In [1]:

```
isinstance(3,int)
```

Out[1]:

```
True
```

In [5]:

```
l1 = ['AA','aa','BB','bb']
l1.sort()
print(l1)
l1.sort(key = str.lower,reverse = True)
print(l1)
```

```
['AA', 'BB', 'aa', 'bb']
['BB', 'bb', 'AA', 'aa']
```

smtplib — SMTP protocol client¶

In [ ]:

```
The smtplib module defines an SMTP client session object that can be used to send mail
to any Internet machine with an SMTP or ESMTP listener daemon. For details of SMTP and
ESMTP operation, consult RFC 821 (Simple Mail Transfer Protocol) and RFC 1869 (SMTP Ser
vice Extensions).

For normal use, you should only require the initialization/connect, sendmail(), and SMT
P.quit() methods.

class smtplib.SMTP(host='', port=0, local_hostname=None, [timeout, ]source_address=None
)¶

The SMTP class supports the with statement. When used like this, the SMTP QUIT command
is issued automatically when the with statement.

from smtplib import SMTP
with SMTP("domain.org") as smtp:
...     smtp.noop()
```

Exceptions

In [ ]:

```
exception smtplib.SMTPException
Subclass of OSError that is the base exception class for all the other exceptions provi
ded by this module.

Changed in version 3.4: SMTPException became subclass of OSError

exception smtplib.SMTPServerDisconnected
This exception is raised when the server unexpectedly disconnects, or when an attempt i
s made to use the SMTP instance before connecting it to a server.

exception smtplib.SMTPResponseException
Base class for all exceptions that include an SMTP error code. These exceptions are gen
erated in some instances when the SMTP server returns an error code. The error code is
stored in the smtp_code attribute of the error, and the smtp_error attribute is set to
the error message.

exception smtplib.SMTPSenderRefused
Sender address refused. In addition to the attributes set by on all SMTPResponseExcepti
on exceptions, this sets 'sender' to the string that the SMTP server refused.

exception smtplib.SMTPRecipientsRefused
All recipient addresses refused. The errors for each recipient are accessible through t
he attribute recipients, which is a dictionary of exactly the same sort as SMTP.sendmai
l() returns.

exception smtplib.SMTPDataError
The SMTP server refused to accept the message data.

exception smtplib.SMTPConnectError
Error occurred during establishment of a connection with the server.

exception smtplib.SMTPHeloError
The server refused our HELO message.

exception smtplib.SMTPNotSupportedError
The command or option attempted is not supported by the server.

New in version 3.5.

exception smtplib.SMTPAuthenticationError
SMTP authentication went wrong. Most probably the server didn't accept the username/pas
sword combination provided.
```

SMTP Objects

In [ ]:

```
SMTP.set_debuglevel(level)
Set the debug output level. A true value for level results in debug messages for connec
tion and for all messages sent to and received from the server.

SMTP.docmd(cmd[, argstring])
Send a command cmd to the server. The optional argument argstring is simply concatenate
d to the command, separated by a space.

This returns a 2-tuple composed of a numeric response code and the actual response line
(multiline responses are joined into one long line.)

In normal operation it should not be necessary to call this method explicitly. It is us
ed to implement other methods and may be useful for testing private extensions.

If the connection to the server is lost while waiting for the reply, SMTPServerDisconne
cted will be raised.

SMTP.connect([host[, port]])
Connect to a host on a given port. The defaults are to connect to the local host at the
standard SMTP port (25). If the hostname ends with a colon (':') followed by a number,
that suffix will be stripped off and the number interpreted as the port number to use.
This method is automatically invoked by the constructor if a host is specified during i
nstantiation. Returns a 2-tuple of the response code and message sent by the server in
its connection response.

SMTP.helo([hostname])
Identify yourself to the SMTP server using HELO. The hostname argument defaults to the
fully qualified domain name of the local host. The message returned by the server is st
ored as the helo_resp attribute of the object.

In normal operation it should not be necessary to call this method explicitly. It will
be implicitly called by the sendmail() when necessary.

SMTP.ehlo([hostname])
Identify yourself to an ESMTP server using EHLO. The hostname argument defaults to the
fully qualified domain name of the local host. Examine the response for ESMTP option an
d store them for use by has_extn(). Also sets several informational attributes: the mes
sage returned by the server is stored as the ehlo_resp attribute, does_esmtp is set to
true or false depending on whether the server supports ESMTP, and esmtp_features will b
e a dictionary containing the names of the SMTP service extensions this server supports
, and their parameters (if any).

Unless you wish to use has_extn() before sending mail, it should not be necessary to ca
ll this method explicitly. It will be implicitly called by sendmail() when necessary.

SMTP.ehlo_or_helo_if_needed()
This method call ehlo() and or helo() if there has been no previous EHLO or HELO comman
d this session. It tries ESMTP EHLO first.

SMTPHeloError
The server didn't reply properly to the HELO greeting.
New in version 2.6.

SMTP.has_extn(name)
Return True if name is in the set of SMTP service extensions returned by the server, Fa
lse otherwise. Case is ignored.

SMTP.verify(address)
Check the validity of an address on this server using SMTP VRFY. Returns a tuple consis
```

```
ting of code 250 and a full RFC 822 address (including human name) if the user address
is valid. Otherwise returns an SMTP error code of 400 or greater and an error string.

SMTP.quit()
SMTP.send_message(msg, from_addr=None, to_addrs=None, mail_options=(), rcpt_options=())
SMTP.sendmail(from_addr, to_addrs, msg, mail_options=(), rcpt_options=())
```

