



Chilkat CkPython Examples

[Chilkat](#) • [HOME](#) • [Android™](#) • [Classic ASP](#) • [C](#) • [C++](#) • [C#](#) • [C# UWP/WinRT](#) • [DataFlex](#) • [Delphi ActiveX](#) • [Delphi DLL](#) • [Visual FoxPro](#) • [Java](#) • [Lianja](#) • [MFC](#) • [Objective-C](#) • [Perl](#) • [PHP ActiveX](#) • [PHP Extension](#) • [PowerBuilder](#) • [PowerShell](#) • [PureBasic](#) • [Python](#) • [Ruby](#) • [SQL Server](#) • [Swift](#) • [Tcl](#) • [Unicode C](#) • [Unicode C++](#) • [Visual Basic 6.0](#) • [VB.NET](#) • [VB.NET UWP/WinRT](#) • [VBScript](#) • [Xojo Plugin](#) • [Node.js](#)

Python Examples

[Async](#)
[Certificates](#)
[ECC](#)
[Email Object](#)
[Encryption](#)
[FTP](#)
[HTML-to-XML/Text](#)
[Gzip](#)
[HTTP](#)
[IMAP](#)
[JSON](#)
[Jwt](#)
[MHT / HTML Email](#)
[PEM](#)
[PFX/P12](#)
[Java KeyStore \(JKS\)](#)
[POP3](#)
[RSA Encryption](#)
[MIME](#)
[REST](#)
[SCP](#)
[SMTP](#)
[Socket/SSL/TLS](#)
[SSH Key](#)
[SSH](#)
[SSH Tunnel](#)
[SFTP](#)
[Stream](#)
[Tar Archive](#)
[XML](#)
[XMP](#)
[Zip](#)

More Examples...

[OpenSSL](#)
[Google APIs](#)
[Google Cloud Storage](#)
[Azure Cloud Storage](#)
[Amazon S3 \(new\)](#)
[Google Drive](#)
[Dropbox](#)
[Firebase](#)
[ASN.1](#)
[PRNG](#)
[Amazon S3](#)
[Bounced Email](#)
[CSV](#)
[Diffie-Hellman](#)
[DKIM / DomainKey](#)
[DSA](#)
[NTLM](#)
[QuickBooks](#)
[Spider](#)
[Digital Signatures](#)

(Python) Search IMAP Mailbox for Email Matching Criteria

Searching an IMAP mailbox for messages that match search criteria.

Chilkat Python Downloads



**Python Module for Windows, Linux, MAC OS X,
Solaris, FreeBSD, and ARM Embedded Linux**

```
import sys
import chilkat

imap = chilkat.CkImap()

# Anything unlocks the component and begins a fully-functional 30-day trial.
success = imap.UnlockComponent("Anything for 30-day trial")
if (success != True):
    print(imap.LastErrorText())
    sys.exit()

# Connect to an IMAP server.
# Use TLS
imap.put_Ssl(True)
imap.put_Port(993)
success = imap.Connect("imap.someMailServer.com")
if (success != True):
    print(imap.LastErrorText())
    sys.exit()

# Login
success = imap.Login("admin@chilkatsoft.com","myPassword")
if (success != True):
    print(imap.LastErrorText())
    sys.exit()

# Select an IMAP mailbox
success = imap.SelectMailbox("Inbox")
if (success != True):
    print(imap.LastErrorText())
    sys.exit()

# We can choose to fetch UIDs or sequence numbers.
fetchUids = True

# Here are examples of different search criteria:

# Return all messages.
allMsgs = "ALL"

# Search for already-answered emails.
answered = "ANSWERED"

# Search for messages on a specific date.
# The date string is DD-Month-YYYY where Month is
# Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, or Dec.
onDate = "SENTON 05-Mar-2007"

# Search for messages between two dates. SENTBEFORE
# finds emails sent before a date, and SENTSINCE finds
# email sent on or after a date. The "AND" operation
# is implied by joining criteria, separated by spaces.
betweenDates = "SENTSINCE 01-Mar-2007 SENTBEFORE 05-Mar-2007"

# Another example of AND: find all unanswered emails
# sent after 04-Mar-2007 with "Problem" in the subject:
complexSearch1 = "UNANSWERED SENTSINCE 04-Mar-2007 Subject \"Problem\""

# Find messages with a specific string in the body:
bodySearch = "BODY \"problem solved\""

# Using OR. The syntax is OR <criteria1> <criteria2>.
# The "OR" comes first, followed by each criteria.
# For example, to match all emails with "Help" or "Question" in the subject.
```

```

# You'll notice that literal strings may be quoted or unquoted.
# If a literal contains SPACE characters, quote it:
orSearch = "OR SUBJECT Help SUBJECT Question"

# -----
# Strings are case-insensitive when searching....
# -----

# Find all emails sent from yahoo.com addresses:
fromSearch = "FROM yahoo.com"
# Find all emails sent from anyone with "John" in their name:
johnSearch = "FROM John"

# Find emails with the RECENT flag set:
recentSearch = "RECENT"

# Find emails that don't have the recent flag set:
notRecentSearch = "NOT RECENT"
# This is synonymous with "OLD":
oldSearch = "OLD"

# Find all emails marked for deletion:
markedForDeleteSearch = "DELETED"

# Find all emails having a specified header field with a value
# containing a substring:
headerSearch = "HEADER DomainKey-Signature paypal.com"

# Find any emails having a specific header field. If the
# 2nd argument to the "HEADER" criteria is an empty string,
# any email having the header field is returned regardless
# of the header field's content.
# Find any emails with a DomainKey-Signature field:
headerExistsSearch = "HEADER DomainKey-Signature \"\""

# Find NEW emails: these are emails that have the RECENT flag
# set, but not the SEEN flag:
newSearch = "NEW"

# Find emails larger than a certain number of bytes:
sizeLargerSearch = "LARGER 500000"

# Find emails marked as seen or not already seen:
seenSearch = "SEEN"
notSeenSearch = "NOT SEEN"

# Find emails having a given substring in the TO header field:
toSearch = "TO support@chilkatsoft.com"
# A more long-winded way to do the same thing:
toSearch2 = "HEADER TO support@chilkatsoft.com"

# Find emails smaller than a size in bytes:
smallerSearch = "SMALLER 30000"

# Find emails that have a substring anywhere in the header
# or body:
fullSubstringSearch = "TEXT \"Zip Component\""

# Pass any of the above strings here to test a search:
# messageSet is a CkMessageSet
messageSet = imap.Search(orSearch,fetchUids)
if (messageSet == None ):
    print(imap.lastErrorText())
    sys.exit()

# Fetch the email headers into a bundle object:

# bundle is a CkEmailBundle
bundle = imap.FetchHeaders(messageSet)
if (bundle == None ):

    print(imap.lastErrorText())
    sys.exit()

# Display the Subject and From of each email.

for i in range(0,bundle.get_MessageCount()):

    # email is a CkEmail
    email = bundle.GetEmail(i)

    print(email.getHeaderField("Date"))
    print(email.subject())
    print(email.ck_from())
    print("---")

# Disconnect from the IMAP server.

```

```
success = imap.Disconnect()
```