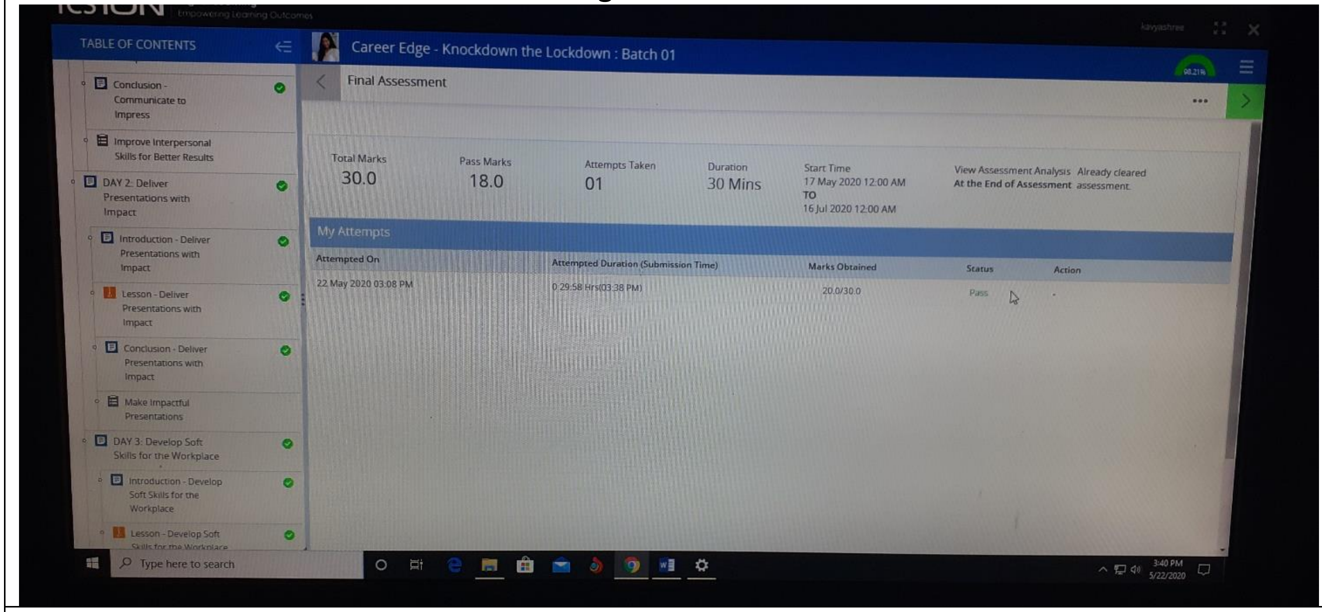
DAILY ASSESSMENT

Date:	**22-5-2020**	**Name:**	**Kavyashree m**
Course:	**TCS ioN**	**USN:**	**4al15ec036**
Topic:	**Technical competency, Understand artificial intelligence**	**Semester & Section:**	**8th A**
Github Repository:	**kavya**		

FORENOON SESSION DETAILS

Image of session



Career Edge - Knockdown the Lockdown : Batch 01

Final Assessment

Total Marks	Pass Marks	Attempts Taken	Duration	Start Time	View Assessment Analysis	
30.0	18.0	01	30 Mins	17 May 2020 12:00 AM TO 16 Jul 2020 12:00 AM	At the End of Assessment	Already cleared assessment.

My Attempts

Attempted On	Attempted Duration (Submission Time)	Marks Obtained	Status	Action
22 May 2020 03:08 PM	0:29:58 Hrs(03:38 PM)	20.0/30.0	Pass	-

TABLE OF CONTENTS
Conclusion - Communicate to Impress
Improve Interpersonal Skills for Better Results
DAY 2: Deliver Presentations with Impact
Introduction - Deliver Presentations with Impact
Lesson - Deliver Presentations with Impact
Conclusion - Deliver Presentations with Impact
Make Impactful Presentations
DAY 3: Develop Soft Skills for the Workplace
Introduction - Develop Soft Skills for the Workplace
Lesson - Develop Soft Skills for the Workplace
Type here to search

Technical competency

What do recruiters expect?

1)Quality vs Quantity

- ➢ Interviewers seem to knowledge on various technologies like active directory,natural language processing, internet of things.
- ➢ Most of this knowledge is superficial and not really working knowledge

2) BASIC IT skills

- ➢ Oops concepts-relate to realworld coding examples
- ➢ Any one programming language

- HTML/JS/CSS(Simple HTML page,with validations ,simply styling)
- Schema/tables/columns/indexes-SQL(DDL/DML/DCL/TCL)

