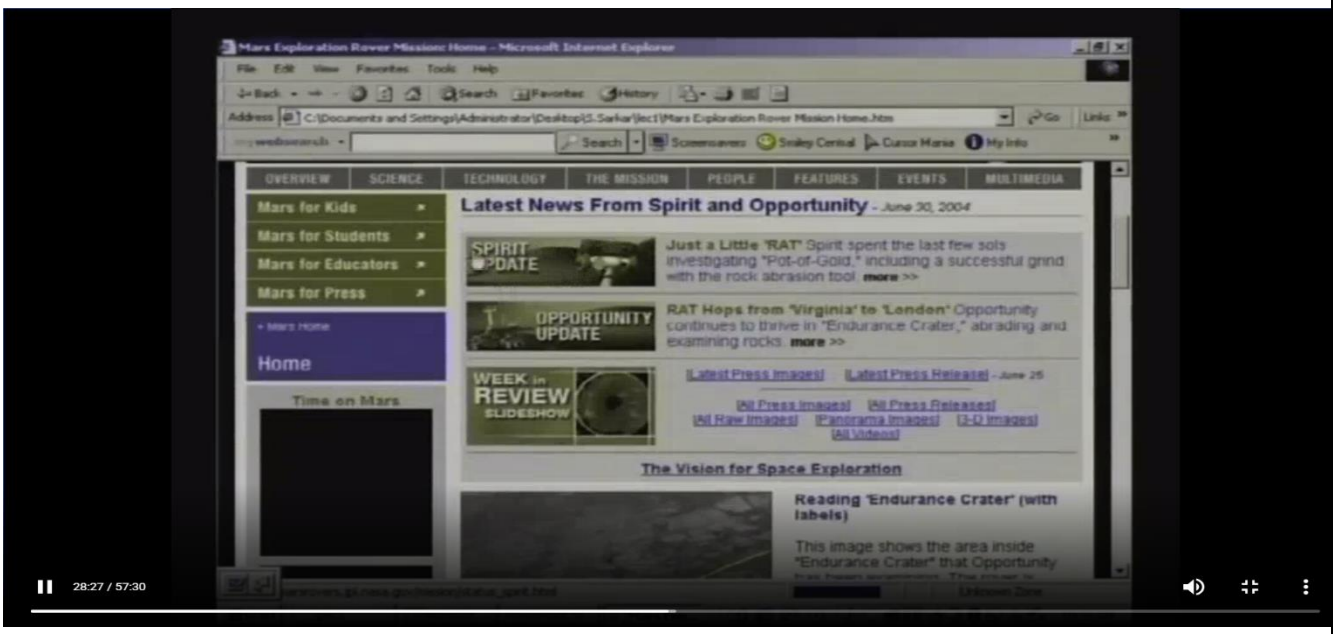
DAILY ASSESSMENT FORMAT

Date:	22/05/2020	Name:	Shilpa N
Course:	TCS ion	USN:	4AL16EC071
Topic:	Understand Artificial Intelligence (AI) - Part 1 Understand Artificial Intelligence (AI) - Part 2	Semester & Section:	8 “B”
Github Repository:	Shilpan-test		

FORENOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

UNDERSTAND ARTIFICIAL INTELLIGENCE (AI) - PART 1

The session aimed to introduce artificial intelligence. The goal of the session was to understand the role of basic knowledge representations, problems solving and learning methods in artificial intelligence. On taking the course one should be able to access the application, strengths and weaknesses of these

methods in solving particular engineering problems. Develop intelligence systems by assembling solutions to concrete computational problems appreciable the role of problem solving, natural language processing and vision in understanding human intelligence from a computational perspective. The objective of the session is to understand the definition of artificial intelligence, discuss the different faculties involved with intelligence behaviour, examine the different ways of approaching artificial intelligence, look at some examples system that use artificial intelligence and trace brief history of artificial intelligence.

