**ფინალური გამოცდა (40 ქულა)**

AF-SM-04\_05.2020.G

|  |  |
| --- | --- |
| **სასწავლო კურსის სახელწოდება:** |  |
| **ლექტორი:** |  |
| **სტუდენტის სახელი და გვარი** |  |

**შუალედური გამოცდის დეტალური ინსტრუქცია**

*მოცემულია 3 სავარჯიშო. პირველ და მეორე დავალებაზე დაწერეთ შესაბამისი კოდი პითონის ფაილში. ორივე დავალება შეასრულეთ* ***ცალ-ცალკე ფაილში.*** *მესამე დავალება შედგება 5 ღია შეკითხვისგან. დავალებების შესრულების შემდეგ ატვირთეთ ორივე დავალების* ***py ფაილები*** *და* ***აღნიშნული word ფაილი, სადაც ღია შეკითხვებზე თქვენი პასუხებია მითითებული.*** *გადაამოწმეთ ატვირთული ფაილების სისწორე. გისურვებთ წარმატებებს!*

**სავარჯიშო 1 (12 ქულა) - Parser:**

მოცემულია **sample.html** ფაილი (იხილეთ თანდართული ფაილი). შეასრულეთ შემდეგი დავალებები:

* **(1 ქულა)** დაწერეთ კოდი, რომელიც წაიკითხვას html ფაილს.
* **(3 ქულა)** Beautiful Soup მოდულის გამოყენებით html კოდიდან წამოიღეთ მარცხენა მხარეს მოთავსებული უნივერსიტეტების დასახელებები და ჩამონათვალი შეინახეთ ცალკე სიის სახით.
* (**3 ქულა**) ცალკე ცვლადში შეინახეთ html-ში მოთავსებული Google-ის ლოგოს ლინკი
* **(5 ქულა)** აღნიშნული ლინკის და requests მოდულის გამოყენებით, დაწერეთ კოდი, რომელიც წამოიღებს სურათს ლინკიდან და შენახავს მას png ფორმატში თქვენს საქაღალდეში.

import os

from bs4 import BeautifulSoup

os.system('py   -m pip install bs4')

with open ('sample.html') as file:

    soup = BeautifulSoup(file,'html.parser')

list = []

nav = soup.find('nav',{'id':'center'})

li = soup.find\_all('a')

for i in li:

    list.append(i.text)

footer = soup.find('footer',{'id' : 'footer'})

logo = footer.find\_all('p')

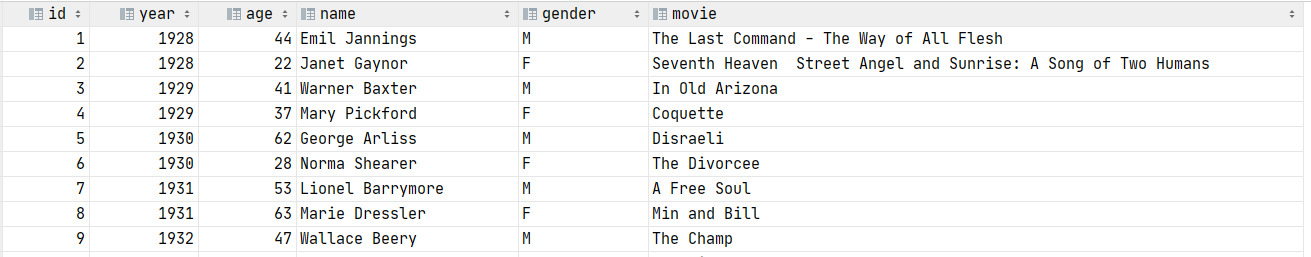
p = logo[1].img['src']

print(logo)

**სავარჯიშო 2 (13 ქულა):**

მარტივი ვებ აპლიკაცია მოცემულია **app.py** ფაილში (იხილეთ თანდართული ფაილი). აღნიშნულ ფაილში მოახდინეთ ბაზის ინტეგრირება და დაწერეთ კოდის ფრაგმენტი შემდეგი პირობების გათვალისწინებით.

* **(3 ქულა)** აღწერეთ oscar ცხრილის მოდელი კლასის სახით Flask-SQLAlchemy მოდულის გამოყენებით (oscar ცხრილში მონაცემები წარმოდგენილია შემდეგი სახით - იხ. ქვედა სურათი)
* **(3 ქულა)** კლასში დაამატეთ \_\_str\_\_() მეთოდი, რომელიც აღწერს ობიექტს სტრიქონის სახით.
* **(3 ქულა)** დაწერეთ ბრძანება, რომლის მეშვეობითაც შექმნით მონაცემთა ბაზას და ცხრილს.
* **(4 ქულა)** დაწერეთ ბრძანება, რომლის მეშვეობით დაამატებთ ახალ ჩანაწერს ბაზაში.

