1. შექმენით პირველი კლასის იგივე უმაღლესი რიგის ფუნქცია, რომელიც <h1> …</h1> ტეგებში ჩასვამს, String ტიპის მონაცემებს btuAI და TeslaAI-ს და <p>…</p> ტეგებში ჩასვამს ასევე String ტიპის მონაცემს PythonAI-ს.

*def* html\_tag(tag):  
 *def* wrap\_text(msg):  
 *print*('<{0}>{1}</{0}>'.format(tag,msg))  
 *return* wrap\_text  
print\_h1=html\_tag('h1')  
print\_h1('btuAI ')  
print\_h1('TeslaAI')  
print\_p=html\_tag('p')  
print\_p('PythonAi')

**2.** Flask-ის REST API გარემოში შექმენით შემდეგი დასახელების კურსები, dictionaty ფორმატში:

Description:Python in AI,

course\_id:0,

name:Python AI Certificate,

site:btu.edu.ge

Description":"CCNA,

course\_id:1,

name:CCNA Certificate,

site:netacad.com

Description:Linux,

course\_id:2,

name:Linux Certificate,

site:netdevgroup.com

შესაბამისი HTTP მეთოდის მეშვეობით მოახდინეთ, ზემოთ აღწერილი ერთ-ერთი კურსის გასაღები სიტყვის განახლება, შემდეგი მონაცემით:

Description:BTULinux

დაწერეთ ბრძანება, რომელიც უნდა დაიწეროს cmd-ში, რომ წარმატებით შესრულდის მონაცემის განახლების მოქმედება

*from* flask *import* Flask  
app=Flask(\_\_name\_\_)  
courses =[  
{'Description':'Python in AI',  
 'course\_id':0,  
 'name':'Python AI Certificate',  
 'site':'btu.edu.ge'  
 },  
{  
 'Description':'CCNA',  
 'course\_id':1,  
 'name':'CCNA Certificate',  
 'site':'netacad.com'  
},  
{'Description':'Linux',  
 'course\_id':2,  
 'name':'Linux Certificate',  
 'site':'netdevgroup.com'  
 }  
]  
@app.route("/")  
*def* main():  
 *return* "hello"  
@app.route('/courses/<int:course\_id>')  
*def* coursese(course\_id):  
 *return* courses[course\_id]  
@app.route('/courses/<int:course\_id>',methods=['PUT'])  
*def* update(course\_id):  
 courses[course\_id]['Description']='BTU DESCRIPTION'  
 *return* courses[course\_id]  
*if* \_\_name\_\_=="\_\_main\_\_":  
 app.run(debug=*True*)  
*#curl -X PUT http://127.0.0.1:5000/courses/1*

1. Flask-ის REST API-ის გარემოში არსებული შემდეგი dictionary:

id:1,

name:Elon Musk,

author:Ashlee Vance

id:2,

name:Steve Jobs,

author:Walter Isaacson

ჩავაშენოთ მონაცემთა ბაზის ცხრილში.

*from* flask *import* Flask  
*import* dataset  
app=Flask(\_\_name\_\_)  
db=dataset.connect('sqlite:///database1412.db')  
books={  
 '1':{  
 "id":"1",  
 "name":"Elon Musk",  
 "author":"Ashlee Vance"  
 },  
 '2':{  
 "id":"2",  
 "name":"Steve Jobs",  
 "author":"Walter Isaacson"  
 }  
 }  
table=db['books']  
table.insert({  
 "book\_id":"1",  
 "name":"Elon Musk",  
 "author":"Ashlee Vance"  
 })  
table.insert({  
 "book\_id":"2",  
 "name":"Steve Jobs",  
 "author":"Walter Isaacson"  
 })  
@app.route('/')  
*def* api\_books():  
 *return* books  
*if* \_\_name\_\_=='\_\_main\_\_':  
 app.run()

**4.**Flask-ის REST API გარემოში შექმენით შემდეგი დასახელების კურსები, dictionaty ფორმატში:

Description:Python in AI,

course\_id:0,

name:Python AI Certificate,

site:btu.edu.ge

Description":"CCNA,

course\_id:1,

name:CCNA Certificate,

site:netacad.com

Description:Linux,

course\_id:2,

name:Linux Certificate,

site:netdevgroup.com

შესაბამისი HTTP მეთოდის მეშვეობით მოახდინეთ, ზემოთ აღწერილ კურსებიდან, ერთ-ერთი წაშლა, დაწერეთ ბრძანება, რომელიც უნდა დაიწეროს cmd-ში, რომ წარმატებით შესრულდის ერთ-ერთი კურსის წაშლა

