Python Basic Syntaxes

Json

Serialization

▼ How can we store images using pickle?

```
with open('roi.obj', 'wb') as roi_freader:
            pickle.dump(r, roi_freader)
with open('img.obj', 'wb') as image_freader:
            pickle.dump(image, image_freader)
```

▼ How can we load images using pickle?

```
pkl_file= open('data.pkl', 'rb')
image = pickle.load(pkl_file)
print image
pkl_file.close()
```

Serializing **Functions**

▼ How can we serialise functions?

```

```

Refernce: https://stackoverflow.com/questions/1253528/is-there-an-easy-way-to-pickle-a-python-function-or-otherwise-serialize-its-cod

Getting date Time

```
from datetime import datetime
now = datetime.now()
date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
print("date and time:",date_time)

#For other formats , you can check below
year = now.strftime("%Y")
print("year:", year)

month = now.strftime("%m")
print("month:", month)

day = now.strftime("%d")
print("day:", day)

time = now.strftime("%H:%M:%S")
print("time:", time)
```

Parsing Date Time from string

```
from datetime import datetime

datetime_object = datetime.strptime('Jun 1 2005  1:33PM', '%b %d %Y %I:%M%p')
```

Creating DateTimewise folder structure

```
def createDateWiseFoldersIfNotExist(path):
    import os
    import datetime
    current_date = datetime.datetime.now()
    if not os.path.exists(path+str(current_date.year)):
        os.mkdir(path+str(current_date.year))
    if not os.path.exists(path+str(current_date.year)+'/'+str(current_date.month)):
        os.mkdir(path+str(current_date.year)+'/'+str(current_date.month))
    if not os.path.exists(path+str(current_date.year)+'/'+str(current_date.month)+'/'+str(current_date.day)):
        os.mkdir(path+str(current_date.year)+'/'+str(current_date.month)+'/'+str(current_date.day))
```

JSON

▼ Loading Json data from a flie

```
# Python program to read
# json file
import json

def loadJsonData(filename):
  # Opening JSON file
  f = open('data.json')
  # returns JSON object as
  # a dictionary
  data = json.load(f)
  # Closing file
  f.close()
  return data

# Iterating through the json
# list
for i in loadJsonData(filename)['emp_details']:
  print(i)
```

▼ Writing JSON data into a file

```
# Python program to write JSON
# to a file


import json
def storeDictInJsonFile(UK_JsonObj,filename):
  # example for filename : rec_m_data.json
  jsonMessage=json.dumps(UK_JsonObj)
  with open(filename, "a+") as outfile:
          print('Writing to ',filename)
          outfile.write(jsonMessage+'\n')
# Data to be written
dictionary ={
  "name" : "sathiyajith",
  "rollno" : 56,
  "cgpa" : 8.6,
  "phonenumber" : "9976770500"
}
storeDictInJsonFile(dictionary, "newFile.json")
```

▼ Append Json data to a file

```
# Python program to write JSON
# to a file


import json
def storeDictInJsonFile(UK_JsonObj,filename):
  # example for filename : rec_m_data.json
```

```
  jsonMessage=json.dumps(UK_JsonObj)
  with open(filename, "a+") as outfile:
          print('Writing to ',filename)
          outfile.write(jsonMessage+'\n')
# Data to be written
dictionary ={
  "name" : "sathiyajith",
  "rollno" : 56,
  "cgpa" : 8.6,
  "phonenumber" : "9976770500"
}
storeDictInJsonFile(dictionary, "newFile.json")
```

▼ Add if not exist , edit if exist JSON file

```
def storeDictInJsonFile(dictionary ,filename):
  # Serializing json
  json_object = json.dumps(dictionary, indent = 4)
  # Writing to sample.json
  with open(filename, "w") as outfile:
    outfile.write(json_object)
def loadJsonData(filename):
  # Opening JSON file
  import os.path
  if(os.path.isfile(filename)):
    f = open('data.json')
    # returns JSON object as
    # a dictionary
    try:
      data = json.load(f)
    except:
      print("File exists but unable to load data")
    # Closing file
    f.close()
  return data
def loadEditOrAddData(filename,key,newVal):
  diction = loadJsonData(filename)
  diction[key]=newVal
  writeJsonToFile(diction,filename)
loadEditOrAddData("newFile.json","cgpa",7)
```

▼ What does json.dumps do?

json.dumps() function converts a Python object into a json string.

