Міністерство освіти і науки, молоді та спорту України
Національний Технічний Університет
України
“Київський Політехнічний Інститут”
Факультет прикладної математики
Кафедра СПіСКС

Лабораторна робота № 1

з *дисципліни*

“Архітектура комп’ютерів 2”

Тема: “Засоби підтримки процесу розробки мовою Python”

Виконали:
Студенти групи КВ-73
Шевченко Д.
Романова Д.
Шуляк А.

Перевірив(-ла):
_____

Київ 2020 р.

Варіант 3

ПЗ для пошуку e-mail. Реалізувати проходження по сторінкам з набору url, які задані у вхідному xml-файлі, а також по сторінкам, на які є посилання з цих сторінок з заданою глибиною вкладеності. На всіх цих сторінках знайти всі e-mail-адреси та зберігти їх у файл в форматі xml. Урахувати те, що e-mail-адреси можуть бути записані у прихованому форматі, наприклад name(at)example.org.

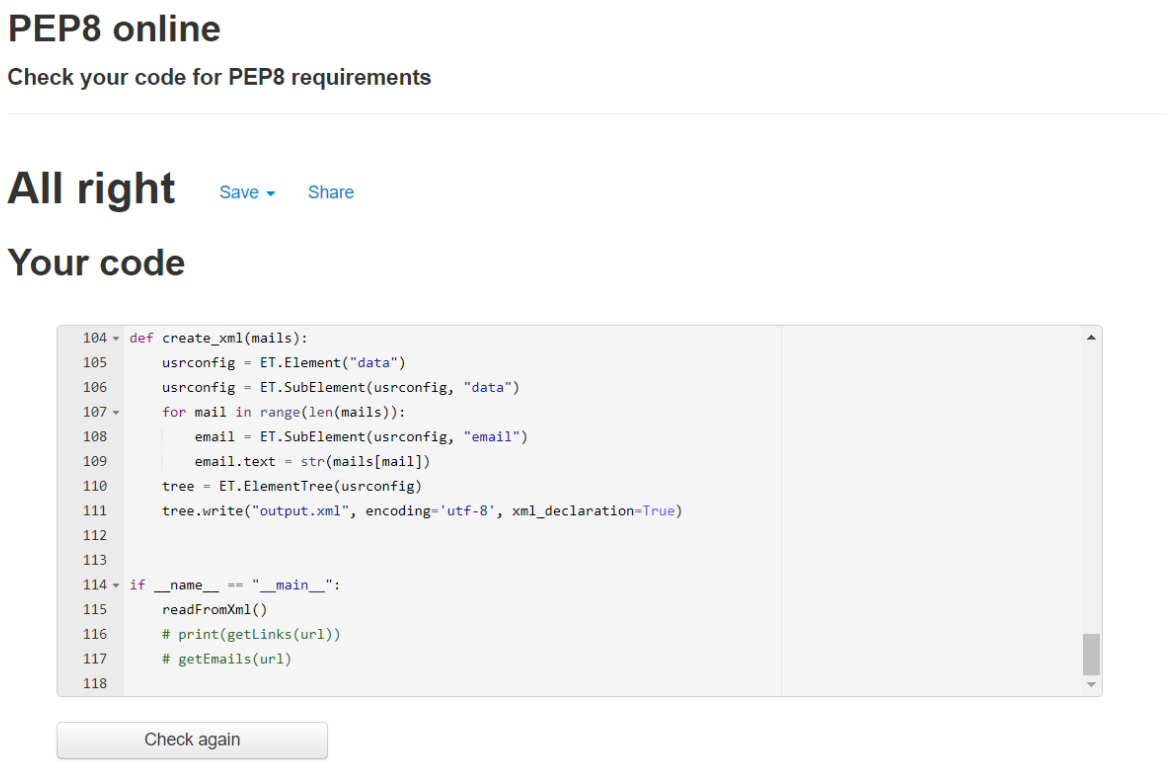
Приклад роботи програми

```
Depth:  1  Index:  38
/actions/gou-za-gopro-kameroj-i-poluchi-nabor-aksessuarov/
Depth:  1  Index:  39
https://www.citrus.ua/planshety/
Depth:  1  Index:  40
/poluchenie-posylki
Depth:  1  Index:  41
/kupit-v-kredit
Depth:  1  Index:  42
https://viber.com/citrus.ua
Depth:  1  Index:  43
/smartfony/brand-realme/
Depth:  1  Index:  44
/search?query=
Depth:  1  Index:  45
/kondicionery/konditsioner-electrolux-eacs-09hg-b2n3-666273.html
Depth:  1  Index:  46
/smartfony/brand-apple/seriya_iphone-se-2-2020/
Depth:  1  Index:  47
```