*from* flask *import* Flask  
app=Flask(\_\_name\_\_)  
courses=[{  
 'Description':'Python in AI',  
 'course\_id':0,  
 'name':'Python AI Certificate',  
 'site':'btu.edu.ge'  
  
},  
 {'Description':'CCNA',  
 'course\_id':1,  
'name':' CCNA Certificate',  
'site': 'netacad.com'},  
 {'Description':'Linux',  
 'course\_id':2,  
 'name':'Linux Certificate',  
 'site':'netdevgroup.com'}  
]  
@app.route("/")  
*def* show():  
 *return* courses  
@app.route("/<int:course\_id>",methods=['DELETE'])  
*def* delete(course\_id):  
 courses.remove(courses[course\_id])  
 *return* courses  
*if* \_\_name\_\_=="\_\_main\_\_":  
 app.run(debug=*True*)

**curl -X DELETE http://127.0.0.1:5000/1**

**5.** შექმენით მონაცემთა ბაზა, რომელშიც იქნება ჩაშენებული ორი ცხრილი, დასახელებებით: gta და cities. ცხრილ gta-ში ჩავაშენოთ, შემდეგი მონაცემები:

**(1997, "GTA",'New Guernsey'),**

**(1999, "GTA", "USA"),**

**(2001, "GTA III","Liberty City"),**

**(2002, "GTA:Vice City", "Vice City"),**

**(2004, "GTA: San Andreas", "San Andreas"),**

**(2008, "GTA IV", "Liberty City")**

**ცხრილ cities-ში ჩავაშენოთ, შემდეგი მონაცემები:**

**("Liberty City", "New York")**

მოვახდინოთ, ორივე ცხრილის მონაცემების ცვლილება, კერძოდ ჩავანაცვლოთ New York-თ ის მნიშვნელობები, რომლებიც უტოლდება Liberty City-ს, სხვა შემთხევაში მივიღოთ შედეგები იტერაციის შედეგად.

*import* sqlite3  
db=sqlite3.connect("databaseSQL.sqlite3")  
list=[(1997, "GTA",'New Guernsey'),  
 (1999, "GTA", "USA"),  
 (2001, "GTA III","Liberty City"),  
 (2002, "GTA:Vice City", "Vice City"),  
 (2004, "GTA: San Andreas", "San Andreas"),  
 (2008, "GTA IV", "Liberty City")  
]  
cursor=db.cursor(  
)  
cursor.execute('''create table if not exists gta(year INTEGER,  
game text,  
city text)''')  
cursor.executemany("insert into gta values(?,?,?)",list)  
cursor.execute("""create table if not exists city(gta\_city text,real\_city text)""")  
listmeore=[("Liberty City", "New York")]  
cursor.executemany("insert into city values (?,?)",listmeore)  
cursor.execute("""update gta set city = 'New York' where city='Liberty City'""")  
cursor.execute("""update city set real\_city ='New York' where gta\_city ='Liberty City'""")  
cursor.execute("""select \* from gta where city is not 'New York'""")  
notnewyork=cursor.fetchall()  
*print*(notnewyork)  
db.commit()  
db.close()

#

**6.** შექმენით Closures(დახურული) ტიპი ფუნქცია ანუკო

დში აღწერეთ შემდეგი ფუნქციები: outer\_func() და inner\_func(), სადაც outer\_func()-ის შიგთავსში შექმნილი იქნება String ტიპის მონაცემის მქონე ცვლადი და inner\_func()-ის მეშვეობით გავუშვებთ მას ბეჭდვაზე, ხოლო outer\_func-ზე მიმართვით ავამუშავებთ პროგრამას**(10 ქულა)**

*def outer\_func():  
 message = 'Python'  
 def inner\_func():  
 print(message)  
 return inner\_func  
def outerFunction(text):  
 def innerFunction():  
 print(text)  
 return innerFunction  
a=outerFunction('Tesla')  
a()*

**7.** ბიბლიოთეკა BeautifulSoup-ის მეშვეობით მოახდინეთ საიტ https://realpython.github.io/fake-jobs/-ის div ტეგ ატრიბუტით id='ResultsContainer'-ზე წვდომა და მის შიგთავსში არსებულ ტეგ 'h2', კლასით class\_='title is-5'-ზე, რისი საშუალებითაც პროგრამის გაშვების შემთხვევაში, უნდა ხდებოდეს ვებ-გვერდზე არსებული სამსახურების დასახელებების გამოტანა IDE-ში.