▼ Getting keys and values of dictionary

```
# Python program to get
# dictionary keys as list

def getList(dict):
    list = []
    for key in dict.keys():
        list.append(key)

    return list

# Driver program
dict = {1:'Geeks', 2:'for', 3:'geeks'}
print(getList(dict))
```

Adding new fields to dictionary

```
addPH({"Water_Temp":22.625}) =>  {"Water_Temp":22.625,"pH":5.158301}
```

▼ Code

```
def addSerialId(jsonDict,topicStr):
    serialId = topicStr.split('/')[3] #We can replace topicStr.split with any string
    jsonDict['serialId']=serialId
    return jsonDict
```

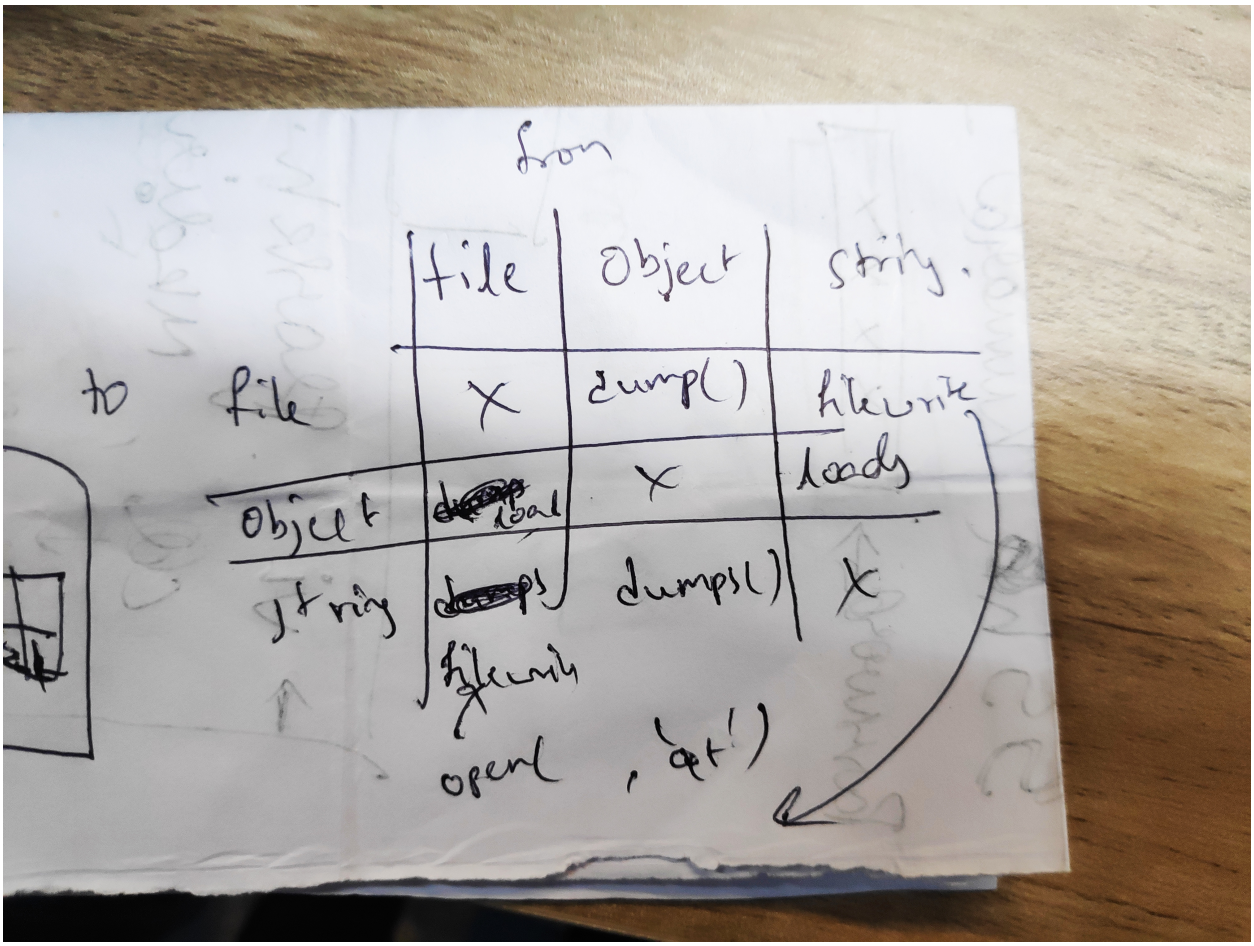
Combining multiple JSON params into one Single JSON Param:

```
[{"pH":5.158301},{"EC":1.415868},{"Water_Temp":22.625}]
                          ↓
{'pH': 5.158301, 'EC': 1.415868, 'Water_Temp': 22.625}
```

