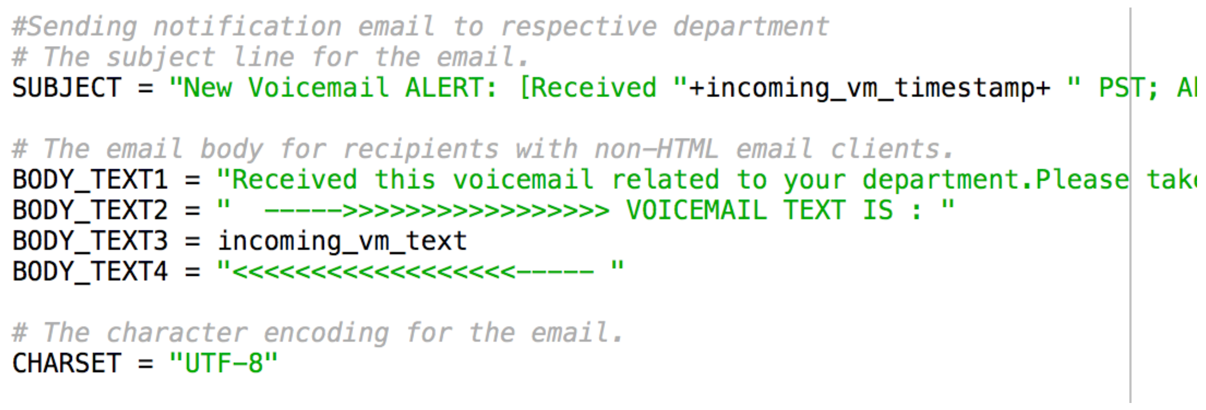


```
#Initializing counters
mem_count = 0
prov_count = 0
claim_count = 0
hix_count = 0
phar_count = 0


'''
1. Identifying key phrases
#2. categorizing them with respect to department terms
#3. incrementing associated counters'''
key_phrases = phrases_response['KeyPhrases']

for detected_phrase in key_phrases:

    phrase_text = detected_phrase["Text"]
    phrase_score = detected_phrase["Score"]

    if any(x in phrase_text.lower() for x in mem_str):
        mem_count = mem_count+1
```



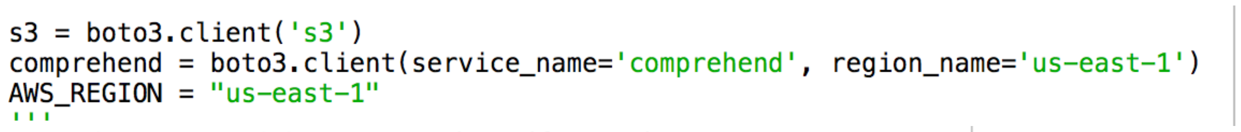
```
#Fetching sender email id from the S3 bucket where it is maintained
#Sender email id to be used should be pre-registered with Amazon SES S
email_id_store_bucket = "vmdeptemailidbucket"
this_sender_file_key = "Sender.txt"
this_sender_file_name = s3.get_object(Bucket=email_id_store_bucket, Ke
defined_sender_email_id = this_sender_file_name['Body'].read()

#Function to fetch recipient email id from the S3 bucket where it is m
def get_dept_entity_list(recp_dept_email_file):
    email_id_store_bucket = "vmdeptemailidbucket"
    this_recp_file_key = recp_dept_email_file
    this_recp_file_name = s3.get_object(Bucket=email_id_store_bucket,
    defined_recp_email_id = this_recp_file_name['Body'].read()
    return(defined_recp_email_id)

#Function to fetch recipient email id from the S3 bucket where it is m
#All recipient email ids/DLs to be used for individual departments sho


if max_dept_wrd_counts =='mem_count':
    print ('Notifying Benefits department')
    SENDER = defined_sender_email_id
    RECIPIENT = get_dept_entity_list("BenefitDeptEmailDL.txt")


if max_dept_wrd_counts =='prov_count':
    print ('Notifying provider department')
    SENDER = defined_sender_email_id
    RECIPIENT = get_dept_entity_list("ProvDeptEmailDL.txt")
```