*import* requests  
*from* bs4 *import* BeautifulSoup  
url='https://realpython.github.io/fake-jobs/'  
html=requests.get(url)  
tesla=BeautifulSoup(html.content, 'html.parser')  
results=tesla.find(id='ResultsContainer')  
job\_title=results.find\_all('h2', class\_='title is-5')  
*for* job *in* job\_title:  
 *print*(job.text)

8. Flask-ის REST API გარემოში შექმენით შემდეგი დასახელების კურსები, dictionaty ფორმატში:

Description:Python in AI,

course\_id:0,

name:Python AI Certificate,

site:btu.edu.ge

Description":"CCNA,

course\_id:1,

name:CCNA Certificate,

site:netacad.com

Description:Linux,

course\_id:2,

name:Linux Certificate,

site:netdevgroup.com

შესაბამისი HTTP მეთოდის მეშვეობით მოახდინეთ, ზემოთ აღწერილ კურსებზე, კიდევ შემდეგი კურსის დამატება:

Description:SQLserver,

course\_id:3,

name:SQLserver Certificate,

site:mygreatlearning.com

დაწერეთ ბრძანება, რომელიც უნდა დაიწეროს cmd-ში, რომ წარმატებით შესრულდის კიდევ ერთი კურსის დამატება.

*from* flask *import* Flask  
app=Flask("\_\_name\_\_")  
dicto=[{  
 'Description': 'Python in AI',  
 'course\_id': 0,  
 'name': 'Python AI Certificate',  
 'site': 'btu.edu.ge'  
},  
 {'Description':'CCNA',  
'course\_id':1,  
'name': 'CCNA Certificate',  
'site': 'netacad.com'  
},{  
'Description': 'Linux',  
'course\_id': 2,  
'name': 'Linux Certificate',  
'site': 'netdevgroup.com'  
 }]  
  
dasamatebeli={  
 'Description': 'SQLserver',  
 'course\_id': 3,  
 'name': 'SQLserver Certificate',  
'site':'mygreatlearning.com'  
  
}  
@app.route("/")  
*def* show():  
 *return* dicto  
  
@app.route("/",methods=["PUT"])  
*def* update():  
 dicto.append(dasamatebeli)  
 *return* dicto  
*if* \_\_name\_\_=="\_\_main\_\_":  
 app.run(debug=*True*)

**curl -X PUT http://127.0.0.1:5000/**

9. დაწერეთ კოდი, რომელიც თავის ლოგიკის მიხედვით მოახდენს ბიბლიოთეკა MechanicalSoup-ის მეშვეობით საიტ https://en.wikipedia.org/wiki/Comparison\_of\_Linux\_distributions -ის ტეგ: th-თან და მის კლასზე: table-rh\_თან დაკავშირებას და მასში არსებულ ინფორმაციაზე ექნება წვდომა, აგრეთვე დაწერეთ კოდი, რომლმაც უნდა მოახდინოს წვდომა საიტის td ტეგებზე. გამოიყენეთ ბიბლიოთეკა Pandas-ის DataFrame ფუნქცია, რომელიც მოახდენს მონაცემთა მოწესრიგებულ წარმოდგენას.

შექმენით მონაცემთა ბაზის სტრუქტურა, მასში ცხრილის ორგანიზებით ანუ სვეტების დასახელებების განსაზღვრით:

["Founder",

"Maintainer",

"Initial\_Release\_Year",

"Current\_Stable\_Version",

"Security\_Updates",

"Release\_Date",

"System\_Distribution\_Commitment",

"Forked\_From",

"Target\_Audience",

"Cost",

"Status"]

რომელშიც კოდის გაშვების შემთხვევაში მოხდება ზემოთ მითითებული საიტის ტეგებიდან იმპორტირებული, მონაცემების ჩაშენება.