Let us say that we have list of multiple JSON parameters , if we want to combine it all into one single dictionary how can we do this ?

▼ Code

```
data = [{"pH":5.158301},{"EC":1.415868},{"Water_Temp":22.625}]
#Note : data here is a JSON object , not string.
for i in data:
  for key in i.keys():
    k[key]=i[key]
```



Getting spedific fields from list of dictionaries

Lets say that I have the following dictionary

```
{
    "table": "Products",
    "attributes": [
        {"pyt_field": "field1","sql_col":"col1","type":"str"},
        {"pyt_field": "field2","sql_col":"col2","type":"str"},
        {"pyt_field": "field3","sql_col":"col3","type":"str"},
        {"pyt_field": "field4","sql_col":"col4","type":"str"}
    ]
}
```

And from this I want only **pyt_field** .How can i do that?

▼ Code

For loadJsonData check this

```
config = loadJsonData('config.json')
[i['pyt_field'] for i in config['attributes']]
```

Attatching list with a specific delimitter

let us say I have list l = ['f','g','h']

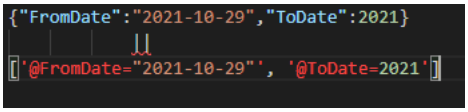
How do I get a string that just joins all of that list?

▼ Code

```
','.join(l)
```

List comprehension with if else logic

Requirement



▼ Code

```
def form_querry_string(json_data):
    cpm = ['@{}="{}"'.format(k,v) if type(v)==str else '@{}={}'.format(k,v) for k,v in json_data.items()]
    print(cpm)
```

/

List of dictionary to dictionary with "list under single key"

Requirement

▼ input

```
list_dict=[{
    "name":"ravi",
    "age":"23"
}
,{
    "name":"nitin",
    "age": "25"
```

```
},
{
    "name":"shruti",
 "age":"26"
}]
```

```
input_list_dict=[{
    "name":"ravi",
    "age":"23"
}
,{
    "name":"nitin",
    "age": "25"

},
{
    "name":"shruti",
 "age":"26"
}]
output=['name', 'age']
{'age': ['23', '25', '26'], 'name': ['ravi', 'nitin', 'shruti']}
```

▼ Code

```
list_keys = list(list_dict[0].keys())
print(list_keys)
diction = {}
for key in list_keys:
  for i in list_dict:
    if not key in diction:
      diction[key]=[]
    diction[key].append(i[key])
diction
```