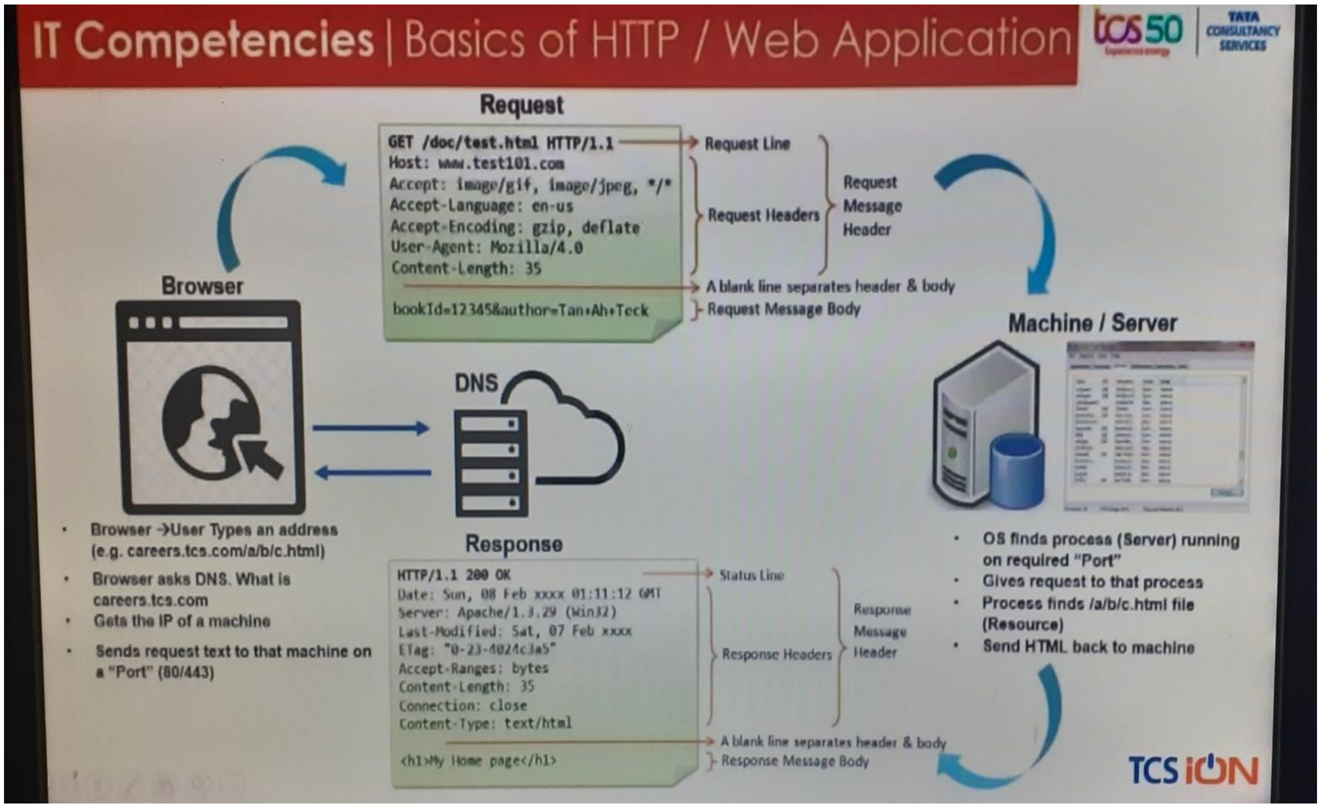
3)Delight factors

- Basic algorithms
- Design patterns
- Web application flow
- One digital skill

4)Pointers

- Spend time on your final project
- Communicate your technical strengthsupfront
- Accept what you don’t knowgive logicalpaths to get to the solution
- Communicate efficiently