Code

Send plain text

In [ ]:

```
from smtplib import SMTP,SMTPAuthenticationError

host = 'smtp.gmail.com'
port = 587
username = 'pytesting1207@gmail.com'
password = 'Python@123'
from_email = 'pytesting1207@gmail.com'
to_mail = ['abhishekebay02@gmail.com']
msg = 'This is a testing mail'
try:
    email_conn = SMTP(host,port)
    print(email_conn.ehlo())
    email_conn.starttls()
    print(email_conn.login(username,password))
    email_conn.sendmail(from_email,to_mail,msg)

except SMTPAuthenticationError as s:
    print(s)
except Exception as e:
    print(e)

finally:
    email_conn.quit()
```

Send html

In [ ]:

```
from smtplib import SMTP,SMTPAuthenticationError,SMTPException
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

host = 'smtp.gmail.com'
port = 587
username = 'pytesting1207@gmail.com'
password = 'Python@123'
from_email = 'pytesting1207@gmail.com'
to_mail = ['abhishekebay02@gmail.com']
msg = 'This is a testing mail'
try:
    email_conn = SMTP(host,port)
    print(email_conn.ehlo())
    email_conn.starttls()
    print(email_conn.login(username,password))
    the_msg = MIMEMultipart('alternative')
    the_msg['Subject'] = 'Testing'
    the_msg['From'] = from_email
    #the_msg['To'] = to_mail
    plain_text = 'This is a testing message'
    html_text = '''
    <html>
    <head></head>
    <body>
    <p>Hey!<br/>This is a testing message.Made by me</p>
    </body>
    </html>
    '''
    part1 = MIMEText(plain_text,'plain')
    part2 = MIMEText(html_text,'html')
    the_msg.attach(part1)
    the_msg.attach(part2)
    email_conn.sendmail(from_email,to_mail,the_msg.as_string())

except SMTPAuthenticationError as s:
    print(s)
except SMTPException as a:
    print(a)
except Exception as e:
    print(e)

finally:
    email_conn.quit()
```

shutil

The shutil module offers a number of high-level operations on files and collections of files. In particular, functions are provided which support file copying and removal. For operations on individual files, see also the os module.

