USN | | | | | | | | | | |

Internal Assessment Test 3 – December 2024

Sub:	**Data Visualization**					Sub Code:	**21AD71**	Branch:	**AInDS**
Date:	**14/12/2024**	Duration	**90 minutes**	Max Marks:	**50**	Sem	**VII**		**OBE**

		Answer any FIVE Questions	**MARKS**	**CO**	**RBT**
1	a	What is bokeh? Explain the features of bokeh. Explain the concept of bokeh with a diagram.	[5] [5]	5	L1
2	a	What is web scraping? How are binary files read using urllib?	[10]	5	L2
3	a	How is HTML parsing done using BeautifulSoup?	[10]	5	L2
4	a	Explain widgets in bokeh. Give the syntax for the following: Import and call the output_notebook method from Bokeh's io interface to display the plots inside Jupyter Notebook. • Import the necessary interact and interact_manual elements from ipywidgets. • Create a checkbox widget and print out the result of the interactive element. • Create a dropdown using a list of options, ['Option1','Option2','Option3', 'Option4'] as the @interact decorator value. • Create a text input using a value of 'Input Text' as the @interact decorator value.	[10]	5	L3
5	a	What is eXtensible Markup Language? Explain looping through nodes?	[10]	5	L2
6	a	Explain JSON and how is parsing done in JSON?	[10]	5	L2

CI **CCI** **HOD**

Answer Key

1.a

Introduction

Bokeh is an interactive visualization library focused on modern browsers and the web. Other than Matplotlib or geoplotlib, the plots and visualizations we are going to create in this chapter will be based on JavaScript widgets. Bokeh allows us to create visually appealing plots and graphs nearly out of the box without much styling. In addition to that, it helps us construct performant interactive dashboards based on large static datasets or even streaming data.

Bokeh has been around since 2013, with *version* 1.4.0 being released in November 2019. It targets modern web browsers to present interactive visualizations to users rather than static images. The following are some of the features of Bokeh:

- **Simple visualizations**: Through its different interfaces, it targets users of many skill levels, providing an API for quick and straightforward visualizations as well as more complex and extremely customizable ones.
- **Excellent animated visualizations**: It provides high performance and can, therefore, work on large or even streaming datasets, which makes it the go-to choice for animated visualizations and data analysis.
- **Inter-visualization interactivity**: This is a web-based approach; it's easy to combine several plots and create unique and impactful dashboards with visualizations that can be interconnected to create inter-visualization interactivity.
- **Supports multiple languages**: Other than Matplotlib and geoplotlib, Bokeh has libraries for both Python and JavaScript, in addition to several other popular languages.
- **Multiple ways to perform a task**: Adding interactivity to Bokeh visualizations can be done in several ways. The simplest built-in way is the ability to zoom and pan in and out of your visualization. This gives the users better control of what they want to see. It also allows users to filter and transform the data.
- **Beautiful chart styling**: The tech stack is based on Tornado in the backend and is powered by D3 in the frontend. D3 is a JavaScript library for creating outstanding visualizations. Using the underlying D3 visuals allows us to create beautiful plots without much custom styling.

Concepts of Bokeh

The basic concept of Bokeh is, in some ways, comparable to that of Matplotlib. In Bokeh, we have a figure as our root element, which has sub-elements such as a title, an axis, and **glyphs**. Glyphs have to be added to a figure, which can take on different shapes, such as circles, bars, and triangles. The following hierarchy shows the different concepts of Bokeh:

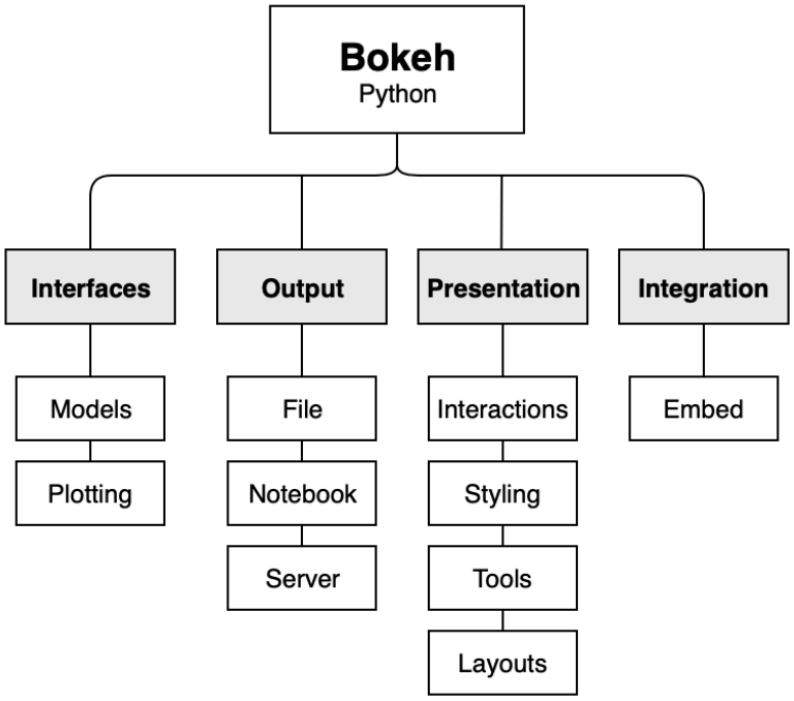

Figure 6.1: Concepts of Bokeh

---

2. a

12.6 Parsing HTML and scraping the web

One of the common uses of the `urllib` capability in Python is to *scrape* the web. Web scraping is when we write a program that pretends to be a web browser and retrieves pages, then examines the data in those pages looking for patterns.

As an example, a search engine such as Google will look at the source of one web page and extract the links to other pages and retrieve those pages, extracting links, and so on. Using this technique, Google *spiders* its way through nearly all of the pages on the web.

Google also uses the frequency of links from pages it finds to a particular page as one measure of how "important" a page is and how high the page should appear in its search results.

12.5 Reading binary files using `urllib`

Sometimes you want to retrieve a non-text (or binary) file such as an image or video file. The data in these files is generally not useful to print out, but you can easily make a copy of a URL to a local file on your hard disk using `urllib`.

The pattern is to open the URL and use `read` to download the entire contents of the document into a string variable (`img`) then write that information to a local file as follows:

```
import urllib.request, urllib.parse, urllib.error

img = urllib.request.urlopen('http://data.pr4e.org/cover3.jpg').read()
fhand = open('cover3.jpg', 'wb')
fhand.write(img)
fhand.close()

# Code: https://www.py4e.com/code3/curl1.py
```

This program reads all of the data in at once across the network and stores it in the variable `img` in the main memory of your computer, then opens the file `cover.jpg` and writes the data out to your disk. The `wb` argument for `open()` opens a binary file for writing only. This program will work if the size of the file is less than the size of the memory of your computer.

However if this is a large audio or video file, this program may crash or at least run extremely slowly when your computer runs out of memory. In order to avoid

running out of memory, we retrieve the data in blocks (or buffers) and then write each block to your disk before retrieving the next block. This way the program can read any size file without using up all of the memory you have in your computer.

---

3. a

Even though HTML looks like XML[2] and some pages are carefully constructed to be XML, most HTML is generally broken in ways that cause an XML parser to reject the entire page of HTML as improperly formed.

There are a number of Python libraries which can help you parse HTML and extract data from the pages. Each of the libraries has its strengths and weaknesses and you can pick one based on your needs.

As an example, we will simply parse some HTML input and extract links using the *BeautifulSoup* library. BeautifulSoup tolerates highly flawed HTML and still lets you easily extract the data you need. You can download and install the BeautifulSoup code from:

```
import urllib.request, urllib.parse, urllib.error
from bs4 import BeautifulSoup
import ssl

# Ignore SSL certificate errors
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

url = input('Enter - ')
html = urllib.request.urlopen(url, context=ctx).read()
soup = BeautifulSoup(html, 'html.parser')

# Retrieve all of the anchor tags
tags = soup('a')
for tag in tags:
    print(tag.get('href', None))

# Code: https://www.py4e.com/code3/urllinks.py
```

The program prompts for a web address, then opens the web page, reads the data and passes the data to the BeautifulSoup parser, and then retrieves all of the anchor tags and prints out the **href** attribute for each tag.

---

4. a

Import and call the **output_notebook** method from Bokeh's **io** interface to display the plots inside Jupyter Notebook:

```
# make bokeh display figures inside the notebook
from bokeh.io import output_notebook
output_notebook()
```

```
# importing the widgets
from ipywidgets import interact, interact_manual
```

```
@interact(Value=False)
def checkbox(Value=False):
    print(Value)
```

```
# creating a dropdown
options=['Option1', 'Option2', 'Option3', 'Option4']

@interact(Value=options)
def dropdown(Value=options[0]):
    print(Value)
```

```
# creating an input text
@interact(Value='Input Text')
def text_input(Value):
    print(Value)
```

---

5. a

13.1 eXtensible Markup Language - XML

XML looks very similar to HTML, but XML is more structured than HTML. Here is a sample of an XML document:

```
<person>
  <name>Chuck</name>
  <phone type="intl">
    +1 734 303 4456
  </phone>
  <email hide="yes" />
</person>
```

Each pair of opening (e.g., `<person>`) and closing tags (e.g., `</person>`) represents a *element* or *node* with the same name as the tag (e.g., `person`). Each element can have some text, some attributes (e.g., `hide`), and other nested elements. If an XML element is empty (i.e., has no content), then it may be depicted by a self-closing tag (e.g., `<email />`).