Detailed explanation of IT competencies



IT Competencies | Basics of HTTP / Web Application

Request

```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

- Browser →User Types an address (e.g. careers.tcs.com/a/b/c.html)
- Browser asks DNS. What is careers.tcs.com
- Gets the IP of a machine
- Sends request text to that machine on a "Port" (80/443)

- OS finds process (Server) running on required "Port"
- Gives request to that process
- Process finds /a/b/c.html file (Resource)
- Send HTML back to machine

Data warehousing

- Student management system
- Library data
- Student data
- Payroll data
- Store data

Understand artificial intelligence

Part 1

Objectives

-Understand the role of basic

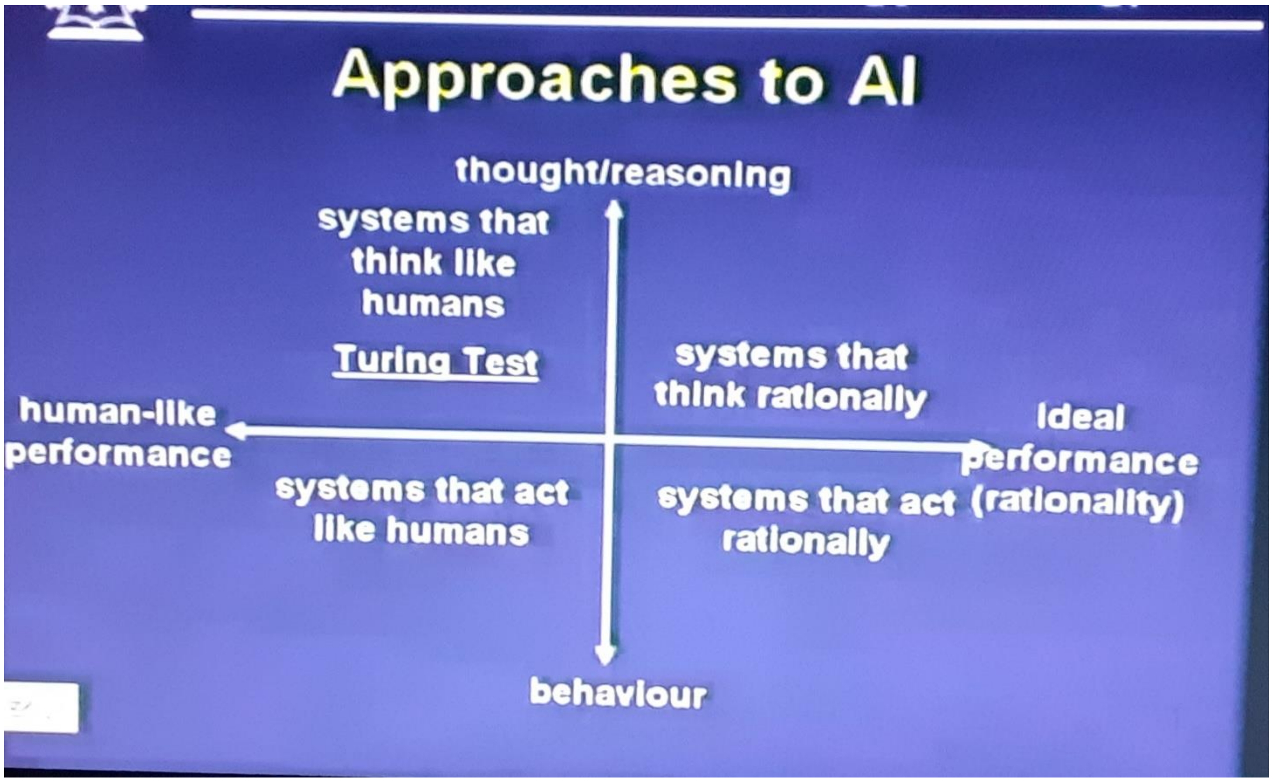
- Knowledge presentation
- Problem solving,and
- Learning methods of AI

In engineering intelligent systems

What is AI ?

It is concerned with with the design of intelligence in an artificial device term coined by McCarthy in 1956.

Approaches to AI


Approaches to AI
thought/reasoning
systems that think like humans
Turing Test
systems that think rationally
human-like performance
Ideal performance (rationality)
systems that act like humans
systems that act rationally
behaviour

The turning test : result

If theinterrogator cannot reliablydistinguish the human from the computer .Then the computer does possess intelligence.

Typical AI problems

- ➢ Intelligent entities need to be able to do both “mundane” and “expert”tasks:
- ➢ Mundane tasks:
- Planning route ,activity.
- Recognizing people,objects.
- Communicating
- Navigating round obstacles on the street
