E-Mail Handling with Python

Dr. Uwe Ziegenhagen

25. Oktober 2021

www.uweziegenhagen.de

Introduction

- I am Vice-President of Dante e.V., the German LaTeX society
- My job: send e-mails to new members and manage mailing lists w.r.t. SPAM
- Automating these two tasks with Python has high ROI (return-on-investment)

https://github.com/UweZiegenhagen/E-Mail-Handling-in-Python

Handling E-Mails

Online Form at www.dante.de

Mitgliedsart:	◉ Privatperson (40,– EUR) ○ Institut (65,– EUR) ○ Firma (150,– EUR) ○ Privatperson – ermäßigt (Schüler) (15,– EUR) ○ Privatperson – ermäßigt (Studierender) (20,– EUR) ○ Privatperson – ermäßigt (Rentner) (20,– EUR) ○ Privatperson – ermäßigt wegen
Vor-/Zuname:	*
Institution:	
Straße bzw. Postfach:	*
PLZ:	*
Stadt:	*
Land (wenn nicht DE):	
E-Mail-Adresse:	*

Zusätzlich TUG Mitglied?	◉ Nein ○ Ja (Beiträge siehe hier.)
dante-ev@dante.de abonnieren?	◉ Nein ○ Ja (Vereinsmailingliste.)
Bankeinzug des Mitgliedsbeitrags	◉ Nein ○ Ja (bitte Formular SEPA-Lastschrift ausfüllen und zusenden)

Online Form at www.dante.de

```
Mitgliedsart:  P
Name:          Test Testerschmidt
Institution:   --
Straße/Postf.: Testweg 123
PLZ + Stadt:   12345 Teststadt
Land:          --
dante-ev:      N
TUG-Mitglied:  J
Bankeinzug:    N

Email:         uwe@ziegenhagen.info
```

Danke. Wir werden uns umgehend bei Ihnen melden.

Online Form at www.dante.de

```
Mitgliedsart:  P
Name:          Test Testerschmidt
Institution:   --
Straße/Postf.: Testweg 123
PLZ + Stadt:   12345 Teststadt
Land:          --
dante-ev:      N
TUG-Mitglied:  J
Bankeinzug:    N

Email:         uwe@ziegenhagen.info
```

Danke. Wir werden uns umgehend bei Ihnen melden.

Received E-Mail

```
Name:              Test Testerschmidt
Mitgliedsart:      P
Institution:       --
Strasse/Postf.:    Testweg 123
PLZ:               12345
Stadt:             Teststadt
Land:              --

Email:     uwe@ziegenhagen.info

dante-ev:          N
TUG-Mitglied:      J
Bankeinzug:        N
```

⇒ My answer depends on the membership type and the last three J/N values

Required Libraries

toml	read PW from config file
imaplib	read mails on server
email	create e-mail objects from server replies
traceback	to handle stack traces

Whole code is „hacky“ and mostly the result of copy & past, but it works…

toml

- toml = „Tom’s Obvious Minimal Language“
- Simple way to store configuration settings in external files
- Much similar to ini files

```
import toml

settings = toml.load('settings.toml')
FROM_EMAIL = settings['myemail']
```

Helper Function

Helper function to parse the mail

```
def parse_mail(text):
    zeilen = text.split('\n') # create array
    daten = {} # empty dict

    for zeile in zeilen:
        splits = zeile.split(':')
        if len(splits) == 2:
            daten[splits[0].strip()] = splits[1].strip()

    return daten
```

Reading mails from Gmail

- the hacky copy & paste part, see the Python file in github
- was quite a struggle: multipart-messages, base64, Ansi/ASCII/UTF8
- latest updates:
  - filter interesting mails on the server (`mail.search()`)
  - trying to get rid of multipart code
- Function returns array of all relevant mails

```python
def read_email_from_gmail():
    try:
        mail = imaplib.IMAP4_SSL(SMTP_SERVER)
        mail.login(FROM_EMAIL,FROM_PWD)
        mail.select('inbox')

        # data = mail.search(None, 'ALL')
        data = mail.search(None, '(SUBJECT "WWW-Formular: Mitgliedsantrag")')
        mail_ids = data[1]
        id_list = mail_ids[0].split()
        first_email_id = int(id_list[0])
        latest_email_id = int(id_list[-1])

        for i in range(latest_email_id,first_email_id-1, -1):
            data = mail.fetch(str(i), '(RFC822)')
            for response_part in data:
                arr = response_part[0]
                if isinstance(arr, tuple):
                    body = ''
                    msg = email.message_from_string(str(arr[1],'ansi'))
                    body = msg.get_payload(decode=False)
                    struct_data = parse_mail(body)
                    print(struct_data,'\n')
                    antraege.append(struct_data.copy())

    except Exception as e:
        traceback.print_exc()
        print(str(e))
```

Preparing answers

- Pre-defined templates for different scenarios: pays him/herself, TUG-membership y/n, reduced fee, etc.
- $\Rightarrow$ nasty if/elif/else statements in the `process_mails` function
- Templates are combined, replacements made, final text printed
- No automated sending, yet, need to build „trust“ in the code, rare scenarios not treated, yet.