Sending json data as a string (json data with list) to server

Keep in mind that for converting json to string you need to use **json.dumps(retList, separators=(',', ':'))**

```
retList = [{"filed": "Bimaid", "label": "Bima Intimation Id:", "status": "", "reason": "", "input_type": "input", "Image_ai_json": "", "Dam
import json
def createProperJsonData(retList):
    toUpload = {"jsondata": json.dumps(retList, separators=(',', ':')),"aI_status":1}
    return toUpload
insertfinalResultInDatabase(1,createProperJsonData(retList))
```

Use of Arguments(Parsing)

▼ How can we make python take in arguments? Sequential commands

```
import argparse
# Parse command line arguments
parser = argparse.ArgumentParser(
      description='Train Mask R-CNN to detect custom class.')
#this is in case we want just sequential commands
parser.add_argument("command",
                    metavar="<command>",
                    help="'train' or 'splash'")
args = parser.parse_args()

#The following code will check if command is train of splash
if args.command == "train":
        train(model)
    elif args.command == "splash":
        detect_and_color_splash(model, image_path=args.image,
                                video_path=args.video)
```

▼ How can we check for arguments with flags in python? "—weights" for example

```
import argparse
# Parse command line arguments
parser = argparse.ArgumentParser(
      description='Train Mask R-CNN to detect custom class.')
#this is in case we want to key value pair fashion
parser.add_argument('--weights', required=True,
                        metavar="/path/to/weights.h5",
                        help="Path to weights .h5 file or 'coco'")
args = parser.parse_args()
```

▼ How can we set arguments as optional? required = false

```
parser = argparse.ArgumentParser(
      description='Train Mask R-CNN to detect custom class.')
#this is in case we want to key value pair fashion
parser.add_argument('--weights', required=True,
                        metavar="/path/to/weights.h5",
                        help="Path to weights .h5 file or 'coco'")
```

In the syntax paser.add argument we can set required as False and it will take it as optional

▼ How can we set default values in arguments? (Take file location path as argument)

```
import argparse
import os
import sys
#Creating folder path
ROOT_DIR = ROOT_DIR = os.getcwd() #CWD stands for Current Working Directory
sys.path.append(ROOT_DIR)#This will generate the full path like ~/home/something
DEFAULT_MODEL_DIR = os.path.join(ROOT_DIR, "mask_rcnn_damage_0001.h5")
# Parse command line arguments
parser = argparse.ArgumentParser(
      description='Train Mask R-CNN to detect custom class.')
#using folder path as part of add_argument .
parser.add_argument('--weights', required=False,
                    default=DEFAULT_MODEL_DIR,
                    metavar="/path/to/weights.h5",
                    help="Path to weights .h5 file or 'coco'")
args = parser.parse_args()
```

We can set default variable in the function while setting required as false

File reader

▼ Reading data from file

```
def readFile(filename):
    line = ""
    with open(filename,'r') as f_reader:
        line += f_reader.read()
    print(line)
    return line
```

▼ readfile and split via newline

```
def splitFileIntoListOfDict(filename):
    contents = open(filename, "r").read()
    data = [json.loads(str(item)) for item in contents.strip().split('\n')]
```

```
    return data
splitFileIntoListOfDict('ukem')
```

▼ Check recursively in a current directory if files are list of directroy

```
import os
for i in os.listdir():
  print(os.path.isdir(i))
  print(os.path.isfile(i))
```

▼ Check recursively in a **specific directory that is not current working directory** if files are list of directory or not

```
import os
for i in os.listdir('output'):
  print(os.path.isdir(i))
  print(os.path.isfile(i))
```

▼ Replace data in a file

```
with open('file.txt', 'r') as file :
  filedata = file.read()

# Replace the target string
filedata = filedata.replace('ram', 'abcd')

# Write the file out again
with open('file.txt', 'w') as file:
  file.write(filedata)
```

▼ Reading one line after another

```
def splitDataNewLine(filename):
  with open(filename, 'r') as f:
    url_list = f.read().split("\n")
  return url_list
```