- ➢ Expert tasks :
- Medical diagnosis
- Mathematical problem solving.

Whats easy and whats hard ?

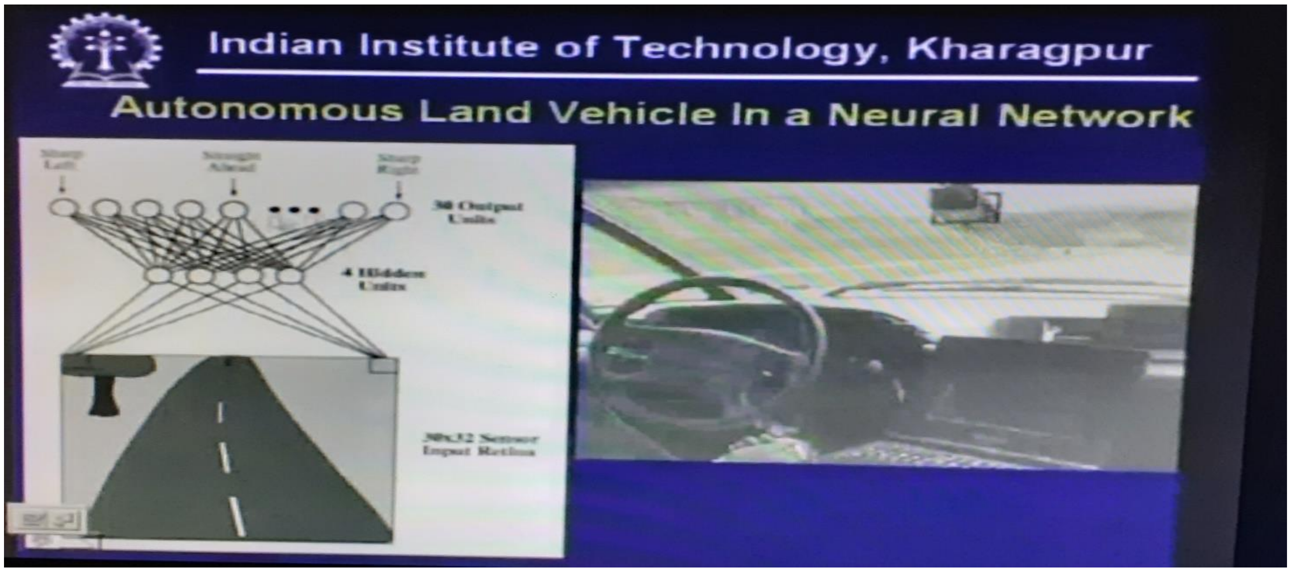
It has been easier to mechanize many of the high level tasks we usually associate with “intelligence”in people

Practical impact of AI

- AI compomenets are embedded in numerous devices e.g. copy machines.
- AI systems are in everyday use
- Detecting credit card fraud
- Configuring products
- Aiding complex planning tasks
- Advicing physicians
- Intelligent tutoring systems provide students with personalized attention.

Applications

- Computer vision
- Image recognition
- Robotics
- Language processing


Indian Institute of Technology, Kharagpur
Autonomous Land Vehicle In a Neural Network
Sharp Left
Straight Ahead
Sharp Right
30 Output Units
4 Hidden Units
30x32 Sensor Input Retina

Machine translation

- Immediate translations between people speaking different languages would be a remarkable achievement of enormous economic and cultural benefit.
- Universal translation is one of 10 emerging technologies that will affect our lives and work ‘inrevolutionary ways’ within a decade ,technology review says.

What can AI systems do

- Computer vision : face recognition.
- Robotics :autonomous automobile.
- Natural language processing : simple machine translation.

PART 2

INTELLIGENT AGENTS

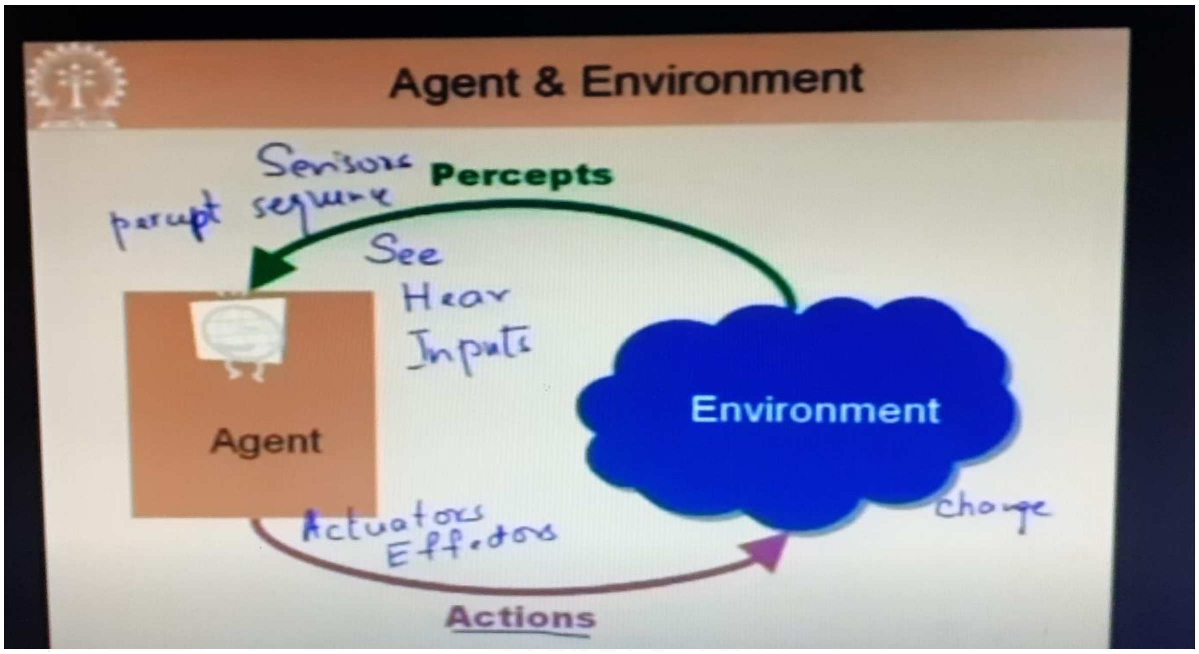
- Define an agent.
- Define an intelligent agent.
- Define a rational agent.
- Explain bounded rationality.
- Discuss different types of environment.
- Explain different agent architectures.

Instructional objectives

