Newer ones on top

import flask

from flask import request, jsonify

import re

import json

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

def get\_data(key,value,current):

print('get\_data()')

results = list()

pattern\_dict = {

'C' : '(C)',

'C++' : '(C\\+\\+)',

'Java' :'(Java)',

'C#' : '(C\\#)',

'Python' :'(Python)',

'Scala' : '(Scala)',

'Oracle' : '(Oracle)',

'SQL Server': '(SQL Server)',

'MySQL Server' :'(MySQL Server)',

'PostgreSQL':'(PostgreSQL)',

'MongoDB' : '(MongoDB)',

'JavaScript' : '(JavaScript)',

'Los Angeles' :'(Los Angeles)',

'New York':'(New York)',

'San Francisco':'(San Francisco)',

'Washington DC':'(Washington DC)',

'Seattle':'(Seattle)',

'Austin':'(Austin)',

'Detroit':'(Detroit)',

'technology':'(technology)'

}

for rec in current:

print(rec[key])

print(type(rec[key]))

print(rec[key].find(value))

#if rec[key].find(value) != -1:

import re

#reex\_str = """(C)|(C\+\+)|(JavaScript)|(Java)|(C\#)|(Python)|(Scala)|(Oracle)|(SQL Server)|(MySQL Server)|(PostgreSQL)|(MongoDB)"""

if re.search(pattern\_dict[value],rec[key]) != None:

results.append(rec)

return results

@app.route('/', methods=['GET'])

def home():

return '''<h1>Welcome to flask JOB search API</p>'''

@app.route('/data/all', methods=['GET'])

def api\_all():

print('api\_all()')

return jsonify(data)

@app.route('/data', methods=['GET'])

def api\_id():

# Check if keys such as Job Title,KeySkills, Role Category and others are provided as part of the URL.

# Assign the keys to the corresponding variables..

# If no key is provided, display an error in the browser.

print('app route data api\_id()')

res = None

for req in request.args:

if req == 'Job Title':

key = 'Job Title'

elif req == 'Job Experience Required' :

key='Job Experience Required'

elif req == 'Key Skills' :

key='Key Skills'

elif req == 'Role Category' :

key='Role Category'

elif req == 'Location' :

key='Location'

elif req == 'Functional Area' :

key='Functional Area'

elif req == 'Industry' :

key='Industry'

elif req == 'Role' :

key='Role'

elif req=="id":

key="id"

else:

#pass

print('bad key in api\_id() =', req)

value = request.args[key]

if (res==None):

print('res is None from api\_id(), get\_data')

res = get\_data(key,value,data)

else:

print('res is not None from api\_id()')

res = get\_data(key,value,res)

# Use the jsonify function from Flask to convert our list of

# Python dictionaries to the JSON format.

return jsonify(res) # note, not json, but response object, supposedly

print('done defs')

import warnings

warnings.filterwarnings("ignore", category=FutureWarning)

import re

#data = None

data = {}

myfile = r"c:\cygwin64\home\krista\READABLE\jobs.json"

#with open(r'c:\cygwin64\home\krista\READABLE\jobshort.json', mode='r', encoding='unicode\_escape').read().encode('raw\_unicode\_escape'):

print('entering open read')

with open(r'c:\cygwin64\home\krista\READABLE\lines2.json', mode='r', encoding='utf-8') as f:

longstring = f.read()

print('type of longstring is ', type(longstring))

# there are special characters and stuff in the file

# and jsonify does not seem to work, so clean it up and write to a new file

# below, white space or a newline is designated [\s\\n] and \* means any number of them

longstring.replace(r'[\n]+', '\\n') # get rid of excess blank lines

# ypeError: 'str' object cannot be interpreted as an integer

longstring.replace(r'^\s\*\\n$' , r'//') # get rid of lines with space only

replacements = {}

# ValueError: dictionary update sequence element #0 has length 1; 2 is required

replist = [{r',[\s\n]\*\]' : r'\\n]' }, # get rid of comma before ]

{r'\{[ \\n]\*\'' : r'{"' }, # change single quote to double after {

{r'\'\s \*\:' : r'"\:' }, # change single quote to double before :

{r'\:\s\*\'' : r':"' }, # change single quote after :

{r'\s\*\'\}' : r'"}'}, # change single quote before }

{r'\s\*\'\}' : r'"}'}, # change single quote before

{r'\\\\' : r'""' } ] # get rid of remaining backslashes

replacements = {k:v for e in replist for(k,v) in e.items()}

print ('entering for loop')

for nonjson, fixjson in replacements.items():

longstring = longstring.replace(nonjson, fixjson)

print('type of longstring is ', type(longstring))

with open(r"\cygwin64\home\krista\READABLE\newjobs.json", mode='w+', encoding='utf-8') as outy:

outy.write('longstring')

