

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 1**

з дисципліни “ Архітектура комп'ютерів”

|  |  |  |
| --- | --- | --- |
| Виконали  студенти III курсу  групи КП-74  Мережко Іван,  Сенеджук Андрій,  Козій Руслан, Журба Антон  (*прізвище, ім’я, по батькові*)  варіант № 5 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  доцент  Замятін Денис Станіславович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Мета роботи - оволодіти сучасним інструментарієм створення, розробки та підтримки програмних проектів з використанням мови програмування Python.

**Етапи виконання роботи**

1. Встановити програмні засоби для створення віртуального середовища virtualenv.
2. Створити віртуальне середовище для нового проекту.
3. Розробити програмні засоби (ПЗ) згідно з варіантом завдання, наведеним нижче.
4. Розроблені ПЗ повинні бути оформлені у вигляді пакета, написаного мовою Python. Взаємодію з мережею реалізувати засобами стандартної бібліотеки Python.
5. Додати до директорії проекту файл requirements.txt зі списком необхідних пакетів.
6. Створити набір модульних тестів для перевірки правильності роботи ПЗ.
7. Перевірити покриття проекту тестами за допомогою утиліти coverage. Покриття повинне бути не менше за 85%.
8. Перевірити відповідність розробленого програмного коду конвенціям PEP8 за допомогою утиліти pep8.
9. Переробити розроблені ПЗ з використанням бібліотеки Gevent. Розробити засоби перемикання між розробленими версіями на основі конфігураційного файлу.
10. Порівняти час роботи першої та другої версії ПЗ.

**Варіант 5**

**ПЗ для створення плейлистів. 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 посилання на mp3-файли, знайдені файли відфільтрувати за заданим жанром, який зберігається у ID3-запису, та результат зберігти у файл в форматі xml.**

**Код програми:**

CONTROLLER\_PROGRAM.py

from reader import XMLReader

from request import Request

import gevent

import gevent.monkey

gevent.monkey.patch\_socket()

from gevent.pool import Pool

import requests

class Controller:

def \_\_init\_\_(self):

self.reader: XMLReader = XMLReader("links.xml")

self.reader.to\_list()

self.request: Request = Request(self.reader.info) #list with links

# self.resultRelatedLinks = list()

self.geventList = list()

def scrapy\_info(self):

*'''*

*gets you tube links on songs from xml file and hrefs on similar songs*

*'''*

#adds you tube link to list

self.request.getLinkOnYouTube()

#gets similar links from current song

self.request.getSimilarLinks()

def control(self, request):

self.request.getRequest(request)

self.scrapy\_info()

def call\_request(self):

for request in self.request.request\_list:

self.control(request)

self.request.hrefOnSimilarSongs = self.unique(self.request.hrefOnSimilarSongs)

#links on you tube for similar links

self.request.getSimilarYouTubeLinks(self.request.hrefOnSimilarSongs)

self.request.hrefOnCurrentSong = self.sort\_by\_genre()

self.write\_to\_xml("result.xml")

def sort\_by\_genre(self):

sort = self.request.hrefOnCurrentSong

sort = sorted(sort, key = lambda i: i['genre'])

return sort

def write\_to\_xml(self, filename):

return self.reader.output(filename, self.request.hrefOnCurrentSong)

def unique(self, to\_be\_unique):

return list(set(to\_be\_unique))

def check\_urls(self, urls):

pool = Pool(len(urls))

for url in urls:

pool.spawn(self.call\_request)

pool.join()

return True

INQUIRY.py

from typing import List, Any

from bs4 import BeautifulSoup

import requests as req

class Request:

def \_\_init\_\_(self, request\_list):

self.baseUrl = 'http://last.fm'

self.soup = ''

self.hrefOnCurrentSong = list()

self.request\_list = request\_list

self.hrefOnSimilarSongs = list()

def getLinkOnYouTube(self):

*'''*

***:return****:*

*'''*

for x in self.soup.findAll('a', {'class': 'header-new-playlink'}):

self.hrefOnCurrentSong.append({"link" : x['href'], 'genre': self.getGenre()})

break

# self.hrefOnCurrentSong = list(set(self.hrefOnCurrentSong))

def getRequest(self, request):