- Understand what an agent is and how an agent interacts with the environment .
- Given a problem statement the student should be able to
- Identify the percepts available to the agent and
- The actions that the agent can execute.
- Understand the performance measures used to evaluate an agent.

Agents

- Operate in environment.
- Perceives its environment through sensors.
- Acts upon its environment through actuators/effectors.
- Have goals.



Examples of agents:

- Humans.
- Robots.

Types of agents

- Softbots.
- Expertsystems.
- Autonomous spacecraft.
- Intelligenty buildings.

Fundamental faculties of intelligence

- Actinbg sensing.
- Understanding ,reasoning ,learning.
- Robotics.

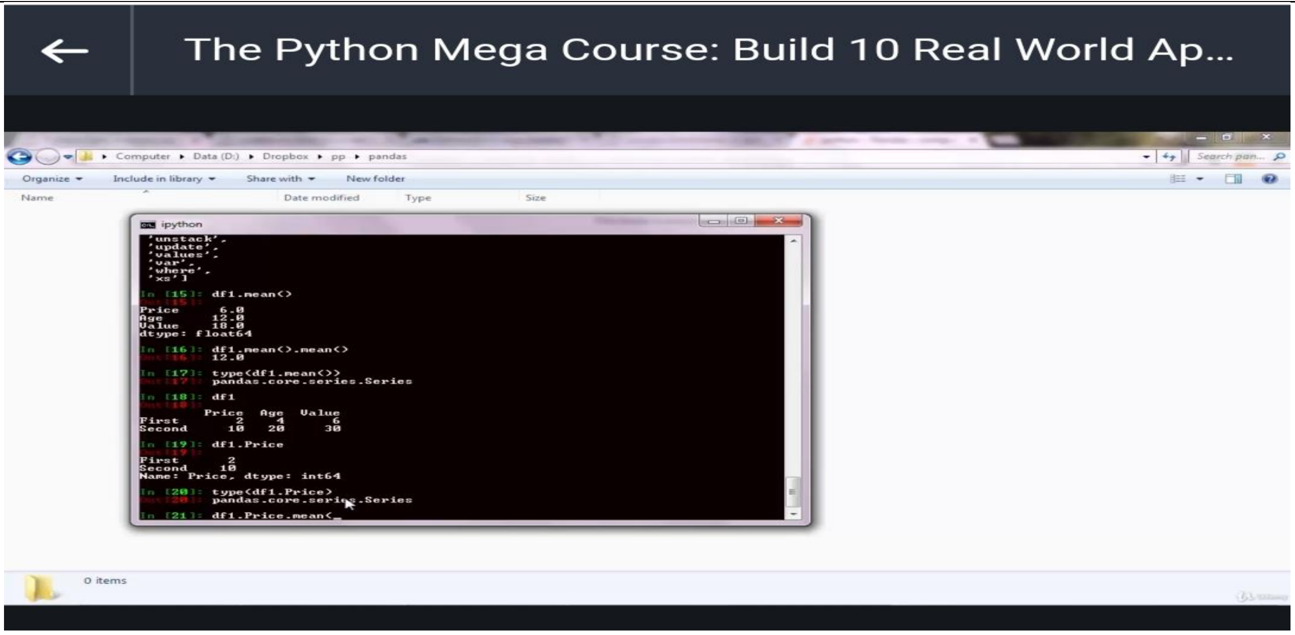
- Sensing needs understanding to be useful**.**

Rationality

- Perfect rationality.
- Bounded rationality.
- Rational action.
- Omniscience.
- Agent environment.
- Environment :observability.
- Environment :determinism.
- Episodicity .
- Dynamism .
- Table based agent.
- Subsumption architecture.
- State-based agent.
- Goal based agent.
- Utility based agent.

AFTERNOON SESSION DETAILS

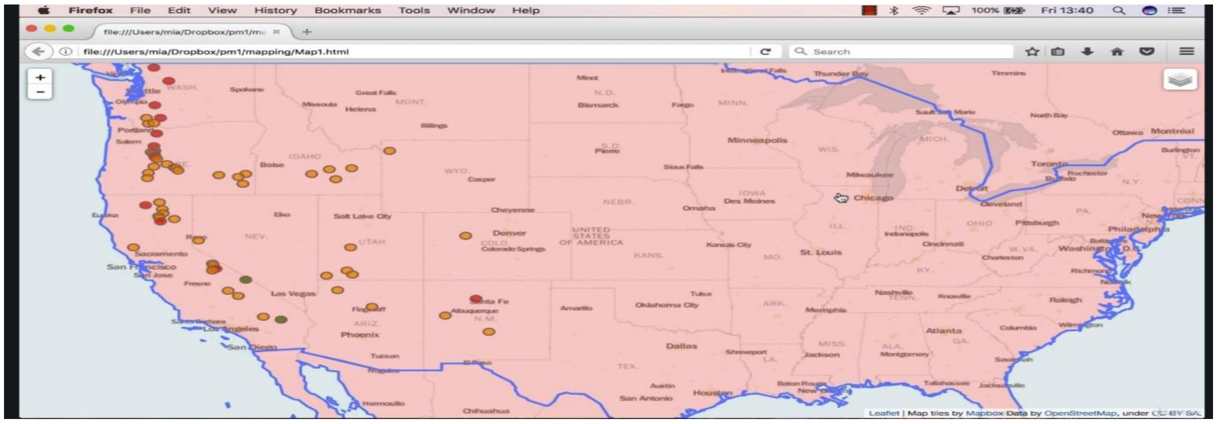
Date:	**22-5-2020**	**Name:**	**Kavyashree m**
Course:	**Python programming**	**USN:**	**4al15ec036**
Topic:	**Data analysis with Pandas ,create webmaps with python and folium**	**Semester & Section:**	**8th A**
Github Repository:	**kavya**		



The Python Mega Course: Build 10 Real World Ap...
Computer ▸ Data (D:) ▸ Dropbox ▸ pp ▸ pandas
Organize
Include in library
Share with
New folder
Name
Date modified
Type
Size
ipython
0 items