Creating a daemon service

First create a service in systemd file

sudo vi /lib/systemd/system/MQTTLogger.service

Then add following details within

```
[Unit]
Description=Flask Webpage
After=multi-user.target
Conflicts=getty@tty1.service

[Service]
Type=simple
ExecStart=/home/Amos/UK_MQTT_Json_Rep/MQTT-PythonVirtEnv/bin/python /home/Amos/UK_MQTT_Json_Rep/basicFlaskHTML.py
StandardInput=tty-force

[Install]
WantedBy=multi-user.target
```

Reload Deamon services

sudo systemctl daemon-reload

Enable and start Daemon service

```
sudo systemctl enable MQTTLogger.service
sudo systemctl start MQTTLogger.service
```

To check the status of service

sudo systemctl status MQTTLogger.service

To stop a service

sudo systemctl stop MQTTLogger.service

Link for further reference : https://tecadmin.net/setup-autorun-python-script-using-systemd/

How to make changes to the code specifically

Warnings

How can we ignore warnings?

```
import warnings
warnings.filterwarnings("ignore")
```

Dictionaries

How can we invert key value pairs in dictionary?

▼ Answer:

```
inv_map = {v: k for k, v in dict_object.items()}
inv_map
```

Getting string of json

▼ Answer

```
import json
a={'1':'num'}
str_dict = json.dumps(a)
```

Json adding dictionary to its own dictionary

▼ Answer

```
def addselfJson(jsonDict):
    jsonDict['selfJson']=dict(jsonDict)
    return jsonDict
a={'a':'1'}
a = addselfJson(a)
```

Tuples

▼ How do we append tuples?

```
output_tuple = IMAGE_SHAPE+(3,)
```

File Navigation

What does os.path.join do?

▼ Answer:

It creates a path by merging all locations that we give as input .

Exmaple :

```
path = os.path.join("/Users/James/tutorials", "index.html")
print(path)
```

This will return us **/Users/James/tutorials/index.html**

▼ What is code to unzip files programatically via python?

```
# Unzip the downloaded file
zip_ref = zipfile.ZipFile("10_food_classes_10_percent.zip", "r")
zip_ref.extractall()
zip_ref.close()
```

Create folder and get its path location

```
#Create Folder and get its path
def create_folder_and_get_path(folder_name):
    if not os.path.exists(folder_name):
        os.mkdir(folder_name)
    return os.path.join(os.getcwd(), folder_name)
```

Python MSSQL connector functions

Connecting to DB

```
import pymssql
def connectToDb():
    server = '185.136.157.12'
    user = 'ukws_user'
    paswd = 'NWzDRtnMDTxf5DqT'
    database = 'ukwebscrapper'
    conn = pymssql.connect(server, user, paswd, database)
    return [conn.cursor(),conn]
cursor,conn = connectToDb()
```

Numpy arrays

Filtering via boolean mask

Let us say that we are trying to fetch specific variables from list

idx_mask = [True , False]

We want corresponding elements from numpy array .

l = [1,2] ⇒ idx_mask ⇒[1]

Answer:

```
nd[idx_mask]
```

Filtering via boolean mask over numpy array .

Let us say that we have numpy array of shape (320,512,2)

We want to get apply the idx_mask over this but it gives me an issue

```
     11 print(r['rois'][idx_mask])
---> 12 print(r['masks'][idx_mask])
     13 print(r['class_ids'][idx_mask])
     14 print(r['scores'][idx_mask])

IndexError: boolean index did not match indexed array along dimension 0; dimension is 339 but corresponding boolean dimension i
s 2
```

How can we accomplish this?