****

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'asdsa'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///oscar.sqlite'

db = SQLAlchemy(app)

class Oscar(db.Model):

    id = db.Column(db.Integer,primary\_key = True,autoincrement=True)

    year = db.Column(db.Integer)

    age = db.Column(db.Integer)

    name = db.Column(db.String(100))

    gender = db.Column(db.String(30))

    movie = db.Column(db.String(100))

    def \_\_str\_\_(self):

        return f"saxelia{self.name} filmis saxelia {self.movie}"

with app.app\_context():

    db.create\_all()

    obj1 = Oscar(year = 1928,age = 22,name = "janet gaynor",gender = 'F',movie = 'seventh heaven' )

    obj2 = Oscar(year = 1929,age = 41,name = "varner baxter",gender = 'M',movie = 'in old arizona' )

    db.session.add(obj1)

    db.session.add(obj2)

    db.session.commit()

@app.route('/')

def hello\_world():

    return 'Hello World!'

if \_\_name\_\_ == 'main':

    app.run()

**ღია კითხვები (15 ქულა):**

1. რა განსხვავებაა GET და POST request-ებს შორის. მოიყვანეთ შესაბამისი მაგალითები.

GET გამოიყენება მაშინ, როდესაც request-ის გაგზავნა ხდება URL-ის მეშვეობით. შესაბამისად, URL-ში ეთითება

საჭირო პარამეტრები (პარამეტრის სახელი და მნიშნველობა). გამოიყენება ‘&’ სიმბოლო პარამეტრების

ერთმანეთისგან გამოსაყოფად და ‘?’ ძირითადი მისამართისა და პარამეტრების ერთმანეთისგან გამოსაყოფად

POST გამოიყენება მონაცემების გასაგზავნად სერვერზე, რომლის შენახვა (ან განახლება) უნდა მოხდეს ბაზაში (ან სხვა

რესურში). POST მეთოდი გამოიყენება ფორმებში შეყვანილი ინფორმაციის სერვერზე გასაგზავნად.

POST მეთოდის ელემენტები არ არის ხილვადი, არ ხდება მისი ქეშირება, არ რჩება ბრაუზერის history-ში, მისი

ელემენტების სიგრძე არ არის ლიმიტირებული.

1. რას წარმოადგენს BeatifulSoup მოდულის find\_all() ფუნქცია და რა ტიპის მონაცემს აბრუნებს? მოიყვანეთ მაგალითი.

find\_all() ფუნქცია აბრუნებს პარამეტრად მითითებული ტეგების შიგთავს; თუ ეს ტეგი

რამდენიმეჯერ არის ნახსენები, აბრუნებს ყველას სიის სახით

print(soup.find\_all('ul', {'class':'mylist'}))

1. Flask-ის მოდულის გამოყენებით ვებ აპლიკაციაში როგორ არის შესაძლებელი მოხდეს კონკრეტულ route-ზე html ფაილის მიბმა. აღწერეთ დეტალურად.
2. რას წარმოადგენს csv მოდული? მოიყვანეთ რომელიმე ფუნქცია შესაბამისი მაგალითით.

Python-ში csv მოდულის გამოყენებით შესაძლებელია csv ფაილიდან მონაცემთა იმპორტი/წაკითხვა

(Read) და/ან მონაცემების ექსპორტი/ჩაწერა (Write) csv ფაილში. csv (Comma Separated Values) ფაილი

გამოიყენება დიდი რაოდენობის მონაცემების შესანახად ტექსტურ ფორმატში. CSV ფაილის გავს

excel-ის ფაილს, სადაც მონაცემები წარმოდგენილია ცხრილის სახით (სტრიქონების და სვეტების

მეშვეობით). csv ფაილში ცხრილის მონაცემები წარმოდგენილია ტექსტურად, რომლის თითოეული

ხაზზე არის ცხრილის თითო სტრიქონის მონაცემი, ხოლო მძიმე გამოიყენება უჯრების (სვეტების)

ერთმანეთისგან გამოსაყოფად.

1. რას ნიშნავს საიტის Deployment? მოიყვანეთ შესაბამისი მაგალითი და აღწერეთ პროცესი.

import os

from bs4 import BeautifulSoup

os.system('py   -m pip install bs4')

with open('sample.html') as file:

    soup = BeautifulSoup(file,'html.parser')

list = []

main = soup.find('main',{'id':'center'})

tr = soup.find\_all('th')

for i in tr:

    list.append(i.text)

nav = soup.find('nav',{'id':'left'})

h3 = nav.find\_all('h3')

heading =h3[1]

print(heading)

<iframe src="https://bing.com" width="70%"></iframe>