```
 'unstack',
 'update',
 'values',
 'var',
 'where',
 'xs']

In [15]: df1.mean()
Out[15]:
Price     6.0
Age      12.0
Value    18.0
dtype: float64

In [16]: df1.mean().mean()
Out[16]: 12.0

In [17]: type(df1.mean())
Out[17]: pandas.core.series.Series

In [18]: df1
Out[18]:
        Price  Age  Value
First       2    4      6
Second     10   20     30

In [19]: df1.Price
Out[19]:
First      2
Second    10
Name: Price, dtype: int64

In [20]: type(df1.Price)
Out[20]: pandas.core.series.Series

In [21]: df1.Price.mean(_
```

Fig 1: Data analysis with Pandas



Firefox File Edit View History Bookmarks Tools Window Help
file:///Users/mia/Dropbox/pm1/mapping/Map1.html
Search

Fig 2: create webmaps with python and folium

Pandas

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python.

Pandas.DataFrame

Syntax: “

Class pandas.DataFrame(data=None, index:Optional[Collection]=None, columns:Optional[Collection] =None, dtype: Union[str, numpy.dtype, ExtensionDtype, None] = None, copy: bool = False).We can analyze panda by :

1)SERIES

2) DATAFRAME

Series:

Series is one dimensional(1-D) array defined in pandas that can be used to store any data type.

# Program to Create series with scalar values

Data =[1, 3, 4, 5, 6, 2, 9] # Numeric data

# Creating series with default index values

s = pd.Series(Data)

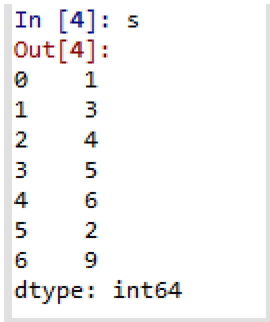
# predefined index values

Index =['a', 'b', 'c', 'd', 'e', 'f', 'g']

# Creating series with predefined index values

si = pd.Series(Data, Index)

Output:



```
In [4]: s
Out[4]:
0    1
1    3
2    4
3    5
4    6
5    2
6    9
dtype: int64
```

Panda is a Python library that provides extensive means for data analysis. Data scientists often work with data stored in table formats like .csv, .tsv, or .xlsx. Pandas makes it very convenient to load, process, and analyze such tabular data using SQL-like queries. In conjunction with Matplotlib and Seaborn, Pandas provides a wide range of opportunities for visual analysis of tabular data.

The main data structures in Pandas are implemented with Series and DataFrame classes. The former is a one-dimensional indexed array of some fixed data type. The latter is a two-dimensional data structure - a table - where each column contains data of the same type. You can see it as a dictionary of Series instances. DataFrames are great for representing real data: rows correspond to instances (examples, observations, etc.), and columns correspond to features of these instances.

In [1]:

```
import numpy as np
import pandas as pd
pd.set_option("display.precision", 2)
```

Understand the basic Pandas data structures

Pandas has two core data structures used to store data: The *Series* and the *DataFrame*.

Series

The series is a one-dimensional array-like structure designed to hold a single array (or 'column') of data and an associated array of data labels, called an *index*. We can create a series to experiment with by simply passing a list of data, let's use numbers in this example:

Copy contents

```
import pandas as pd


my_series = pd.Series([4.6, 2.1, -4.0, 3.0])
print(my_series)
```

The output should be:

Copy contents

```
0    4.6
1    2.1
2   -4.0
3    3.0
dtype: float64
```

Note that printing out our *Series* object prints out the values and the index numbers. If we just wanted the values, we can add to our script the following line:

Copy contents

```
print(my_series.values)
```

Which in addition will print:

Copy contents

```
array([ 4.6,  2.1, -4. ,  3. ])
```

For a lot of applications, a plain old *Series* is probably not a lot of use, but it is the core component of the Pandas workhorse, the *DataFrame*, so it's useful to know about.

DataFrames

The DataFrame represents tabular data, a bit like a spreadsheet. DataFrames are organised into colums (each of which is a *Series*), and each column can store a single data-type, such as floating point numbers, strings, boolean values etc. DataFrames can be indexed by either their row or column names. (They are similar in many ways to R's data.frame.)

We can create a DataFrame in Pandas from a Python dictionary, or by loading in a text file containing tabular data. First we are going to look at how to create one from a dictionary.

Setup

Let's create a pandas DataFrame with 5 columns and 1000 rows:

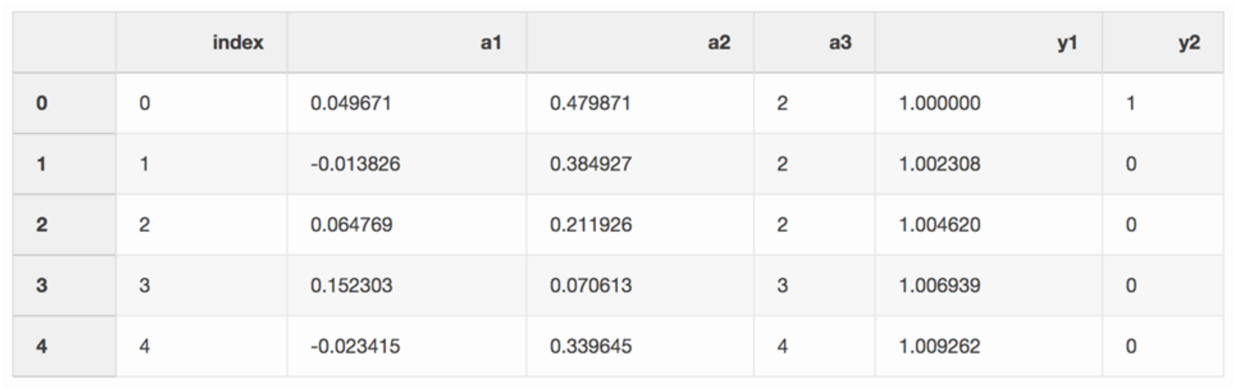
- a1 and a2 have random samples drawn from a normal (Gaussian) distribution,

- a3 has randomly distributed integers from a set of (0, 1, 2, 3, 4),
- y1 has numbers spaced evenly on a log scale from 0 to 1,
- y2 has randomly distributed integers from a set of (0, 1).