Directory and files operations

In [ ]:

```
shutil.copyfileobj(fsrc, fdst[, length])
Copy the contents of the file-like object fsrc to the file-like object fdst. The intege
r length, if given, is the buffer size.

shutil.copyfile(src, dst, *, follow_symlinks=True)
Copy the contents (no metadata) of the file named src to a file named dst and return ds
t. src and dst are path names given as strings. dst must be the complete target file na
me;
look at shutil.copy() for a copy that accepts a target directory path. If src and dst s
pecify the same file, SameFileError is raised.

exception shutil.SameFileError
This exception is raised if source and destination in copyfile() are the same file.

shutil.copymode(src, dst, *, follow_symlinks=True)
Copy the permission bits from src to dst. The file contents, owner, and group are unaff
ected. src and dst are path names given as strings.

shutil.copystat(src, dst, *, follow_symlinks=True)
Copy the permission bits, last access time, last modification time, and flags from src
to dst. On Linux, copystat() also copies the “extended attributes” where possible.
The file contents, owner, and group are unaffected. src and dst are path names given as
strings.

shutil.copy(src, dst, *, follow_symlinks=True)
Copies the file src to the file or directory dst. src and dst should be strings.
If dst specifies a directory, the file will be copied into dst using the base filename
from src. Returns the path to the newly created file.

shutil.copy2(src, dst, *, follow_symlinks=True)
Identical to copy() except that copy2() also attempts to preserve file metadata.

shutil.ignore_patterns(*patterns)
This factory function creates a function that can be used as a callable for copytree()'
s ignore argument, ignoring files and directories that match one of the glob-style patt
erns provided.

shutil.copytree(src, dst, symlinks=False, ignore=None, copy_function=copy2, ignore_dang
ling_symlinks=False)
Recursively copy an entire directory tree rooted at src, returning the destination dire
ctory. The destination directory,
named by dst, must not already exist; it will be created as well as missing parent dire
ctories.
Permissions and times of directories are copied with copystat(), individual files are c
opied using shutil.copy2().

shutil.rmtree(path, ignore_errors=False, onerror=None)
Delete an entire directory tree; path must point to a directory (but not a symbolic lin
k to a directory).

shutil.move(src, dst, copy_function=copy2)
Recursively move a file or directory (src) to another location (dst) and return the des
tination.

shutil.disk_usage(path)
Return disk usage statistics about the given path as a named tuple with the attributes
total, used and free, which are the amount of total, used and free space, in bytes.
```

```
shutil.chown(path, user=None, group=None)
Change owner user and/or group of the given path.

shutil.which(cmd, mode=os.F_OK | os.X_OK, path=None)
Return the path to an executable which would be run if the given cmd was called. If no
cmd would be called, return None.

exception shutil.Error
This exception collects exceptions that are raised during a multi-file operation. For c
opytree(), the exception argument is a list of 3-tuples (srcname, dstname, exception).
```

Archiving operations

In [ ]:

```
shutil.make_archive(base_name, format[, root_dir[, base_dir[, verbose[, dry_run[, owner
[, group[, logger]]]]]]])
Create an archive file (such as zip or tar) and return its name.
```

tempfile — Generate temporary files and directories

In [ ]:

```
This module generates temporary files and directories. It works on all supported platfo
rms.t now provides three new functions, NamedTemporaryFile(), mkstemp(), and mkdtemp(),
which should eliminate all remaining need to use the insecure mktemp() function. Tempor
ary file names created by this module no longer contain the process ID;
instead a string of six random characters is used.
```

In [ ]:

```
tempfile.TemporaryFile([mode='w+b'[, bufsize=-1[, suffix=''[, prefix='tmp'[, dir=None
]]]]])
Return a file-like object that can be used as a temporary storage area. The file is cre
ated using mkstemp().
It will be destroyed as soon as it is closed (including an implicit close when the obje
ct is garbage collected).
Under Unix, the directory entry for the file is removed immediately after the file is c
reated.
Other platforms do not support this; your code should not rely on a temporary file crea
ted using this function having or not having a visible name in the file system.
The mode parameter defaults to 'w+b' so that the file created can be read and written w
ithout being closed.
Binary mode is used so that it behaves consistently on all platforms without regard for
the data that is stored. bufsize defaults to -1, meaning that the operating system defa
ult is used.

tempfile.NamedTemporaryFile([mode='w+b'[, bufsize=-1[, suffix=''[, prefix='tmp'[, dir=N
one[, delete=True]]]]]])
This function operates exactly as TemporaryFile() does, except that the file is guarant
eed to have a visible name in the file system (on Unix, the directory entry is not unli
nked).
That name can be retrieved from the name attribute of the returned file-like object.
Whether the name can be used to open the file a second time, while the named temporary
file is still open, varies across platforms (it can be so used on Unix; it cannot on Wi
ndows NT or later).
If delete is true (the default), the file is deleted as soon as it is closed.
The returned object is always a file-like object whose file attribute is the underlying
true file object. This file-like object can be used in a with statement, just like a no
rmal file.

tempfile.mkstemp([suffix=''[, prefix='tmp'[, dir=None[, text=False]]]])
Creates a temporary file in the most secure manner possible. There are no race conditio
ns in the file's creation, assuming that the platform properly implements the os.O_EXCL
flag for os.open().
The file is readable and writable only by the creating user ID. If the platform uses pe
rmission bits to indicate whether a file is executable, the file is executable by no on
e.
The file descriptor is not inherited by child processes.
Unlike TemporaryFile(), the user of mkstemp() is responsible for deleting the temporary
file when done with it.

tempfile.mkdtemp([suffix=''[, prefix='tmp'[, dir=None]]])
Creates a temporary directory in the most secure manner possible. There are no race con
ditions in the directory's creation.
The directory is readable, writable, and searchable only by the creating user ID.
The user of mkdtemp() is responsible for deleting the temporary directory and its conte
nts when done with it.
The prefix, suffix, and dir arguments are the same as for mkstemp().
mkdtemp() returns the absolute pathname of the new directory.
```

