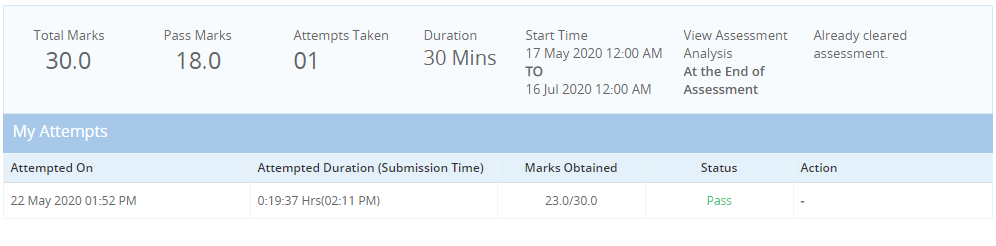
***Day 5 report***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Date:*** | ***22-May-2020*** | ***Name:*** | ***Srinidhi J C*** |
| ***Course:*** | ***TCS iON*** | ***USN:*** | ***4al16ec078*** |
| ***Topic:*** | ***1.* Understand artificial intelligence (AI). Part 1**  **2. Understand artificial intelligence (AI). Part 2**  **3. Final assessment, feedback, certificate issued.** | ***Semester & Section:*** | ***8th sem & B*** |
| ***Github***  ***Repository:*** | **https://github.com/alvas-education-foundation/SrinidhiJC078.git** |  |  |

A screenshot of a cell phone

Description automatically generated

***Image of session***

**FINAL ASSESSMENT**:

***MORNING SESSION DETAILS***

**Report of the course:**

**UNDERSTAND ARTIFICIAL INTELLIGENCE (AI) PART 1**

* Introduction
* Definition of AI
* Approaches to AI
* The turning test
* Typical AI problems
* Intelligent behavior
* Autonomous land vehicle in a neural network
* Machine translation
* Autonomous agent
* Mars rover
* Internet agent
* Limits of AI today
* What can AI system do?
* AI history
* AI foundations

1. Psychology
2. Philosophy
3. Mathematics
4. Economics
5. Linguistic
6. Computer engineering
7. Biology
8. Questions discussed

**UNDERSTAND ARTIFICIAL INTELLIGENCE (AI) PART 2**

* Introduction to agent
* Agents and environment
* Sensors and effectors
* Performance of AI
* Example of agent

1. Human
2. Robot
3. Software agents

* Rationality

1. Perfect rationality
2. Bounded rationality
3. Rational action

* Omniscience

1. A rational agent is not an omniscient’s
2. Rationality must taken into limitation agent

* Agent environment

1. Fully observable
2. Partially observable

* Environment episodicity
* Complex environments

1. Knowledge rich
2. Input rich

* Test based agent
* Table based agent
* Percept based agent
* Subsumption architecture
* State based agent
* Goal- based agent
* Utility based agent
* Questions discussed

|  |  |  |  |
| --- | --- | --- | --- |
| ***Date:*** | ***22-5-2020*** | ***Name:*** | ***Srinidhi J C*** |
| ***Course:*** | ***Python programming*** | ***USN:*** | ***4AL16EC078*** |
| ***Topic:*** | ***Application 2 Create Webmaps with Python and Folium*** | ***Semester & Section:*** | ***8th B*** |
| ***Github Repository:*** | **https://github.com/alvas-education-foundation/SrinidhiJC078.git** |  |  |

***Image of session***

A close up of a map

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a map

Description automatically generated

**Report –**

**Today’s lesion is about to build an webpage to view world maps and to create markers on it.**

* In today’s section I have learnt these many programs:

tiles = "Mapbox Bright"

* Please use this instead:

tiles = "Stamen Terrain"

* In the popup window use HTML

import folium

import pandas

data = pandas.read\_csv("Volcanoes.txt")

lat = list(data["LAT"])

lon = list(data["LON"])

elev = list(data["ELEV"])

html = """<h4>Volcano information:</h4>

Height: %s m

"""

map = folium.Map(location=[38.58, -99.09], zoom\_start=5, tiles="Mapbox Bright")

fg = folium.FeatureGroup(name = "My Map")

for lt, ln, el in zip(lat, lon, elev):

iframe = folium.IFrame(html=html % str(el), width=200, height=100)

fg.add\_child(folium.Marker(location=[lt, ln], popup=folium.Popup(iframe), icon = folium.Icon(color = "green")))

map.add\_child(fg)

map.save("Map\_html\_popup\_simple.html")

You can even put links in the popup window. For example, the code below will produce a popup window with the name of the volcano as a link which does a Google search for that particular volcano when clicked:

import folium

import pandas

data = pandas.read\_csv("Volcanoes.txt")

lat = list(data["LAT"])

lon = list(data["LON"])

elev = list(data["ELEV"])

name = list(data["NAME"])

html = """

Volcano name:<br>

<a href="https://www.google.com/search?q=%%22%s%%22" target="\_blank">%s</a><br>

Height: %s m

"""

map = folium.Map(location=[38.58, -99.09], zoom\_start=5, tiles="Mapbox Bright")

fg = folium.FeatureGroup(name = "My Map")

for lt, ln, el, name in zip(lat, lon, elev, name):

iframe = folium.IFrame(html=html % (name, name, el), width=200, height=100)

fg.add\_child(folium.Marker(location=[lt, ln], popup=folium.Popup(iframe), icon = folium.Icon(color = "green")))

map.add\_child(fg)

map.save("Map\_html\_popup\_advanced.html")

You can use dir(folium)  to look for possible methods of creating circle markers. Among the methods you will see Marker, which we previously used.

Once you locate the method consider using the help  function to look for possible arguments you can pass to the method for styling the circle markers.

*import folium*

*import pandas*

*data = pandas.read\_csv("Volcanoes.txt")*

*lat = list(data["LAT"])*

*lon = list(data["LON"])*

*elev = list(data["ELEV"])*

*def color\_producer(elevation):*

*if elevation < 1000:*

*return 'green'*

*elif 1000 <= elevation < 3000:*

*return 'orange'*

*else:*

*return 'red'*

*map = folium.Map(location=[38.58, -99.09], zoom\_start=6, tiles="Mapbox Bright")*

*fgv = folium.FeatureGroup(name="Volcanoes")*

*for lt, ln, el in zip(lat, lon, elev):*

*fgv.add\_child(folium.CircleMarker(location=[lt, ln], radius = 6, popup=str(el)+" m",*

*fill\_color=color\_producer(el), fill=True, color = 'grey', fill\_opacity=0.7))*

*fgp = folium.FeatureGroup(name="Population")*

*fgp.add\_child(folium.GeoJson(data=open('world.json', 'r', encoding='utf-8-sig').read(),*

*style\_function=lambda x: {'fillColor':'green' if x['properties']['POP2005'] < 10000000*

*else 'orange' if 10000000 <= x['properties']['POP2005'] < 20000000 else 'red'}))*

*map.add\_child(fgv)*

*map.add\_child(fgp)*

*map.add\_child(folium.LayerControl())*

*map.save("Map1.html")*