Often it is helpful to think of an XML document as a tree structure where there is a top element (here: `person`), and other tags (e.g., `phone`) are drawn as *children* of their *parent* elements.

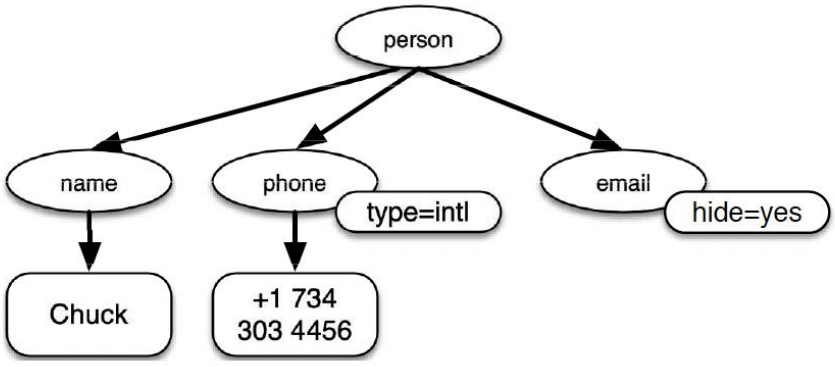

Figure 13.1: A Tree Representation of XML

13.3 Looping through nodes

Often the XML has multiple nodes and we need to write a loop to process all of the nodes. In the following program, we loop through all of the `user` nodes:

```
import xml.etree.ElementTree as ET

input = '''
<stuff>
  <users>
    <user x="2">
      <id>001</id>
      <name>Chuck</name>
    </user>
    <user x="7">
      <id>009</id>
      <name>Brent</name>
    </user>
  </users>
</stuff>'''

stuff = ET.fromstring(input)
lst = stuff.findall('users/user')
print('User count:', len(lst))

for item in lst:
    print('Name', item.find('name').text)
    print('Id', item.find('id').text)
    print('Attribute', item.get('x'))

# Code: https://www.py4e.com/code3/xml2.py
```

The `findall` method retrieves a Python list of subtrees that represent the `user` structures in the XML tree. Then we can write a `for` loop that looks at each of the user nodes, and prints the `name` and `id` text elements as well as the `x` attribute from the `user` node.

```
User count: 2
Name Chuck
Id 001
Attribute 2
Name Brent
Id 009
Attribute 7
```

---

6. a

13.4 JavaScript Object Notation - JSON

The JSON format was inspired by the object and array format used in the JavaScript language. But since Python was invented before JavaScript, Python's syntax for dictionaries and lists influenced the syntax of JSON. So the format of JSON is nearly identical to a combination of Python lists and dictionaries.

Here is a JSON encoding that is roughly equivalent to the simple XML from above:

```
{
  "name" : "Chuck",
  "phone" : {
    "type" : "intl",
    "number" : "+1 734 303 4456"


  },
  "email" : {
    "hide" : "yes"
  }
}
```

You will notice some differences. First, in XML, we can add attributes like "intl" to the "phone" tag. In JSON, we simply have key-value pairs. Also the XML "person" tag is gone, replaced by a set of outer curly braces.

13.5 Parsing JSON

We construct our JSON by nesting dictionaries and lists as needed. In this example, we represent a list of users where each user is a set of key-value pairs (i.e., a dictionary). So we have a list of dictionaries.

In the following program, we use the built-in `json` library to parse the JSON and read through the data. Compare this closely to the equivalent XML data and code above. The JSON has less detail, so we must know in advance that we are getting a list and that the list is of users and each user is a set of key-value pairs. The JSON is more succinct (an advantage) but also is less self-describing (a disadvantage).

```
import json

data = '''
[
  { "id" : "001",
    "x" : "2",
    "name" : "Chuck"
  } ,
  { "id" : "009",
    "x" : "7",
    "name" : "Brent"
  }
]'''

info = json.loads(data)
print('User count:', len(info))

for item in info:
    print('Name', item['name'])


     print('Id', item['id'])
     print('Attribute', item['x'])
```

 # Code: https://www.py4e.com/code3/json2.py

If you compare the code to extract data from the parsed JSON and XML you will see that what we get from `json.loads()` is a Python list which we traverse with a `for` loop, and each item within that list is a Python dictionary. Once the JSON has been parsed, we can use the Python index operator to extract the various bits of data for each user. We don't have to use the JSON library to dig through the parsed JSON, since the returned data is simply native Python structures.

The output of this program is exactly the same as the XML version above.

```
User count: 2
Name Chuck
Id 001
Attribute 2
Name Brent
Id 009
Attribute 7
```