argparse — Parser for command-line options, arguments and sub-commands

In [ ]:

```
The argparse module makes it easy to write user-friendly command-line interfaces. The p
rogram defines what arguments it requires, and argparse will figure out how to parse th
ose out of sys.argv.
The argparse module also automatically generates help and usage messages and issues err
ors when users give the program invalid arguments.
create a __init__.py and __main__.py files in folder.open __main__ file and write below
code
import argparse
parser = argparse.ArgumentParser(prog='folder_name')
parser.add_argument('-id','--user_id', type=int, nargs='+',
                    help='an integer for the accumulator')
args = parser.parse_args()
print(args.accumulate(id = args.id))
```

Requests: HTTP for Humans

In [ ]:

```
Requests is the only Non-GMO HTTP library for Python, safe for human consumption.
import requests
>>> r = requests.get('https://api.github.com/events')
>>> r = requests.post('https://httpbin.org/post', data = {'key':'value'})
>>> r = requests.put('https://httpbin.org/put', data = {'key':'value'})
>>> r = requests.delete('https://httpbin.org/delete')
>>> r = requests.head('https://httpbin.org/get')
>>> r = requests.options('https://httpbin.org/get')
```

Passing Parameters In URLs

In [ ]:

```
>>> payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.get('https://httpbin.org/get', params=payload)
>>> print(r.url)
https://httpbin.org/get?key2=value2&key1=value1
>>> payload = {'key1': 'value1', 'key2': ['value2', 'value3']}

>>> r = requests.get('https://httpbin.org/get', params=payload)
>>> print(r.url)
https://httpbin.org/get?key1=value1&key2=value2&key2=value3
```

Response Content

In [ ]:

```
>>> import requests

>>> r = requests.get('https://api.github.com/events')
>>> r.text
u'[{"repository":{"open_issues":0,"url":"https://github.com/...
Requests will automatically decode content from the server. Most unicode charsets are s
eamlessly decoded.
When you make a request, Requests makes educated guesses about the encoding of the resp
onse based on the HTTP headers.
The text encoding guessed by Requests is used when you access r.text. You can find out
what encoding Requests is using, and change it, using the r.encoding property:
>>> r.encoding
'utf-8'
>>> r.encoding = 'ISO-8859-1'
```

Web scrapping

In [ ]:

```
#with fix url
import requests
from bs4 import BeautifulSoup
url = 'https://www.yelp.com/search?cflt=restaurants&find_loc=San+Francisco%2C+CA'
yelp_r = requests.get(url)
print(yelp_r)
print(yelp_r.status_code)
soup = BeautifulSoup(yelp_r.text,'html.parser')
print(soup.prettify())
```

