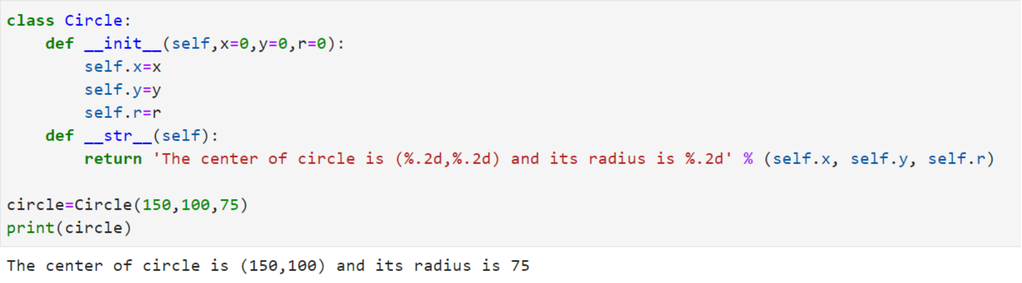
USN □□□□□□□□□□

CMRIT
CELEBRATING 25 YEARS
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 3 – Jan 2022

Sub:	Application Development using Python					Sub Code:	18CS55	Branch:	CSE
Date:	27-01-2022	Duration:	90 mins	Max Marks:	50	Sem / Sec:	A, B & C		OBE

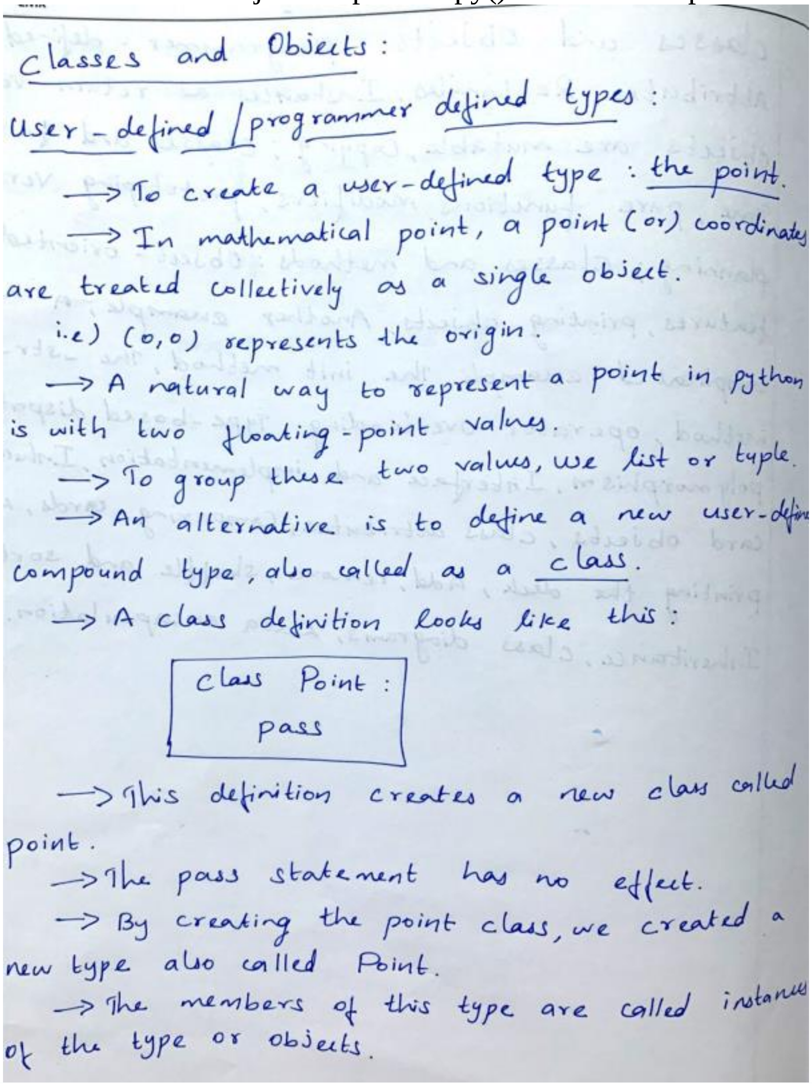
	Answer any FIVE FULL Questions	MARKS	CO	RBT
1 (a)	Write a program that instantiate a Circle object that represents a circle with its center at (150,100) and radius 75.	[05]	CO4	L3



```
class Circle:
    def __init__(self,x=0,y=0,r=0):
        self.x=x
        self.y=y
        self.r=r
    def __str__(self):
        return 'The center of circle is (%.2d,%.2d) and its radius is %.2d' % (self.x, self.y, self.r)

circle=Circle(150,100,75)
print(circle)
```

```
The center of circle is (150,100) and its radius is 75
```

b)	Define class and object. Explain copy() with an example.	[05]	CO4	L2



Classes and Objects :

User-defined / programmer defined types :

→ To create a user-defined type : the point.