*import* mechanicalsoup  
*import* pandas *as* pd  
*import* sqlite3  
openai = mechanicalsoup.StatefulBrowser()  
openai.open("https://en.wikipedia.org/wiki/Comparison\_of\_Linux\_distributions")  
th = openai.page.find\_all("th", attrs={"class": "table-rh"})  
distro = [value.text.replace("\n", "") *for* value *in* th]  
distro = distro[:95]  
td = openai.page.find\_all("td")  
columns = [value.text.replace("\n", "") *for* value *in* td]  
columns = columns[6:1051]  
column\_names = ["Founder",  
 "Maintainer",  
 "Initial\_Release\_Year",  
 "Current\_Stable\_Version",  
 "Security\_Updates",  
 "Release\_Date",  
 "System\_Distribution\_Commitment",  
 "Forked\_From",  
 "Target\_Audience",  
 "Cost",  
 "Status"]  
dictionary = {"Distribution": distro}  
*for* idx, key *in enumerate*(column\_names):  
 dictionary[key] = columns[idx:][::11]  
df = pd.DataFrame(data = dictionary)  
connection = sqlite3.connect("linux\_distro.db")  
cursor = connection.cursor()  
cursor.execute("create table linux (Distro, " + ",".join(column\_names)+ ")")  
*for* i *in range*(*len*(df)):  
 cursor.execute("insert into linux values (?,?,?,?,?,?,?,?,?,?,?,?)", df.iloc[i]) *#iloc=integer location*connection.commit()  
connection.close()

10. დაწერეთ კოდი, რომელიც თავის ლოგიკის მიხედვით მოახდენს ბიბლიოთეკების BeautifulSoup-ის, urllib.request-ის და re-ს მეშვეობით საიტ https://docs.python.org/3/library/random.html -ის ტეგებზე dt (ატრიბუტით id=random) და dd-ზე წვდომას, random ბიბლიოთეკის კლასების დასახელებების და მათი აღწერილობის გამოსატანად ბიბლიოთეკა Pandas-ის DataFrame ფორმატში.

# import requests  
# from bs4 import BeautifulSoup  
# import pandas as pd  
# url = "https://docs.python.org/3/library/random.html"  
# response = requests.get(url)  
# soup = BeautifulSoup(response.content, "html.parser")  
# functions = []  
# descriptions = []  
# dl\_tags = soup.find\_all("dl", class\_="function")  
# for dl in dl\_tags:  
# function\_name = dl.dt.get\_text(strip=True)  
# description = dl.dd.get\_text(strip=True)  
# functions.append(function\_name)  
# descriptions.append(description)  
# data = {"Function": functions, "Description": descriptions}  
# df = pd.DataFrame(data)  
# print(df)

11. Flask-ის გარემოში შექმენით წიგნების სია, მათი დასახელებით, ფასით და isbn კოდით, შემდეგ შესაბამისი url მისამართის საშუალებით, წიგნის isbn კოდის მიხედვით მოახდინეთ წიგნის დასახელების და ფასის, ბრაუზერის ეკრანზე გამოტანა.

*from* flask *import* Flask  
app = Flask("\_\_name\_\_")  
wignebi=[{  
 "Saxeli":"Saqartvelos wigni",  
 "fasi":"5lari",  
 "isbn":100  
  
},{  
 "Saxeli":"Tbilisis wigni",  
 "fasi": "5lari",  
 "isbn": 101  
}, {"Saxeli":"Samoqalaqo wigni",  
"fasi": "10lari",  
"isbn": 102  
 }]  
da=[]  
@app.route("/")  
*def* main():  
 *return* wignebi  
@app.route("/<int:isbn\_code>")  
*def* show(isbn\_code):  
 *for* i *in* wignebi:  
 *if* i["isbn"] == isbn\_code:  
 da={"Saxeli":i["Saxeli"],  
 "fasi":i["fasi"]}  
 *return* da  
  
*if* \_\_name\_\_=="\_\_main\_\_":  
 app.run(debug=*True*)

12. შექმენით Flask-ის Framework-ზე დაფუძნებული ინტერნეტ-მაღაზია, დიზაინის შესაქმნელად გამოიყენეთ Bootstrap, ჩააშენეთ რამოდენიმე გვერდი, შექმენით სარეგისტრაციო ფორმა, მიაბით მონაცემთა ბაზასთან.