In [55]:

```
#with dynamic url
import requests
from bs4 import BeautifulSoup
file = 'scrapping_data.txt'
base_url = 'https://www.yelp.com/search?cflt=restaurants&find_loc='
page = 0
while page < 200:
    print(page)
    loc = 'New York,NY'
    url = base_url + loc + '&start=' + str(page)
    yelp_r = requests.get(url)
    #print(yelp_r.status_code)
    soup = BeautifulSoup(yelp_r.text,'html.parser')
    #print(soup.prettify())
    bel = soup.findAll('div',{'class':'lemon--div__373c0__1mboc businessName__373c0__1f
Tgn border-color--default__373c0__2oFDT'})
    addphone = soup.findAll('div',{'class':'lemon--div__373c0__1mboc container__373c0__
19wDx u-padding-l2 border-color--default__373c0__2oFDT text-align--right__373c0__3fmmn'
})
    for ad,b in zip(addphone,bel):
        with open(file,'a') as f:
            try:
                title = b.findAll('a')[0].text
                address = ad.findAll('address')[0].text
                phone = ad.findAll('div',{'class':'lemon--div__373c0__1mboc display--in
line-block__373c0__2de_K u-space-b1 border-color--default__373c0__2oFDT'})[0].text
                page_line = f'{title} has address {address} and phone number {phone}'
                f.write(page_line)
            except:
                pass
    page = page + 30
```

```
0
30
60
90
120
150
180
```

Javascript Scrapping

In [ ]:

```
import os
import requests
from selenium import webdriver
from bs4 import BeautifulSoup
filepath = os.path.dirname(__file__) + '/chromedriver_win32/chromedriver.exe'
print(filepath)
url = 'https://www.google.com/search?q=avengers&safe=active&source=lnms&tbm=isch&sa=X&v
ed=0ahUKEwilgYXLgN_hAhWf6nMBHTKYB8EQ_AUIDygC&biw=1920&bih=975'
web_r = requests.get(url)
print(web_r.status_code)
soup = BeautifulSoup(web_r.text,'html.parser')
print(len(soup.findAll('img')))

driver = webdriver.Chrome(filepath)
driver.get(url)
html = driver.execute_script("return document.documentElement.outerHTML")
sel_soup = BeautifulSoup(html,'html.parser')
print(len(sel_soup.findAll('img')))

images = []
for i in sel_soup.findAll('img'):
    print(i)
    src = i['src']
    images.append(src)
print(images)
```

In [ ]:

```
import os
import shutil
import time
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
url = 'https://www.google.com/search?q=avengers&safe=active&source=lnms&tbm=isch&sa=X&v
ed=0ahUKEwilgYXLgN_hAhWf6nMBHTKYB8EQ_AUIDygC&biw=1920&bih=975'
filepath = os.path.dirname(__file__) + '/chromedriver_win32/chromedriver.exe'
print(filepath)
web_r = requests.get(url)
web_soup = BeautifulSoup(web_r.text, 'html.parser')
print(web_soup.findAll("img"))
#<img src=''/>
driver = webdriver.Chrome(filepath)
driver.get(url)
iterations = 0
while iterations < 10:
    html = driver.execute_script("return document.documentElement.outerHTML")
    sel_soup = BeautifulSoup(html, 'html.parser')
    print(len(sel_soup.findAll("img")))
    images = []
    for i in sel_soup.findAll("img"):
        src = i["src"]
        images.append(src)
    print(images)
    current_path = os.getcwd()
    for img in images:
        try:
            file_name = os.path.basename(img)
            img_r = requests.get(img, stream=True)
            new_path = os.path.join(current_path, "images", file_name)
            with open(new_path, "wb") as output_file:
                shutil.copyfileobj(img_r.raw, output_file)
            del img_r
        except:
            pass
    iterations += 1
    time.sleep(5)
```

ImageScraper 2.0.7

install: pip install ImageScraper Format: image-scraper [OPTIONS] URL Scrape all images: image-scraper ananth.co.in/test.html Scrape at max 2 images: image-scraper -m 2 ananth.co.in/test.html Scrape only gifs and download to folder ./mygifs: image-scraper -s mygifs ananth.co.in/test.html --formats gif

Get Data Using API for yelp

In [ ]:

```
import requests
import json
api_key = 'QckQ4YpbZ819kvujGxxBHwU3Uc4_z99l9JoP8PJY287cboAvrMsptlsiEaN0jhRsOf71xF2Bgq1p
HlOJWboOZJqEfA5TNycA9xMNkTrCXZ1hBYgn4NOdwvlzRlO8XHYx'
headers = {'Authorization':'Bearer %s' %api_key}

url = 'https://api.yelp.com/v3/businesses/search'
param = {'term':'seafood','location':'New York City'}

req = requests.get(url,params = param,headers= headers)
print(req.status_code)
response = req.json()
for i in response['businesses']:
    print(i["name"])
    print(i["phone"])
    print(i["location"]["display_address"])
    print(i["location"]["city"])
```