Answer:

```
r['masks'][:,:,idx_mask]
#note : r['masks'] is an numpy array
#NOTE 2: dim(r['masks'][2])==dim(idx_mask)
```

Since we are trying to filter only the last dimension, we can apply over boolen mask on THAT particular dimension. Keep in mind that this is known as **Broadcasting** .And to perform broadcasting , the shape of the boolean mask **must be the same as sub-dimension(In this case 2)**

XML to JSon conversion

▼ Code

Link for reference :

```
# Program to convert an xml
# file to json file

# import json module and xmltodict
# module provided by python
import json
import xmltodict


# open the input xml file and read
# data in form of python dictionary
# using xmltodict module
with open("test.xml") as xml_file:

  data_dict = xmltodict.parse(xml_file.read())
  xml_file.close()

  # generate the object using json.dumps()
  # corresponding to json data

  json_data = json.dumps(data_dict)

  # Write the json data to output
  # json file
  with open("data.json", "w") as json_file:
    json_file.write(json_data)
    json_file.close()
```

RSS to Json conversion

▼ Code

Link for reference : https://rss2json.com/rss-to-json-api-python-example

```
import urllib2
import json


url = "https://api.rss2json.com/v1/api.json?rss_url=https%3A%2F%2Fnews.ycombinator.com%2Frss"

response = urllib2.urlopen(url)

data = json.loads(response.read());


print '====== ' + data['status'] + ' ======';

for item in data['items']:
    print item['title']
```

Running function Regular intervals

Basic class for task repeaters

```
argsfrom threading import Timer, Lock


class Periodic(object):
    """
    A periodic task running in threading.Timers
    """

    def __init__(self, interval, function, *args, **kwargs):
        self._lock = Lock()
        self._timer = None
        self.function = function
        self.interval = interval
        self.args = args
        self.kwargs = kwargs
        self._stopped = True
        if kwargs.pop('autostart', True):
            self.start()

    def start(self, from_run=False):
        self._lock.acquire()
        if from_run or self._stopped:
            self._stopped = False
            self._timer = Timer(self.interval, self._run)
            self._timer.start()
            self._lock.release()

    def _run(self):
        self.start(from_run=True)
        self.function(*self.args, **self.kwargs)

    def stop(self):
        self._lock.acquire()
        self._stopped = True
        self._timer.cancel()
        self._lock.release()
```

Implementation of python class

```
from time import sleep

def hello(name):
    print "Hello %s!" % name

print "starting..."
rt = Periodic(1, hello, "World") # it auto-starts, no need of rt.start()
try
    sleep(5) # your long-running job goes here...
finally:
    rt.stop()
```

More interesting answers can be found here : https://stackoverflow.com/questions/2398661/schedule-a-repeating-event-in-python-3

Pandas Dataframe

▼ Noob errors that you want to avoid:

1)NEVER pass original dataframe as a function , always pass a copy

Example :

WRONG PRACTICE

```
snatchJson(df)
```

CORRECT PRACTICE

```
snatchJson(df.copy())
```

Can we append data to an existing excel sheet?

Inefficient : Yes we can but we would first have to load the data and then push the data.

Efficient : Append data to json and then load the data as dataframe

Loading list of dictionary in to pandas dataframe

```
lod = [{"a":1,"b",2},{"a":3,"b",4}]
df = pd.DataFrame(lod)
```

Dropping duplicates

```
df_club = df_club.drop_duplicates(subset=['Amazon_ID'])
```

Dropping dupicates INCLUDING THE SOURCE

```
df_club = df_club.drop_duplicates(subset=['Amazon_ID'],keep=False)
```

Renaming columns

```
# Import pandas package
import pandas as pd

# Define a dictionary containing ICC rankings
rankings = {'test': ['India', 'South Africa', 'England',
               'New Zealand', 'Australia'],
      'odi': ['England', 'India', 'New Zealand',
               'South Africa', 'Pakistan'],
      't20': ['Pakistan', 'India', 'Australia',
               'England', 'New Zealand']}

# Convert the dictionary into DataFrame
rankings_pd = pd.DataFrame(rankings)

# Before renaming the columns
print(rankings_pd)

#This will rename all the correct columns
```

```
rankings_pd.rename(columns = {'test':'TEST'}, inplace = True)

# After renaming the columns
print("\nAfter modifying first column:\n", rankings_pd.columns)
```