→ In mathematical point, a point (or) coordinates are treated collectively as a single object.

i.e) (0,0) represents the origin.

→ A natural way to represent a point in python is with two floating-point values.

→ To group these two values, we list or tuple.

→ An alternative is to define a new user-defined compound type, also called as a class.

→ A class definition looks like this :

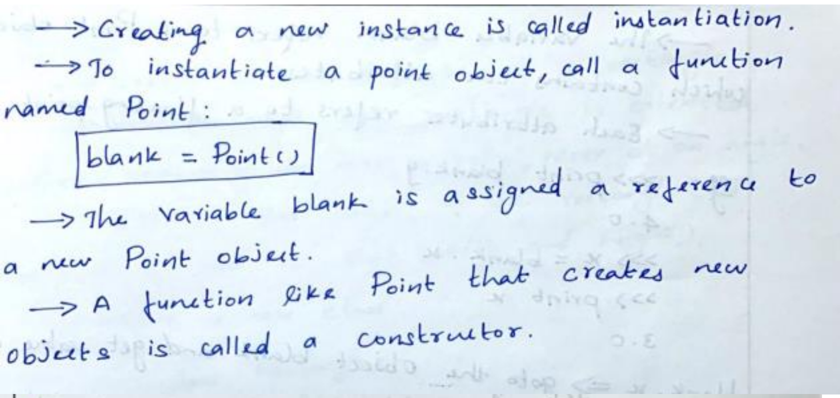
```
class Point :
    pass
```

→ This definition creates a new class called point.

→ The pass statement has no effect.

→ By creating the point class, we created a new type also called Point.

→ The members of this type are called instances of the type or objects.



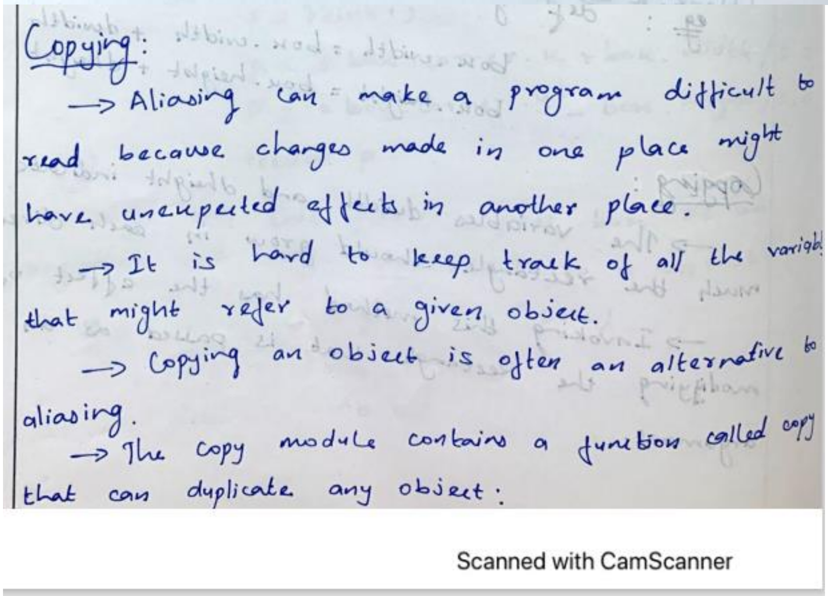
→ Creating a new instance is called instantiation.

→ To instantiate a point object, call a function named Point :

```
blank = Point()
```

→ The variable blank is assigned a reference to a new Point object.

→ A function like Point that creates new objects is called a constructor.