Send Messages using Twilio

using programmable sms api

In [ ]:

```
url = https://www.twilio.com/
```

In [ ]:

```
from twilio.rest import Client
# Your Account Sid and Auth Token from twilio.com/console
# DANGER! This is insecure. See http://twil.io/secure
account_sid = 'ACf888ccf98d4bab54aeb599a9bfe076fb'
auth_token = 'from_account'
client = Client(account_sid, auth_token)

message = client.messages.create(
                              from_='+12019480914',
                              body='this is a test message',
                              to='+919455112807'
                          )
print(message.sid)
message_data = client.messages.get(sid = '#after creating sms')
print(message_data)
print(dir(message_data))
```

Use an image url

In [ ]:

```
# Download the helper library from https://www.twilio.com/docs/python/install
from twilio.rest import Client


# Your Account Sid and Auth Token from twilio.com/console
# DANGER! This is insecure. See http://twil.io/secure
account_sid = 'ACXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
auth_token = 'your_auth_token'
client = Client(account_sid, auth_token)

message = client.messages \
    .create(
         body='This is the ship that made the Kessel Run in fourteen parsecs?',
         from_='+15017122661',
         media_url='https://c1.staticflickr.com/3/2899/14341091933_1e92e62d12_b.jpg',
         to='+15558675310'
     )

print(message.sid)
```

Twitter Api and python

In [ ]:

```
developer_url :https://developer.twitter.com/
app_url : https://developer.twitter.com/en/apps
```

In [ ]:

```
'''
Install: pip install python-twitter
Project Github: https://github.com/codingforentrepreneurs/30-Days-of-Python
Github: https://github.com/bear/python-twitter
Docs: https://python-twitter.readthedocs.io
'''
import twitter

consumer_key = 'G2vW5RTt5y8gPutlPvWgIc54d'
consumer_secret = 'CbsWYdcygM0q7zXzk9mWbE2OvMNQSKBJhikOt6Z8xAxm3c1NdC'
access_token = '787057602484051968-mdYWmpLaVZj8emlltHs0eIXVjRuzULQ'
access_secret = 'd7OgtcX2wgF7ttHcKqtDFDgZB9b8jYjDcnzb9awyQnY8T'

api = twitter.Api(consumer_key=consumer_key,
                  consumer_secret=consumer_secret,
                  access_token_key=access_token,
                  access_token_secret=access_secret)

print(api.VerifyCredentials())

follwers = api.GetFollowers()
friends = api.GetFriends()

status_var = '@justinmitchel #Python is amazing! #30daysofpython http://joincfe.com/pro
jects'
post_update = api.PostUpdates(status=status_var)

length_status = twitter.twitter_utils.calc_expected_status_length(status=status_var)

new_messsage = api.PostDirectMessage(screen_name='justinmitchel', text='Hi there')
print(new_messsage)

new_magic_message = api.PostDirectMessage(screen_name='MagicJohnson', text='Hey Magic!
 Big fan.')
print(new_magic_message)

api.GetUser(user)
api.GetReplies()
api.GetUserTimeline(user)
api.GetHomeTimeline()
api.GetStatus(status_id=787079994451202048) #status_id = 787079994451202048
api.DestroyStatus(status_id)
api.GetFriends(user)
api.GetFollowers()
api.GetFeatured()
api.GetDirectMessages()
api.GetSentDirectMessages()
api.PostDirectMessage(user, text)
api.DestroyDirectMessage(message_id)
api.DestroyFriendship(user)
api.CreateFriendship(user)
api.LookupFriendship(user)
api.VerifyCredentials()
```