Середній час роботи

Звичайна – 290 секунд
З gevent – 199 секунд

Перевірка pep8



Код програми

emails.py – файл для пошуку emails у веб сторінці.

```
import requests
import re
from bs4 import BeautifulSoup

allLinks = [];
mails = []
url = 'https://docs.python.org/3/library/termios.html'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
links = [a.attrs.get('href') for a in
soup.select('a[href]')]
for i in links:
    if (("contact" in i or "Contact") or ("Career" in i or
"career" in i)) or ('about' in i or "About" in i) or (
            'Services' in i or 'services' in i):
        allLinks.append(i)
allLinks = set(allLinks)
```

```
def findMails(soup):
    for name in soup.find_all('a'):
        if (name is not None):
            emailText = name.text
            match = bool(re.match('[a-zA-Z0-9_.+-]+@[a-zA-
Z0-9-]+\.[a-zA-Z0-9-.]+$', emailText))
            if ('@' in emailText and match == True):
                emailText = emailText.replace(" ",
'').replace('\r', '')
                emailText = emailText.replace('\n',
'').replace('\t', '')
                if (len(mails) == 0) or (emailText not in
mails):
                    print(emailText)
                mails.append(emailText)


for link in allLinks:
    if (link.startswith("http") or link.startswith("www")):
        r = requests.get(link)
        data = r.text
        soup = BeautifulSoup(data, 'html.parser')
        findMails(soup)

    else:
        newurl = url + link
        r = requests.get(newurl)
        data = r.text
        soup = BeautifulSoup(data, 'html.parser')
        findMails(soup)

mails = set(mails)
if (len(mails) == 0):
    print("NO MAILS FOUND")
```

main.py – файл для рекурсивного пошуку на сайті усіх email-ів, з параметром глибини рекурсії.

```
import time

import requests
from bs4 import BeautifulSoup
import urllib.request
import re
from email_scraper import scrape_emails
from requests_html import HTMLSession
from selectolax.parser import HTMLParser
```

```
import xml.etree.ElementTree as ET


def findMails(soup, mails):
    for name in soup.find_all('a'):
        if (name is not None):
            emailText = name.text
            match = bool(re.match('[a-zA-Z0-9_.+-]+@[a-zA-
Z0-9-]+\.[a-zA-Z0-9-.]+$', emailText))
            if '@' in emailText and match:
                emailText = emailText.replace(" ",
'').replace('\r', '')
                emailText = emailText.replace('\n',
'').replace('\t', '')
                if (len(mails) == 0) or (emailText not in
mails):
                    print(emailText)
                mails.append(emailText)


def getEmails(url, depth, mails):
    if depth == 0:
        return
    time.sleep(1)
    print("Depth: ", depth)
    print("Processing url: ", url)

    allLinks = []
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    links = [a.attrs.get('href') for a in
soup.select('a[href]')]
    for i in links:
        if (("contact" in i or "Contact") or
            ("Career" in i or "career" in i)) \
                or ('about' in i or "About" in i) \
                or ('Services' in i or 'services' in i):
            allLinks.append(i)
    allLinks = set(allLinks)

    i = 0
    for link in allLinks:
        i += 1
        print("Depth: ", depth, " Index: ", i)
        try:
            if link.startswith("http") or
```

```
link.startswith("www"):
                time.sleep(1)
                # Recursive call
                depth -= 1
                getEmails(link, depth, mails)
                depth += 1
                #
                time.sleep(1)
                print(link)
                r = requests.get(link)
                data = r.text
                soup = BeautifulSoup(data, 'html.parser')
                findMails(soup, mails)

            else:
                time.sleep(1)
                # Recursive call
                depth -= 1
                getEmails(link, depth, mails)
                depth += 1
                #
                print(link)
                time.sleep(1)
                newurl = url + link
                r = requests.get(newurl)
                data = r.text
                soup = BeautifulSoup(data, 'html.parser')
                findMails(soup, mails)
        except Exception:
            print("Error: ", link)

    mails = set(mails)
    if len(mails) == 0:
        print("NO MAILS FOUND")


def readFromXml():
    mails = []
    tree = ET.parse('input.xml')
    root = tree.getroot()

    # one specific item attribute
    depth = int(root[0].text)
    # all item attributes
    print('\nAll attributes:')
    for elem in root:
```

```
        for subelem in elem:
            print(subelem.text)
            print("Loading all emails...")
            getEmails(subelem.text, depth, mails)
            print("All emails:\n", mails)
            create_xml(mails)
            mails = []


def create_xml(mails):
    usrconfig = ET.Element("data")
    usrconfig = ET.SubElement(usrconfig, "data")
    for mail in range(len(mails)):
        email = ET.SubElement(usrconfig, "email")
        email.text = str(mails[mail])
    tree = ET.ElementTree(usrconfig)
    tree.write("output.xml", encoding='utf-8',
xml_declaration=True)


if __name__ == "__main__":
    readFromXml()
    # print(getLinks(url))
    # getEmails(url)
```