1.9. Step 9 – Formulating email notification content

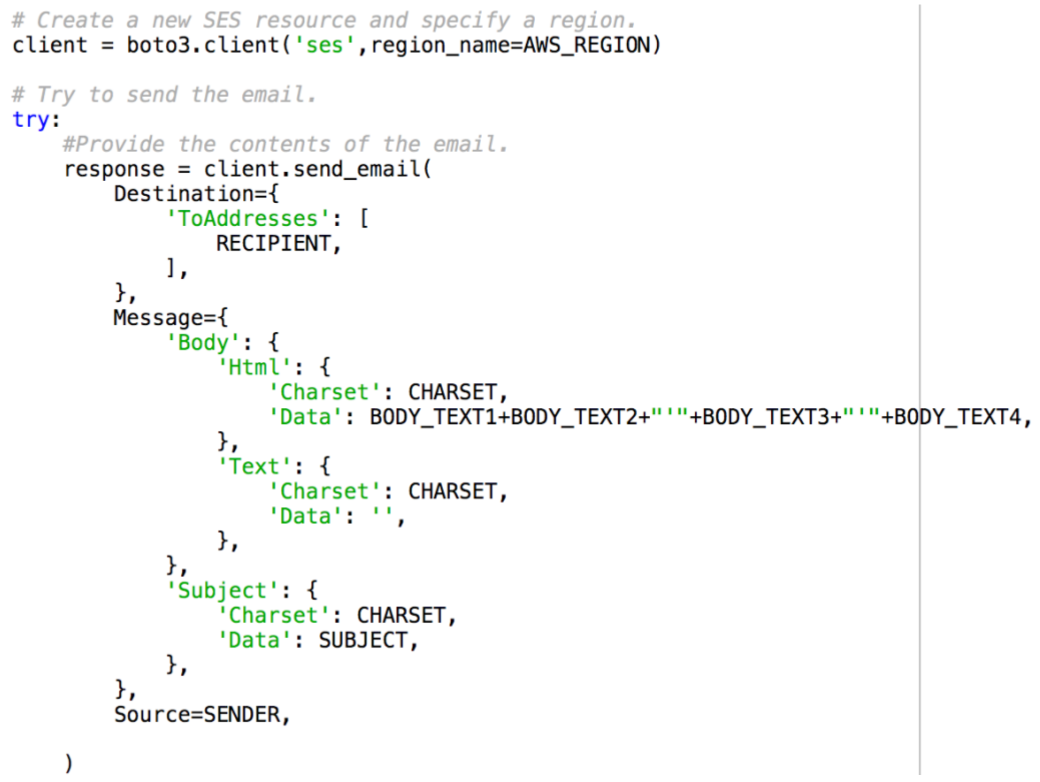


```
#Sending notification email to respective department
# The subject line for the email.
SUBJECT = "New Voicemail ALERT: [Received "+incoming_vm_timestamp+ " PST; Al

# The email body for recipients with non-HTML email clients.
BODY_TEXT1 = "Received this voicemail related to your department.Please take
BODY_TEXT2 = "  ----->>>>>>>>>>>>>>>>>> VOICEMAIL TEXT IS : "
BODY_TEXT3 = incoming_vm_text
BODY_TEXT4 = "<<<<<<<<<<<<<<<<<<------ "

# The character encoding for the email.
CHARSET = "UTF-8"
```

1.10. Step 10 – Calling AWS SES for sending email notification

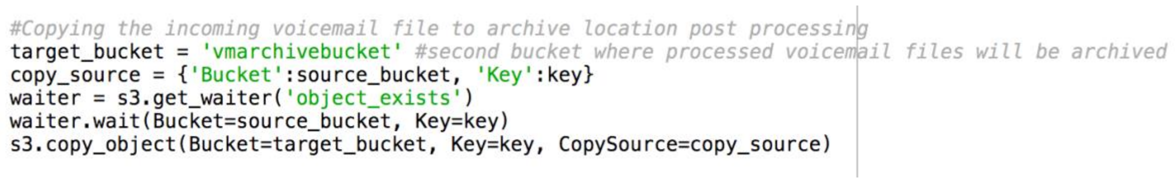


```
s3 = boto3.client('s3')
comprehend = boto3.client(service_name='comprehend', region_name='us-east-1')
AWS_REGION = "us-east-1"
'''
```



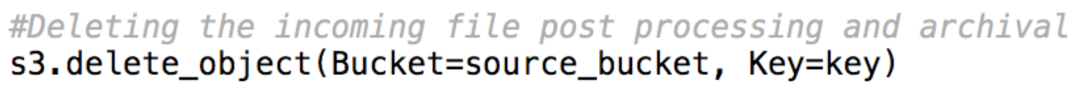
```
    # Create a new SES resource and specify a region.
    client = boto3.client('ses',region_name=AWS_REGION)

    # Try to send the email.
    try:
        #Provide the contents of the email.
        response = client.send_email(
            Destination={
                'ToAddresses': [
                    RECIPIENT,
                ],
            },
            Message={
                'Body': {
                    'Html': {
                        'Charset': CHARSET,
                        'Data': BODY_TEXT1+BODY_TEXT2+"'"+BODY_TEXT3+"'"+BODY_TEXT4,
                    },
                    'Text': {
                        'Charset': CHARSET,
                        'Data': '',
                    },
                },
                'Subject': {
                    'Charset': CHARSET,
                    'Data': SUBJECT,
                },
            },
            Source=SENDER,

        )
```

1.11. Step 11 – Copying incoming file to archive S3 bucket post processing



```
#Copying the incoming voicemail file to archive location post processing
target_bucket = 'vmarchivebucket' #second bucket where processed voicemail files will be archived
copy_source = {'Bucket':source_bucket, 'Key':key}
waiter = s3.get_waiter('object_exists')
waiter.wait(Bucket=source_bucket, Key=key)
s3.copy_object(Bucket=target_bucket, Key=key, CopySource=copy_source)
```

1.12. Step 12 – Deleting the incoming file from inbound S3 bucket post processing



```
#Deleting the incoming file post processing and archival
s3.delete_object(Bucket=source_bucket, Key=key)
```