Working me Gmail

In [6]:

```
import email
import imaplib

username = 'pytesting1207@gmail.com'
password = 'Python@123'

mail = imaplib.IMAP4_SSL('imap.gmail.com')
print(mail.login(username,password))
print(mail.select('inbox'))
print(mail.list())
```

```
('OK', [b'pytesting1207@gmail.com authenticated (Success)'])
('OK', [b'12'])
('OK', [b'(\\HasNoChildren) "/" "INBOX"', b'(\\HasChildren \\Noselect) "/"
"[Gmail]"', b'(\\All \\HasNoChildren) "/" "[Gmail]/All Mail"', b'(\\Drafts
\\HasNoChildren) "/" "[Gmail]/Drafts"', b'(\\HasNoChildren \\Important)
"/" "[Gmail]/Important"', b'(\\HasNoChildren \\Sent) "/" "[Gmail]/Sent Mai
l"', b'(\\HasNoChildren \\Junk) "/" "[Gmail]/Spam"', b'(\\Flagged \\HasNoC
hildren) "/" "[Gmail]/Starred"', b'(\\HasNoChildren \\Trash) "/" "[Gmail]/
Trash"'])
```

In [7]:

```
mail.create('Important')
```

Out[7]:

```
('OK', [b'Success'])
```

In [8]:

```
mail.list()
```

Out[8]:

```
('OK',
 [b'(\\HasNoChildren) "/" "INBOX"',
  b'(\\HasNoChildren) "/" "Important"',
  b'(\\HasChildren \\Noselect) "/" "[Gmail]"',
  b'(\\All \\HasNoChildren) "/" "[Gmail]/All Mail"',
  b'(\\Drafts \\HasNoChildren) "/" "[Gmail]/Drafts"',
  b'(\\HasNoChildren \\Important) "/" "[Gmail]/Important"',
  b'(\\HasNoChildren \\Sent) "/" "[Gmail]/Sent Mail"',
  b'(\\HasNoChildren \\Junk) "/" "[Gmail]/Spam"',
  b'(\\Flagged \\HasNoChildren) "/" "[Gmail]/Starred"',
  b'(\\HasNoChildren \\Trash) "/" "[Gmail]/Trash"'])
```

In [10]:

```
result,data = mail.uid('search',None,'ALL')
data
```

Out[10]:

```
[b'2 3 4 5 6 7 8 9 10 11 12 13']
```

In [11]:

```
mail.uid('search',None,'ALL')
```

Out[11]:

```
('OK', [b'2 3 4 5 6 7 8 9 10 11 12 13'])
```

In [ ]:

```
import email
import imaplib #imap pop
from bs4 import BeautifulSoup

import os
import mimetypes
username = 'hungrypy@gmail.com'
password = 'iamhungry2016day19'
mail = imaplib.IMAP4_SSL("imap.gmail.com") # https://www.google.com/settings/security/l
esssecureapps
mail.login(username, password)
mail.select("inbox")
#Create new folder
# mail.create("Item2")
#list Folders
#mail.list()
result, data = mail.uid('search', None, "ALL")
inbox_item_list = data[0].split()
for item in inbox_item_list:
    result2, email_data = mail.uid('fetch', item, '(RFC822)')
    raw_email = email_data[0][1].decode("utf-8")
    email_message = email.message_from_string(raw_email)
    to_ = email_message['To']
    from_ = email_message['From']
    subject_ = email_message['Subject']
    date_ = email_message['date']
    counter = 1
    for part in email_message.walk():
        if part.get_content_maintype() == "multipart":
            continue
        filename = part.get_filename()
        content_type = part.get_content_type()
        if not filename:
            ext = mimetypes.guess_extension(content_type)
            if not ext:
                ext = '.bin'
            if 'text' in content_type:
                ext = '.txt'
            elif 'html' in content_type:
                ext = '.html'
            filename = 'msg-part-%08d%s' %(counter, ext)
        counter += 1
    #save file
    save_path = os.path.join(os.getcwd(), "emails", date_, subject_)
    if not os.path.exists(save_path):
        os.makedirs(save_path)
    with open(os.path.join(save_path, filename), 'wb') as fp:
        fp.write(part.get_payload(decode=True))
# if "plain" in content_type:
#     #print(part.get_payload())
#     pass
# elif "html" in content_type:
#     html_ = part.get_payload()
#     soup = BeautifulSoup(html_, "html.parser")
#     text = soup.get_text()
#     print(subject_)
#     print(text)
# else:
#     pass
```

```
#     #print(content_type)
# #email_message.get_payload()
```