Loading pandas Dataframe to SQL

```

```

Converting pandas dataframe to list of tuples

```
[tuple(x) for x in df.to_numpy()]
```

Converting pandas dataframe to list of dictionaries

```
df.to_dict('records')
```

Converting specific column as list in pandas dataframe

```
[i for i in df['event_json']]
or list(df['event_json'])
```

Converting pandas date time column to list of python Datetime

```
df = pd.read_excel('dummy.xlsx')
time_stamp_list = list(df['Transdate'])
[ind_time_stamp.strftime("%m/%d/%Y, %H:%M:%S") for ind_time_stamp in time_stamp_list]
```

Inserting list into a specific column of dataframe

If col doesent exist :

```
if(not len(df['Time'])==len(list_to_insert)):
  print("Error : Length doesent match")
else :
  df['Time']=list_to_insert
```

If col already exists

List

0	99,636	2021-11-02 04:3	349246F23A08
1	99,619	2021-11-02 04:3	349246F23A08
2	99,605	2021-11-02 04:1	349246F23A08
3	99,589	2021-11-02 04:1	349246F23A08
4	99,573	2021-11-02 04:1	349246F23A08

Replacing **RIGHT(List) in LEFT(Pandas)**

Pandas

0	99,636	2021-11-02 04:3	349246F23A08
1	99,619	2021-11-02 04:3	349246F23A08
2	99,605	2021-11-02 04:1	349246F23A08
3	99,589	2021-11-02 04:1	349246F23A08
4	99,573	2021-11-02 04:1	349246F23A08
5	99,557	2021-11-02 04:0	349246F23A08
6	99,542	2021-11-02 04:0	349246F23A08
7	99,528	2021-11-02 04:0	349246F23A08
8	99,511	2021-11-02 03:5	349246F23A08
9	99,497	2021-11-02 03:5	349246F23A08
10	99,484	2021-11-02 03:5	349246F23A08

```
#First we read the excel file and load it to dataframe prior to this .
list_to_insert = list(df['Transdate'])[:200]
#df[Time] has length 400 while list to insert is 200
minIndex = min(len(df['Time']),len(list_to_insert))
print(len(df.iloc[:minIndex,2]),len(list_to_insert[:minIndex]))
df.iloc[:minIndex,2]=list_to_insert[:minIndex]
```

NOTE : 2 is the index of that column in this case its **Transdate at index 2**

Adding pandas column to pandas dataframe json column

Before

Transdate	deviceid	farmid	jsondata
2021-11-02 04:3	349246F23A08	430	{'pH': 2.568364, 'EC': 2.812788, 'Water_Temp': 26.25, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2}
2021-11-02 04:3	349246F23A08	430	{'pH': 2.568364, 'EC': 2.812788, 'Water_Temp': 26.25, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2}
2021-11-02 04:1	349246F23A08	430	{'pH': 2.664879, 'EC': 2.812788, 'Water_Temp': 26.25, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2}
2021-11-02 04:1	349246F23A08	430	{'pH': 2.608579, 'EC': 2.794159, 'Water_Temp': 26.25, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2}