*'''*

*sends get-request and the result wraps in beautiful soup*

***:param*** *request:*

***:return****: none*

*'''*

try:

result = req.get(request).text

self.soup = BeautifulSoup(result, 'html.parser')

except req.exceptions.ConnectionError:

print("Error")

def getSimilarLinks(self):

*'''*

*gets similar links from the page*

***:return****: none*

*'''*

countSongs = 0;

for x in self.soup.findAll('a', {'class': 'js-link-block-cover-link link-block-cover-link'}):

if countSongs == 3: break

countSongs+=1;

self.hrefOnSimilarSongs.append(x['href'])

def getSimilarYouTubeLinks(self, links):

for elem in links:

self.getRequest(self.baseUrl+elem)

self.getLinkOnYouTube()

def getGenre(self):

*'''*

*gets genre*

***:return****: the first genre that is on the page*

*'''*

for x in self.soup.findAll('li', {'class': 'tag'}):

# print(x.text)

return x.text

INQUIRY.\_Tpy

import unittest

from controller import Controller

class MyTestCase(unittest.TestCase):

tst = Controller()

def test\_contoller\_control(self):

self.tst.control("https://www.last.fm/ru/music/Nirvana/\_/Smells+Like+Teen+Spirit")

self.assertTrue(self.tst.reader.info, ['https://www.last.fm/ru/music/Nirvana/\_/Smells+Like+Teen+Spirit', 'https://www.last.fm/ru/music/Nirvana/\_/Come+as+You+Are', 'https://www.last.fm/ru/music/Kanye+West/\_/Stronger'])

def test\_controller\_(self):

self.tst.control("https://www.last.fm/ru/music/Nirvana/\_/Smells+Like+Teen+Spirit")

self.assertTrue(self.tst.request.hrefOnCurrentSong, [{'link': 'https://www.youtube.com/watch?v=hTWKbfoikeg', 'genre': 'grunge'}, {'link': 'https://www.youtube.com/watch?v=vabnZ9-ex7o', 'genre': 'grunge'}])

def test\_controller\_unique(self):

lst = ["a", "a, b"]

self.assertTrue(list(set(lst)), self.tst.unique(lst))

def test\_controller\_unique(self):

self.assertTrue(True, self.tst.write\_to\_xml("test.xml"))

def test\_controller\_unique(self):

self.assertTrue(True, self.tst.check\_urls(self.tst.request.request\_list))

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

main.py

**from CONTROLLER\_PROGRAM import Controller**

import configparser

if \_\_name\_\_ == "\_\_main\_\_":

config = configparser.ConfigParser()

config.read("config.ini")

try:

to\_run = int(config.get("program", "to\_run"))

except:

print("Wrong configurations")

controller: Controller = Controller()

if(to\_run == 0):

controller.call\_request()

else:

controller.check\_urls(controller.request.request\_list)

PERFORMING.py

import xml.etree.ElementTree as etree

import xml.etree.cElementTree as ET

class XMLReader:

def \_\_init\_\_(self, filename):

self.doc = etree.parse(filename).getroot()

self.info = list()

def to\_list(self):

for elem in self.doc:

print(elem.text)

self.info.append(elem.text)

def output(self, filename, lst):

root = ET.Element("root")

doc = ET.SubElement(root, "doc")

for elem in lst:

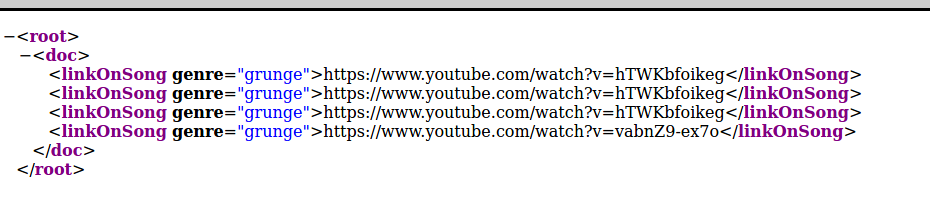
ET.SubElement(doc, "linkOnSong", genre=elem['genre']).text = elem['link']

tree = ET.ElementTree(root)

tree.write(filename)

return True

**Результат роботи:**



**Висновки**

Отже, в результаті виконання цієї лабораторної роботи, було опрацьовано процедурні конструкції мови Python та застосовано на практиці основний інструментарій розробника.

Розроблені програмні засоби розміщено у публічному сховищі програмного коду github.