```
mu1,sigma1=0,0.1
mu2,sigma2=0.2,0.2
n=1000df=pd.DataFrame(
    {
        "a1":pd.np.random.normal(mu1,sigma1,n),
        "a2":pd.np.random.normal(mu2,sigma2,n),
        "a3":pd.np.random.randint(0,5,n),
        "y1":pd.np.logspace(0,1,num=n),
        "y2":pd.np.random.randint(0,2,n),
    }
)
```

Readers with Machine Learning background will recognize the notation where a1, a2 and a represent attributes and y1 and y2 represent target variables. In short, Machine Learning algorithms try to find patterns in the attributes and use them to predict the unseen target variable — but this is not the main focus of this blog post. The reason that we have two target variables (y1 and y2) in the DataFrame (one binary and one continuous) is to make examples easier to follow.

We reset the index, which adds the index column to the DataFrame to enumerates the rows.
df.reset_index(inplace=True)



	index	a1	a2	a3	y1	y2
0	0	0.049671	0.479871	2	1.000000	1
1	1	-0.013826	0.384927	2	1.002308	0
2	2	0.064769	0.211926	2	1.004620	0
3	3	0.152303	0.070613	3	1.006939	0
4	4	-0.023415	0.339645	4	1.009262	0

Creating Web Maps in Python Using Folium

Folium

It is a Python package built to bridge the data wrangling muscle of Python with Leaflet's easy-to-use JavaScript library for creating attractive, interactive web maps. The open source Leaflet is a highly popular web mapping tool due to its flexibility, with a healthy number of community-developed plug-ins further expanding its native capabilities. While Python is a robust programming language, with many packages contributing to geospatial analysis– Pandas, GeoPandas, Fiona, Shapely, Matplotlib, and Descartes to name a few– Folium differentiates itself through ease of use and the interactive potential of the final product. After some experimentation with the library, it did not take very long to produce a functional, albeit simple, web map with clustered point data, accompanied by popup windows. However, it was obvious that there is more to explore with Folium, as it plays well with many types of geospatial data, includes built-in functions and methods for producing choropleths, temporal visualizations, and allows for the marriage of the best of Python and Leaflet.

The code and resulting maps show a straightforward exercise in extracting the geographic coordinates (already matching Leaflet's default web-mercator projection) and a few attribute values corresponding to warehouse/distribution centers in Pennsylvania's Lehigh Valley from an excel spreadsheet. The Pandas library was used to read the excel document and convert the desired information to a dataframe. Folium was used to initialize a Leaflet map, add records as points with some stylization applied. This is brief code that could easily be added at the end of a more intensive spatial analysis using Python. It can provide a quick way to publish results in an interactive format without necessitating the use of JavaScript/html/CSS, or could serve as a jump start on more elaborate styling.

Map #1 – Mostly default styling

#converting the imported data to a pandas dataframe object df = pd.read_excel("Warehouses.xlsx")

#identifying the long, lat columns,andb\other properties for the popup window df.head()

pdlat='Latitude'

pdlon='Longitude'

#removingunwanted columns avail =df[3:]

#initalizing folium map object as m and using the geographic mean of the data points to center the viewpoint;base map defaults to OSM m=folium.Map(location=[df[pdlat].mean(), df[pdlon].mean()], zoom_start=12)

#Iterate through edited dataframe to extract coordinates and property name for each record forrowinavail.iterrows():

prop=str(row[1]['Property'])

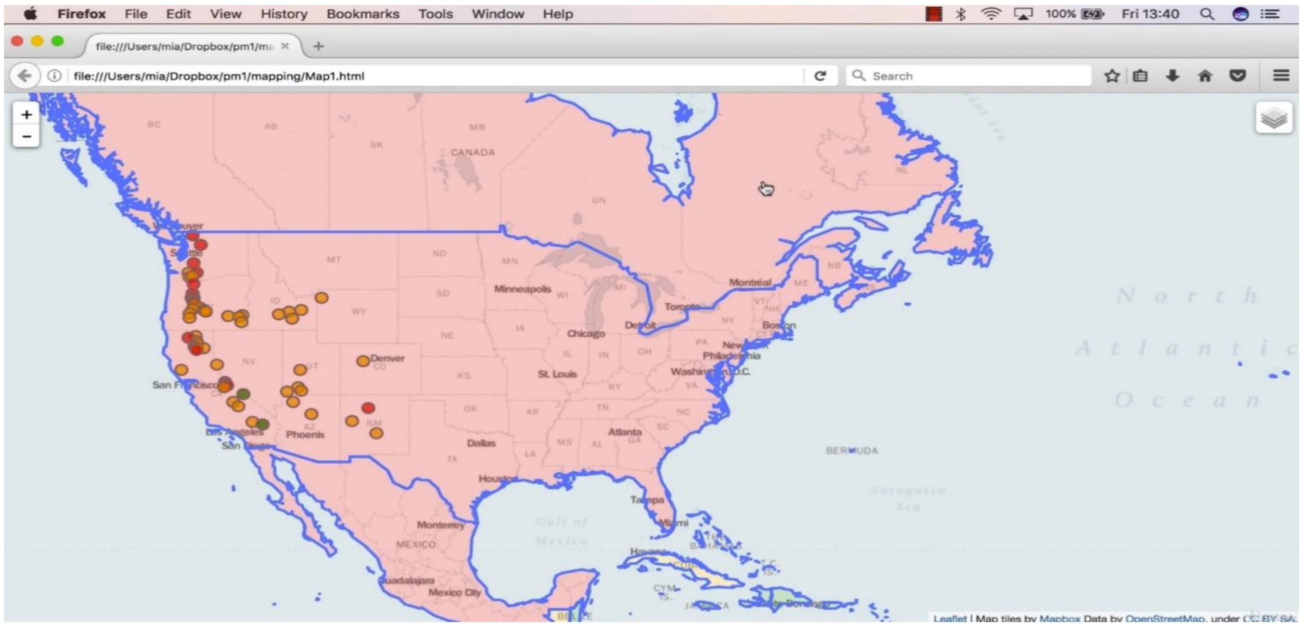
lat=row[1][pdlat]

lon = row[1][pdlon]

#attach each record to a default marker style based on coordinates and add property name to popup window

m.simple_marker(location=[lat, lon], marker_color='red', popup=prop)

#outputting html document with code for an interactive map to working directory m.create_map('map.html')



Map #2 – Custom MapBox Tiles, Clustered Markers, Different Icon Imagery

Only showing code that differs.

#Generate map using custom Mapbox tiles

m=folium.Map(location=[df[pdlat].mean(),

df[pdlon].mean()],zoom_start=9,

tiles='https://api.mapbox.com/styles/v1/username/yourstyle/tiles/256/{z}/{x}/{y}?

access_token=pk.yourtokenhere',attr='My data attribution')

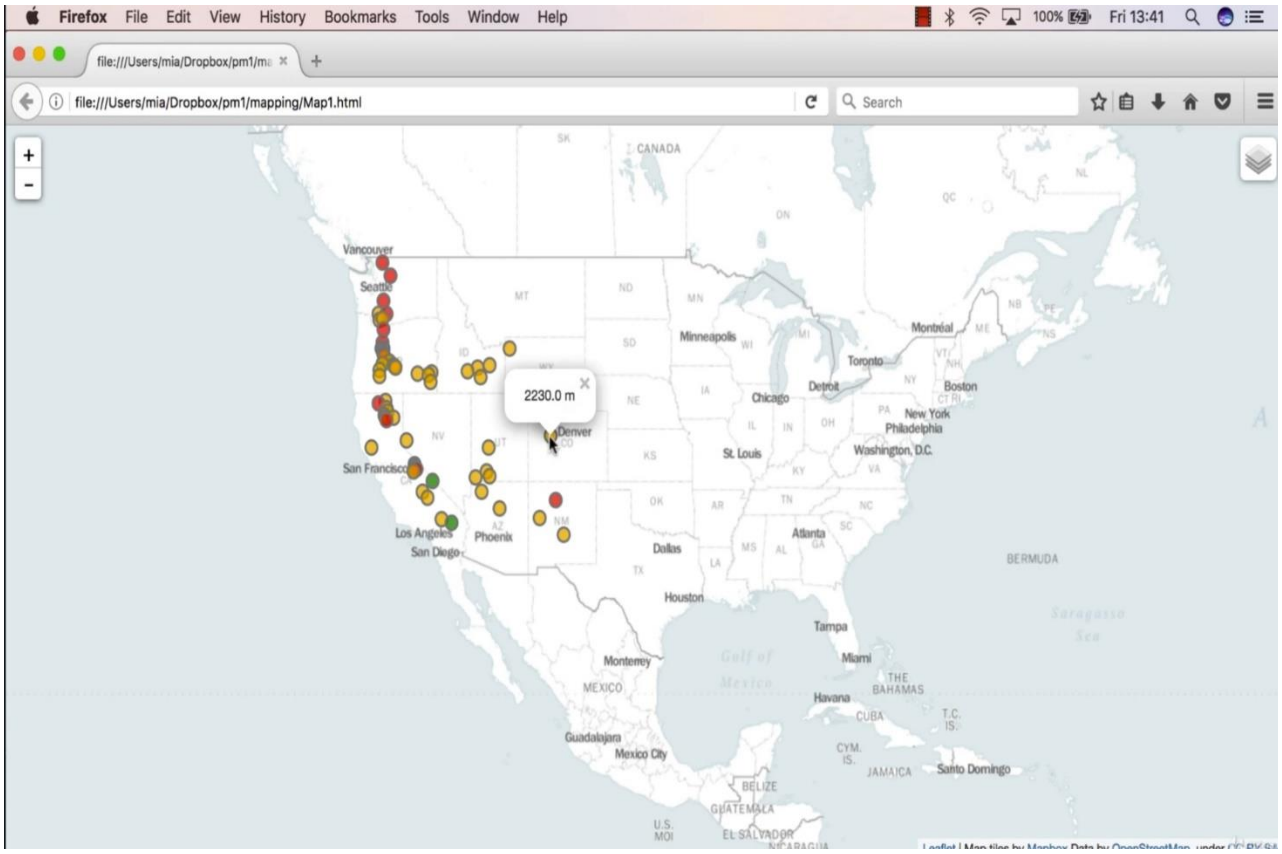
#Iterate through edited dataframe to extract coordinates and property name for each record for row in df.iterrows():

prop=str(row[1][‘Property’])

lat=row[1][pdlat]

lon=row[1][pdlon]

#used the marker_icon argument to select from natively supported bootstrap supported icons and added clustering affect to markers m.simple_marker(location=[lat, lon], marker_color='red', marker_icon='flag', clustered_marker=True, popup=prop)


Firefox File Edit View History Bookmarks Tools Window Help
Fri 13:41
file:///Users/mia/Dropbox/pm1/mapping/Map1.html
Search
2230.0 m
Vancouver
Seattle
San Francisco
Los Angeles
San Diego
Phoenix
Denver
Minneapolis
Chicago
St. Louis
Dallas
Houston
Toronto
Detroit
Montréal
Boston
New York
Philadelphia
Washington, D.C.
Atlanta
Tampa
Miami
Havana
Monterrey
Guadalajara
Mexico City
CANADA
MEXICO
BERMUDA
CUBA
THE BAHAMAS
Santo Domingo