After

349246F23A08	430	{'pH': 2.632707, 'EC': 2.324636, 'Water_Temp': 25.8125, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 11:14:22'}
349246F23A08	430	{'pH': 2.777479, 'EC': 2.324636, 'Water_Temp': 25.8125, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 11:09:22'}
349246F23A08	430	{'pH': 2.375335, 'EC': 2.333183, 'Water_Temp': 25.8125, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 11:04:22'}
349246F23A08	430	{'pH': 2.117962, 'EC': 2.333183, 'Water_Temp': 25.8125, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 10:59:21'}
349246F23A08	430	{'pH': 2.174262, 'EC': 2.335844, 'Water_Temp': 25.75, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 10:54:21'}
349246F23A08	430	{'pH': 2.689008, 'EC': 2.335844, 'Water_Temp': 25.75, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 10:49:20'}
349246F23A08	430	{'pH': 2.761394, 'EC': 2.352956, 'Water_Temp': 25.75, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 10:44:20'}
349246F23A08	430	{'pH': 2.793566, 'EC': 2.335844, 'Water_Temp': 25.75, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 10:39:19'}
349246F23A08	430	{'pH': 2.214477, 'EC': 2.335844, 'Water_Temp': 25.75, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 10:34:19'}
349246F23A08	430	{'pH': 2.22252, 'EC': 2.335844, 'Water_Temp': 25.75, 'Grow_Stand': 1, 'Set_ec': 1.8, 'Set_ph': 6.2, 'Time': '11/02/2021, 10:29:18'}

```
def addParamToJsonInDf(df,json_col = 'jsondata',add_col = 'sno',new_key='sno'):
    json_col_data = df[json_col]
#     print(json_col_data)
    json_list = list(json_col_data)
#     print(json_list)
    val_to_insert = list(df[add_col])
    # json_list = list(json_col)
    print(len(json_list),len(val_to_insert))
#         print('str :',isinstanc)
    for i in range(len(json_list)):
#         print('str :',isinstance(json_list[i],str))
        if(isinstance(json_list[i],str)):
            json_list[i]=json_list[i].replace("'",'"')
        try:
            json_list[i]=json.loads(json_list[i])
        except Exception as e:
            print(type(json_list[i]) ," - ",e)
#         print("JSON list : ",json_list[i])
        print(i,type(val_to_insert[i]))
        if(isinstance(val_to_insert[i],pd._libs.tslibs.timestamps.Timestamp)):
            json_list[i][new_key]=val_to_insert[i].strftime("%m/%d/%Y, %H:%M:%S")
        else:
            json_list[i][new_key]=val_to_insert[i]
#         print("ind_json : ",ind_json)
        df[json_col]=json_list
    return df
```

Converting JSON file to excel sheet

```
import pandas as pd
import json
def splitFileIntoListOfDict(filename):
    contents = open(filename, "r").read()
    data = [json.loads(str(item)) for item in contents.strip().split('\n')]
    return data
df = pd.DataFrame(splitFileIntoListOfDict('rawdata.txt'))
df.to_excel('finalData.xlsx')
```

Checking instances of data

How to check if data is of type list?

```
element = {'a':2}
isinstance(element,dict)
```

Converting string to DatetimeIndex format

```
pd.DatetimeIndex(df{'Date'}).year
```

Regex

Get all elements with only alphabets

```
import re
testSearch = 'Hello gsd 123'
match = re.findall('[a-zA-Z]+',testSearch)
print(match)
```

Regex within a string where I get all numbers and letters

```
#Create a regex where within a string I get all numbers and letters and return a list of them
def get_numbers_and_letters(string):
    import re
    return re.findall(r'[0-9a-zA-Z]+', string)
```

Regex within a string where I get everything EXCEPT numbers and letters

```

```

Miscalaneous Exercises

Running simple HTTP server

```
For python2
  python -m SimpleHTTPServer 8000
For python3
  python -m http.server 8000
```

Using Icecream

```
pip install icecream
```

Dont even bother , this guy wrote much better documentation than you

https://towardsdatascience.com/do-not-use-print-for-debugging-in-python-anymore-6767b6f1866d

HTTPS Flask with python

Uploading image through form via python

```
def uploadImageAndReturnImageLink(filePath):
    import requests
    url = 'https://ezeeapi.meanhost.in/v1/api/Upload/upload'
    files = {'media': open(filePath, 'rb')}
    header_str = 'bearer '+tokenAuth()
    headers = {'Authorization':header_str}
    resp = requests.post(url, files=files,headers=headers)
    retURLString = 'https://ezeeapi.meanhost.in/uploadimage/' + json.loads(resp.content.decode())['document']
    return retURLString
```

Python requests and updating fields on database

PDF compression

https://www.thepythoncode.com/article/compress-pdf-files-in-python