UNDERSTAND ARTIFICIAL INTELLIGENCE (AI) - PART 2

In the second part of the artificial intelligence the session started with intelligent agent. The objective of the chapter is to define an agent, intelligent agent. On completing the chapter, we are able to understand what an agent is and how an agent interacts with the environment. Understand the performance measure used to evaluate an agent. On completion of the lesson one will be familiar with different agent architecture, stimulus, response agents, state-based agents, deliberate or goal directed agents and utility-based agent. The agents should operate in an environment, perceives its environment through sensors, acts upon its environment through actuators and sensors. Behaviour and performance of the agent interns of agent functions. Performance measure is a subjective measure to characterize how successful an agent is in terms of money, accuracy etc.

Date:	22/05/2020	Name:	Shilpa N
Course:	PYTHON	USN:	4AL16EC071
Topic:	Application 2: create web maps with python and folium	Semester & Section:	8 “B”

AFTERNOON SESSION DETAILS

Python Data. Leaflet.js Maps.

Folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Manipulate your data in Python, then visualize it in on a Leaflet map via Folium.

Concept

Folium makes it easy to visualize data that’s been manipulated in Python on an interactive Leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing Vincent/Vega visualizations as markers on the map.

The library has a number of built-in tilesets from OpenStreetMap, MapQuest Open, MapQuest Open Aerial, Mapbox, and Stamen, and supports custom tilesets with Mapbox or Cloudmade API keys. Folium supports both GeoJSON and TopoJSON overlays, as well as the binding of data to those overlays to create choropleth maps with color-brewer color schemes.