Copying:

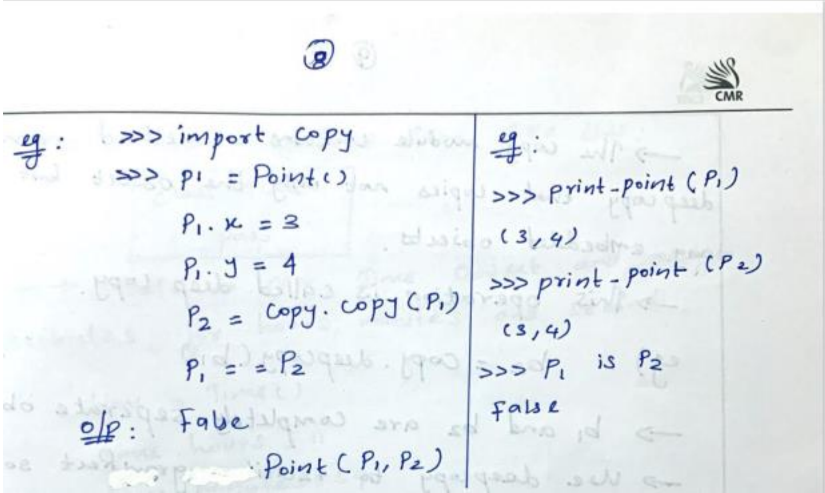
→ Aliasing can make a program difficult to read because changes made in one place might have unexpected effects in another place.

→ It is hard to keep track of all the variabl that might refer to a given object.

→ Copying an object is often an alternative to aliasing.

→ The copy module contains a function called copy that can duplicate any object :

Scanned with CamScanner



8

eg :

```
>>> import copy
>>> p1 = Point()
    P1.x = 3
    P1.y = 4
    P2 = copy.copy(P1)
    P1 = = P2
o/p : False
          Point (P1, P2)
```

eg :

```
>>> print_point (P1)
(3, 4)
>>> print_point (P2)
(3, 4)
>>> P1 is P2
False
```

2 (a) Describe pure functions and modifiers with examples [05] CO4 L2

Pure function: The function creates a new object, initializes its attributes, and returns a reference to the new object. This is called a pure function because it does not modify any of the objects passed to it as arguments and it has no effect, like displaying a value or getting user input, other than returning a value.

Here is a simple prototype of
add_time:

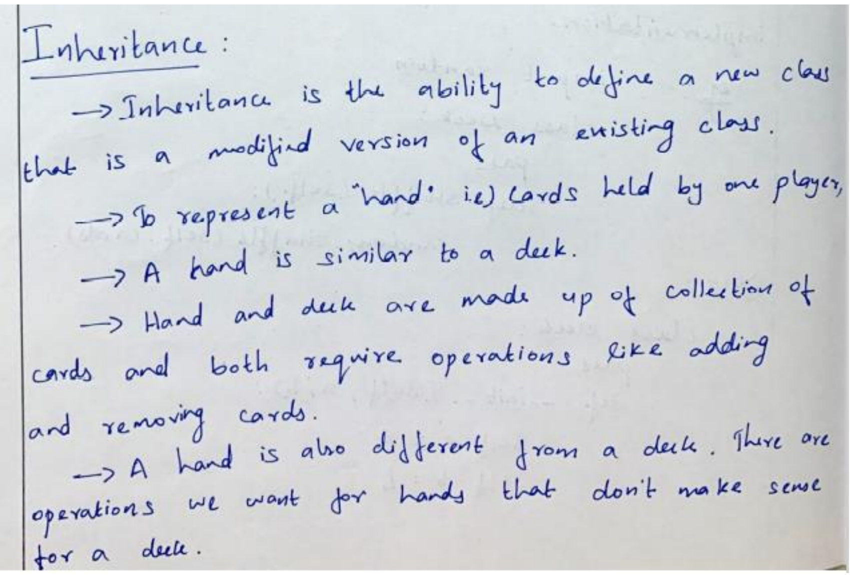
```
def add_time(t1, t2):
sum = Time()
sum.hour = t1.hour + t2.hour
sum.minute = t1.minute + t2.minute
sum.second = t1.second + t2.second
```

```
    return sum
>>> start = Time()
>>> start.hour = 9
>>> start.minute = 45
>>> start.second = 0
>>> duration = Time()
>>> duration.hour = 1
>>> duration.minute = 35
>>> duration.second = 0
>>> done = add_time(start, duration)
>>> print_time(done)

10:80:00
```