mainGevent.py – файл для рекурсивного пошуку на сайті усіх email-ів, з параметром глибини рекурсії та використання gevent

```
import time

import requests
from bs4 import BeautifulSoup
import urllib.request
import re
from email_scraper import scrape_emails
from requests_html import HTMLSession
from selectolax.parser import HTMLParser
import xml.etree.ElementTree as ET
import gevent


def findMails(soup, mails):
    for name in soup.find_all('a'):
        if (name is not None):
            emailText = name.text
            match = bool(re.match('[a-zA-Z0-9_.+-]+@[a-zA-
Z0-9-]+\.[a-zA-Z0-9-.]+$',
```

```
                                    emailText))
            if '@' in emailText and match:
                emailText = emailText.replace(" ",
'').replace('\r', '')
                emailText = emailText.replace('\n',
'').replace('\t', '')
                if (len(mails) == 0) or (emailText not in
mails):
                    print(emailText)
                mails.append(emailText)


def getEmails(url, depth, mails):
    if depth == 0:
        return
    time.sleep(1)
    print("Depth: ", depth)
    print("Processing url: ", url)

    allLinks = []
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    links = [a.attrs.get('href') for a in
soup.select('a[href]')]
    for i in links:
        if (("contact" in i or "Contact") or
            ("Career" in i or "career" in i)) \
                or ('about' in i or "About" in i) \
                or ('Services' in i or 'services' in i):
            allLinks.append(i)
    allLinks = set(allLinks)

    i = 0
    for link in allLinks:
        i += 1
        print("Depth: ", depth, " Index: ", i)
        try:
            if link.startswith("http") or
link.startswith("www"):
                time.sleep(1)
                # Recursive call
                depth -= 1
                getEmails(link, depth, mails)
                depth += 1
                #
                time.sleep(1)
```

```
                print(link)
                r = requests.get(link)
                data = r.text
                soup = BeautifulSoup(data, 'html.parser')
                findMails(soup, mails)

            else:
                time.sleep(1)
                # Recursive call
                depth -= 1
                getEmails(link, depth, mails)
                depth += 1
                #
                print(link)
                time.sleep(1)
                newurl = url + link
                r = requests.get(newurl)
                data = r.text
                soup = BeautifulSoup(data, 'html.parser')
                findMails(soup, mails)
        except Exception:
            print("Error: ", link)

    mails = set(mails)
    create_xml(mails)
    if len(mails) == 0:
        print("NO MAILS FOUND")


def readFromXml():
    mails = []
    tree = ET.parse('input.xml')
    root = tree.getroot()
    threads = []
    # one specific item attribute
    depth = int(root[0].text)
    # all item attributes
    print('\nAll attributes:')
    for elem in root:
        for subelem in elem:
            print(subelem.text)
            print("Loading all emails...")
            threads.append(gevent.spawn(getEmails,
subelem.text, depth, mails))
            print("All emails:\n", mails)
    gevent.joinall(threads)
```

```
def create_xml(mails):
    usrconfig = ET.Element("data")
    usrconfig = ET.SubElement(usrconfig, "data")
    for mail in range(len(mails)):
        email = ET.SubElement(usrconfig, "email")
        email.text = str(mails[mail])
    tree = ET.ElementTree(usrconfig)
    tree.write("output.xml", encoding='utf-8',
xml_declaration=True)


if __name__ == "__main__":
    readFromXml()
    # print(getLinks(url))
    # getEmails(url)
```

runMain, runMainGevent – файли для тестування coverage.

```
import main

def test():
    main.readFromXml()

if __name__ == "__main__":
    test()
```

```
import main_gevent

def test():
    main_gevent.readFromXml()


if __name__ == "__main__":
    test()
```

У файлі output.xml знаходиться результат роботи програми.