Installation

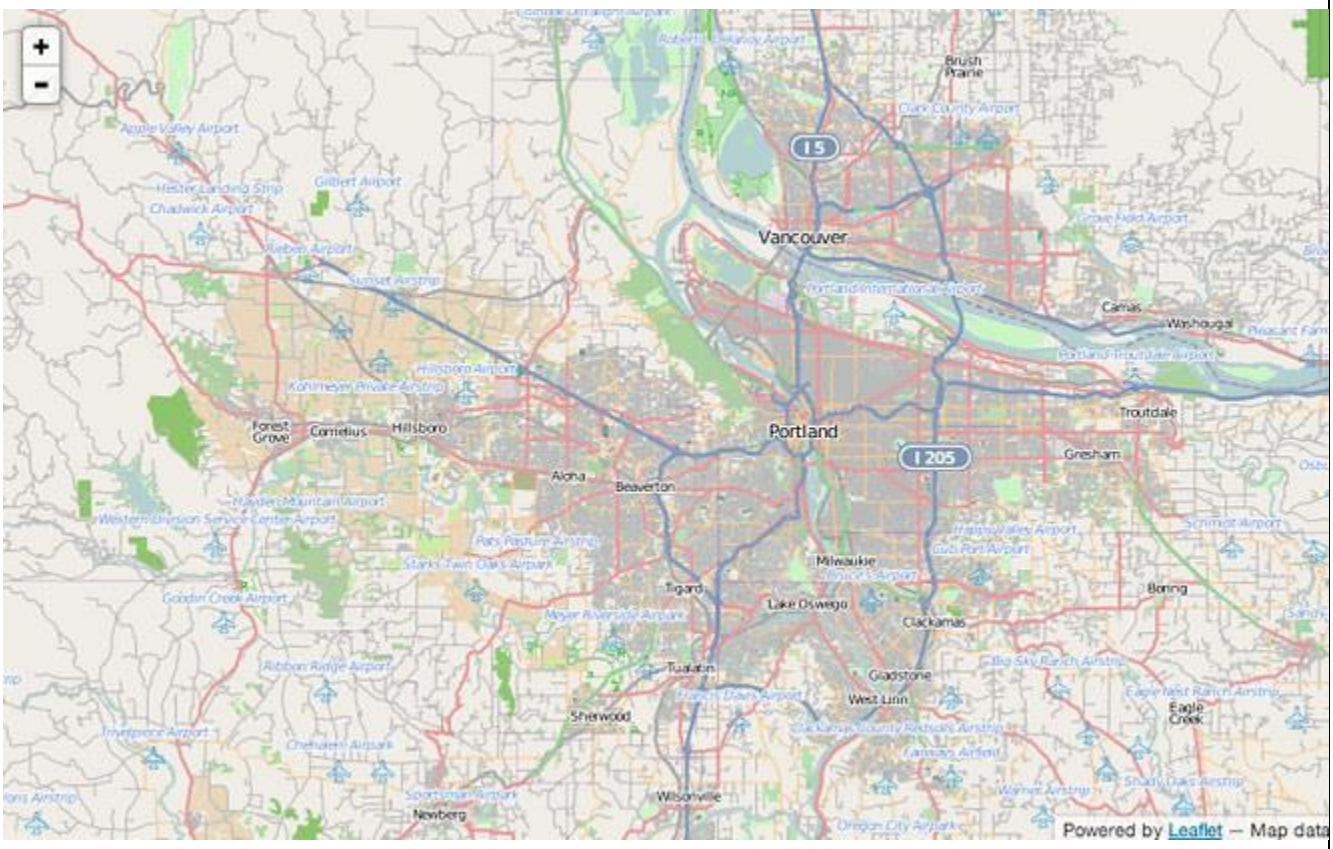
```
$ pip install folium
```

To create a base map, simply pass your starting coordinates to Folium:

```
import folium
```

```
map_osm = folium.Map(location=[45.5236, -122.6750])

map_osm.create_map(path='osm.html')
```