# then read outy into datalist?

with open(r'c:\cygwin64\home\krista\READABLE\newjobs.json', mode='r', encoding='utf-8') as f:

longstring = f.read()

datalist = json.load(longstring)

print('type of datalist is ',type(datalist)) # list of key:values

data = {k:v for e in datalist for(k,v) in e.items()}

print('type of data is ', type(data))

# returns JSON object as

# a dictionary; NO IT IS Returning List

#print(data) # very, very long string

# encoding='utf-8', mode='r') as f:

# print('data[0].keys()=', data[0].keys())

# print('data[0:3]=\n', data[0:3]) # Warning about control

# print('dumps(dict = ', json.dumps(data,indent=4,ensure\_ascii=False))

app.run()

!pip install flask

The dataset used in this lab comes from the following source: <https://www.kaggle.com/promptcloud/jobs-on-naukricom> under the under a **Public Domain license**.

Note: We are using a modified subset of that dataset for the lab, so to follow the lab instructions successfully please use the dataset provided with the lab, rather than the dataset from the original source.

The original dataset is a csv. We have converted the csv to json as per the requirement of the lab.

[2]:

**# !**wget https:**//**cf**-**courses**-**data.s3.us.cloud**-**object**-**storage.appdomain.cloud**/**IBM**-**DA0321EN**-**SkillsNetwork**/**labs**/**module**%**201**/**Accessing**%**20Data**%**20Using**%**20APIs**/**jobs.json

--2024-09-03 14:59:09-- <https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/jobs.json>

Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)... 169.63.118.104, 169.63.118.104

Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)|169.63.118.104|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 12878382 (12M) [application/json]

Saving to: ‘jobs.json’

jobs.json 100%[===================>] 12.28M 32.4MB/s in 0.4s

2024-09-03 14:59:09 (32.4 MB/s) - ‘jobs.json’ saved [12878382/12878382]

[\*]:

**import** flask

**from** flask **import** request, jsonify

**import** requests

**import** re

​

**def** get\_data(key,value,current):

results **=** list()

pattern\_dict **=** {

'C' : '(C)',

'C++' : '(C\+\+)',

'Java' :'(Java)',

'C#' : '(C\#)',

'Python' :'(Python)',

'Scala' : '(Scala)',

'Oracle' : '(Oracle)',

'SQL Server': '(SQL Server)',

'MySQL Server' :'(MySQL Server)',

'PostgreSQL':'(PostgreSQL)',

'MongoDB' : '(MongoDB)',

'JavaScript' : '(JavaScript)',

'Los Angeles' :'(Los Angeles)',

'New York':'(New York)',

'San Francisco':'(San Francisco)',

'Washington DC':'(Washington DC)',

'Seattle':'(Seattle)',

'Austin':'(Austin)',

'Detroit':'(Detroit)',

}

for rec in current:

print(rec[key])

print(type(rec[key]))

print(rec[key].find(value))

#if rec[key].find(value) != -1:

import re

#reex\_str = """(C)|(C\+\+)|(JavaScript)|(Java)|(C\#)|(Python)|(Scala)|(Oracle)|(SQL Server)|(MySQL Server)|(PostgreSQL)|(MongoDB)"""

if re.search(pattern\_dict[value],rec[key]) != None:

results.append(rec)

return results

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

import json

data = None

with open('jobs.json',encoding='utf-8') as f:

# returns JSON object as

# a dictionary

data = json.load(f)

with open(myfile, ‘r’,encoding='utf-8') as f:

@app.route('/', methods=['GET'])

def home():

return '''<h1>Welcome to flask JOB search API</p>'''

@app.route('/data/all', methods=['GET'])

def api\_all():

return jsonify(data)

@app.route('/data', methods=['GET'])

def api\_id():

# Check if keys such as Job Title,KeySkills, Role Category and others are provided as part of the URL.

# Assign the keys to the corresponding variables..

# If no key is provided, display an error in the browser.

res = None

for req in request.args:

if req == 'Job Title':

key = 'Job Title'

elif req == 'Job Experience Required' :

key='Job Experience Required'

elif req == 'Key Skills' :

key='Key Skills'

elif req == 'Role Category' :

key='Role Category'

elif req == 'Location' :

key='Location'

elif req == 'Functional Area' :

key='Functional Area'

elif req == 'Industry' :

key='Industry'

elif req == 'Role' :

key='Role'

elif req=="id":

key="id"

else:

pass

value = request.args[key]

if (res==None):

res = get\_data(key,value,data)

else:

res = get\_data(key,value,res)

# Use the jsonify function from Flask to convert our list of

# Python dictionaries to the JSON format.

return jsonify(res)

app.run()

\* Serving Flask app '\_\_main\_\_'

\* Debug mode: off

**WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.**

\* Running on [http://127.0.0.1:5000](http://127.0.0.1:5000/)