Preparing answers

```
def process_antraege(antraege):
    mailtext = ''
    for antrag in antraege:
        if 'Bankeinzug' in antrag:
            if antrag['Bankeinzug'] == 'J':
                mailtext = '\n'.join([anrede % antrag['Name'], bankeinzug, schluss])
                print(mailtext,'\n\n\n')
            elif antrag['Bankeinzug'] == 'N' and antrag['TUG-Mitglied'] == 'N':
                    if antrag['Mitgliedsart'] == 'S':
                        beitrag = '20'
                        mailtext = '\n'.join([anrede % antrag['Name'],
                                              dante_ermaessigt_selbstzahler.replace('xx',beitrag) % '2021', schluss])
                    else:
                        beitrag = '20'
                        mailtext = '\n'.join([anrede % antrag['Name'],
                                              dante_selbstzahler.replace('xx',beitrag) % '2021', schluss])
            elif antrag['Bankeinzug'] == 'N' and antrag['TUG-Mitglied'] == 'J':
                    mailtext = '\n'.join([anrede % antrag['Name'],
                                          TUG_bekannt, schluss])
        elif 'Achtung' in antrag and 'SCHNUPPER' in antrag['Achtung']:
                    mailtext = '\n'.join([anrede % antrag['Name'],
                                          schnupper, schluss])

    print(mailtext,'\n\n' + '%' * 25 + '\n')
```

Managing Mailinglists

Mailinglists

- Dante e.V. provides various public/semi-public mailing lists
- SPAM is huge problem, some months ago: dozens of new signup-requests per day
- $\Rightarrow$ Automation necessary $\Rightarrow$ Selenium
- Selenium = web-testing framework, controls browser from Python

Selenium 1

- load selenium packages
- defines the sites to be handled
- load passwords via toml

```
from selenium.webdriver import Firefox
from selenium.webdriver.firefox.options import Options
from selenium.common.exceptions import NoSuchElementException
import toml

settings = toml.load('selenium_settings.toml')

opts = Options()
browser = Firefox(executable_path=r"C:\geckodriver-v0.27.0\geckodriver.exe", options=opts)
browser.implicitly_wait(3)

sites = {'https://lists.dante.de/mailman/admindb/dante-ev':settings['dante-ev'],
         'https://lists.dante.de/mailman/admindb/vorstand':settings['vorstand'],
         'https://lists.dante.de/mailman/admindb/beraterkreis':settings['beraterkreis'],
         'https://lists.dante.de/mailman/admindb/wwwmaint':settings['wwwmaint']}
```

Selenium 2

- login to each mailinglist
- enable deletion of all requests
- click submit button

```
for site in sites:
    browser.get(site)
    search_form = browser.find_element_by_name('adminpw')
    search_form.send_keys(sites[site])
    search_form.submit()

    try:
        field = browser.find_element_by_name('discardalldefersp')
        field.click()
        browser.implicitly_wait(3)
        submit = browser.find_element_by_name('submit')
        browser.implicitly_wait(5)
        submit.click()
    except NoSuchElementException:
        print('No new messages to be discarded for: ', site[39:])

    browser.implicitly_wait(3)
```

Selenium 3

- ban all new-user requests if they do not contain „)“

```
# Neue Anträge für dante-ev
# Kein Passwort notwendig, da gecached!
ANTRAGS_URL = 'https://lists.dante.de/mailman/admindb/dante-ev'
browser.implicitly_wait(3)
browser.get(ANTRAGS_URL)

fields = browser.find_elements_by_xpath("//input[@value='3']")
emails = browser.find_elements_by_xpath('//td[contains(text(),"@")]')
banfields = browser.find_elements_by_xpath('//input[contains(@name,"ban-")]')

if len(fields) == 0:
    print('No new requests to be discarded, closing browser')
    browser.close()
else:
    if len(fields) == len(emails) and len(fields) == len(banfields):
        zipped_list = list(zip(emails, fields, banfields))

        for i in zipped_list:
            email, field, banfield = i
            if not email.text.endswith(')'):
                field.click()
                banfield.click()
```