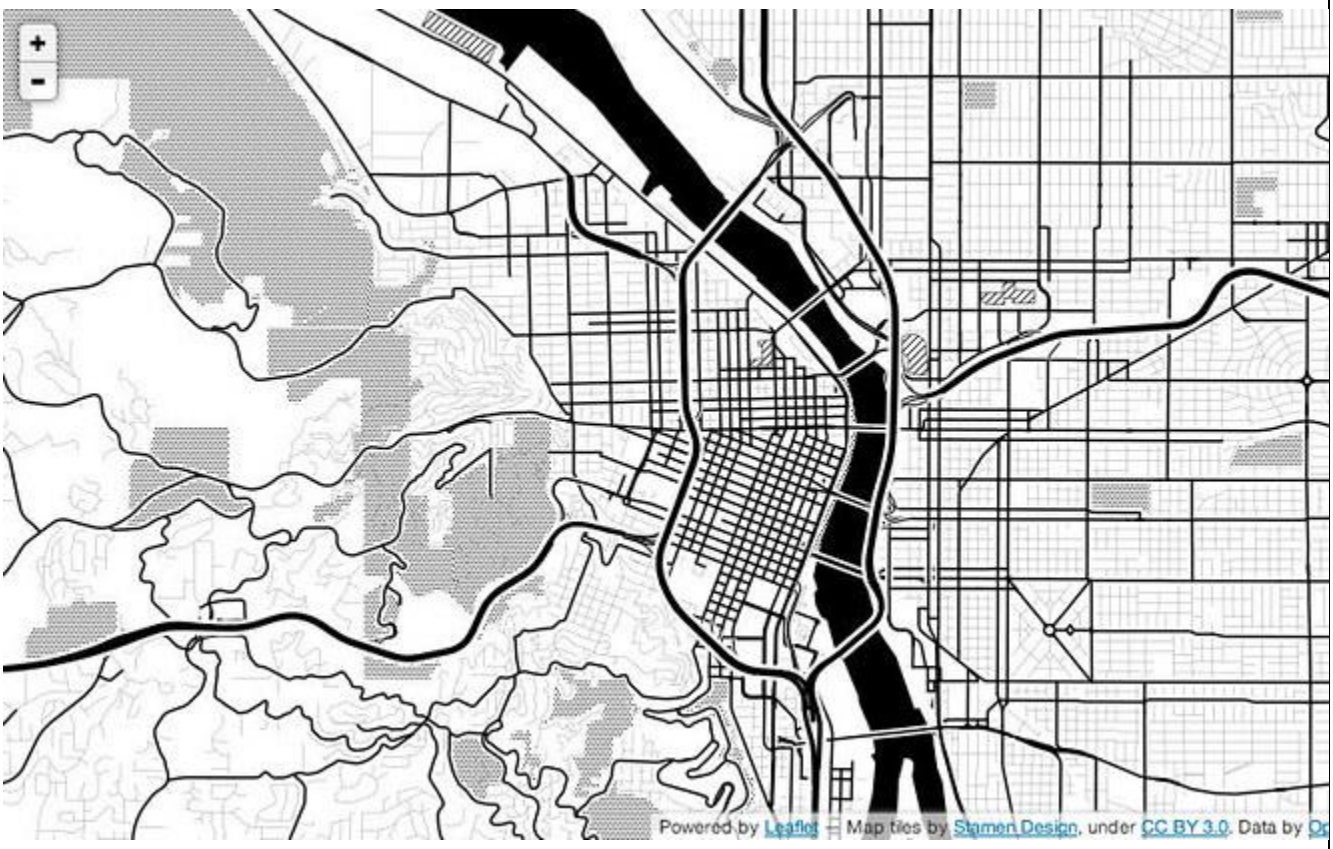


Folium defaults to OpenStreetMap tiles, but Stamen Terrain, Stamen Toner, Mapbox Bright, and Mapbox Control room tiles are built in:

```
stamen = folium.Map(location=[45.5236, -122.6750], tiles='Stamen Toner',

                    zoom_start=13)

stamen.create_map(path='stamen_toner.html')
```



Folium also supports Cloudmade and Mapbox custom tilesets- simply pass your key to the API_key keyword:

```
custom = folium.Map(location=[45.5236, -122.6750], tiles='Mapbox',
                    API_key='wrobstory.map-12345678')
```

Lastly, Folium supports passing any Leaflet.js compatible custom tileset:

```
tileset = r'http://{s}.tiles.yourtiles.com/{z}/{x}/{y}.png'

map = folium.Map(location=[45.372, -121.6972], zoom_start=12,
                 tiles=tileset, attr='My Data Attribution')
```

Markers

Folium supports the plotting of numerous marker types, starting with a simple Leaflet style location marker with popup text:

```
map_1 = folium.Map(location=[45.372, -121.6972], zoom_start=12,
                   tiles='Stamen Terrain')
map_1.simple_marker([45.3288, -121.6625], popup='Mt. Hood Meadows')
map_1.simple_marker([45.3311, -121.7113], popup='Timberline Lodge')
map_1.create_map(path='mthood.html')
```

Live example

Folium supports colors and marker icon types (from bootstrap)

```
map_1 = folium.Map(location=[45.372, -121.6972], zoom_start=12,tiles='Stamen Terrain')

map_1.simple_marker([45.3288, -121.6625], popup='Mt. Hood Meadows',marker_icon='cloud')

map_1.simple_marker([45.3311, -121.7113], popup='Timberline Lodge',marker_color='green')

map_1.simple_marker([45.3300, -121.6823], popup='Some Other Location',marker_color='red',marker_icon='info-sign')

map_1.create_map(path='iconTest.html')
```

Folium also supports circle-style markers, with custom size and color:

```
map_2 = folium.Map(location=[45.5236, -122.6750], tiles='Stamen Toner',

                   zoom_start=13)

map_2.simple_marker(location=[45.5244, -122.6699], popup='The Waterfront')

map_2.circle_marker(location=[45.5215, -122.6261], radius=500,

                    popup='Laurelhurst Park', line_color='#3186cc',

                    fill_color='#3186cc')

map_2.create_map(path='portland.html')
```