The problem is that this function does not deal with cases where the number of seconds or minutes adds up to more than sixty. Modifiers: Sometimes it is useful for a function to modify the objects it gets as parameters. In that case, the changes are visible to the caller. Functions that work this way are called modifiers.

```
#addtime.py
class Time:
    def gettime(self):
       self.hr=int(input("Enter hours: "))
       self.min=int(input("Enter minutes: "))
       self.sec=int(input("Enter seconds: "))

    def add_time(self,t1,t2):
        sumt=Time()
       sumt.sec=t1.sec+t2.sec
      xmin, rsec = divmod(sumt.sec, 60)
      sumt.sec=rsec sumt.min=t1.min+t2.min+xmin
      xhr, rmin = divmod(sumt.min, 60)
      sumt.min=rmin sumt.hr=t1.hr+t2.hr+xhr
      return sumt
t1=Time()
t1.gettime()
t2=Time()
t2.gettime()
t3=Time()
t3=t3.add_time(t1,t2)
print("Adding both times : ",t3.hr,t3.min,t3.sec)
#expected output
Enter hours: 23
Enter minutes: 45
Enter seconds: 54
Enter hours: 9
Enter minutes: 32
Enter seconds: 51
Adding both times : 33 18 45
```

(b)	Illustrate the concept of inheritance and class diagrams with examples.	[05]	CO4	L2



Inheritance:

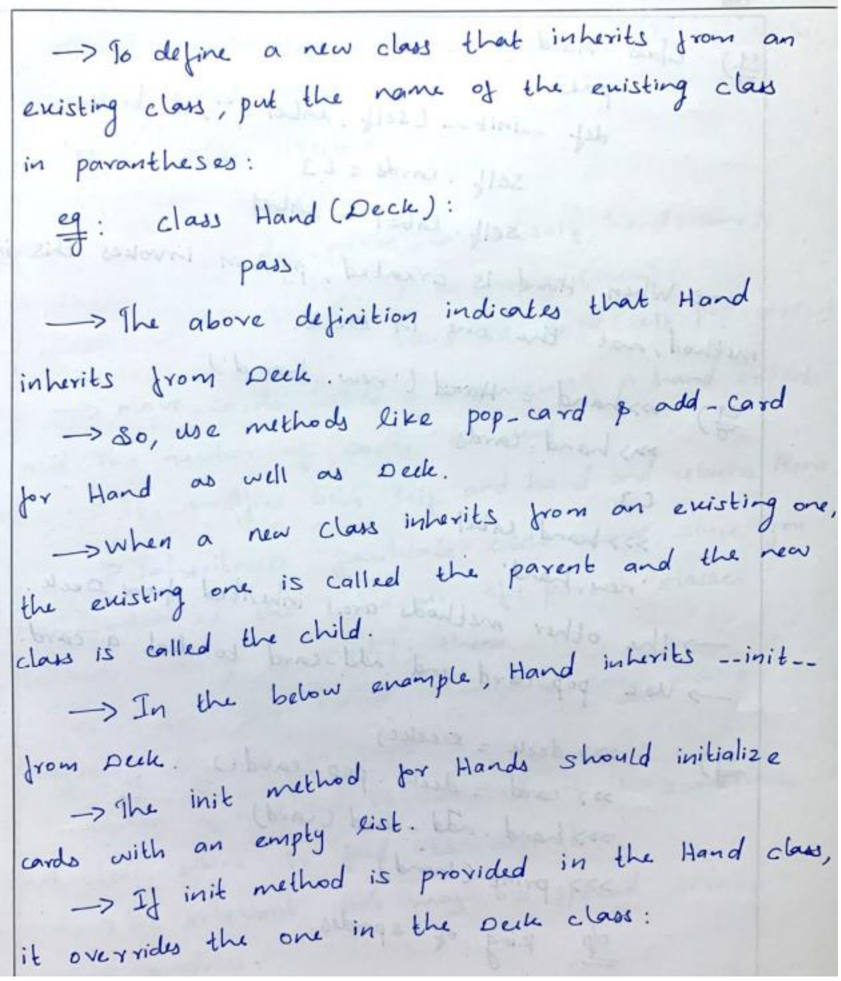
-> Inheritance is the ability to define a new class that is a modified version of an existing class.

-> To represent a 'hand' ie) cards held by one player,

-> A hand is similar to a deck.

-> Hand and deck are made up of collection of cards and both require operations like adding and removing cards.

-> A hand is also different from a deck. There are operations we want for hands that don't make sense for a deck.



-> To define a new class that inherits from an existing class, put the name of the existing class in parantheses:

eg:
```
class Hand (Deck):
        pass
```

-> The above definition indicates that Hand inherits from Deck.

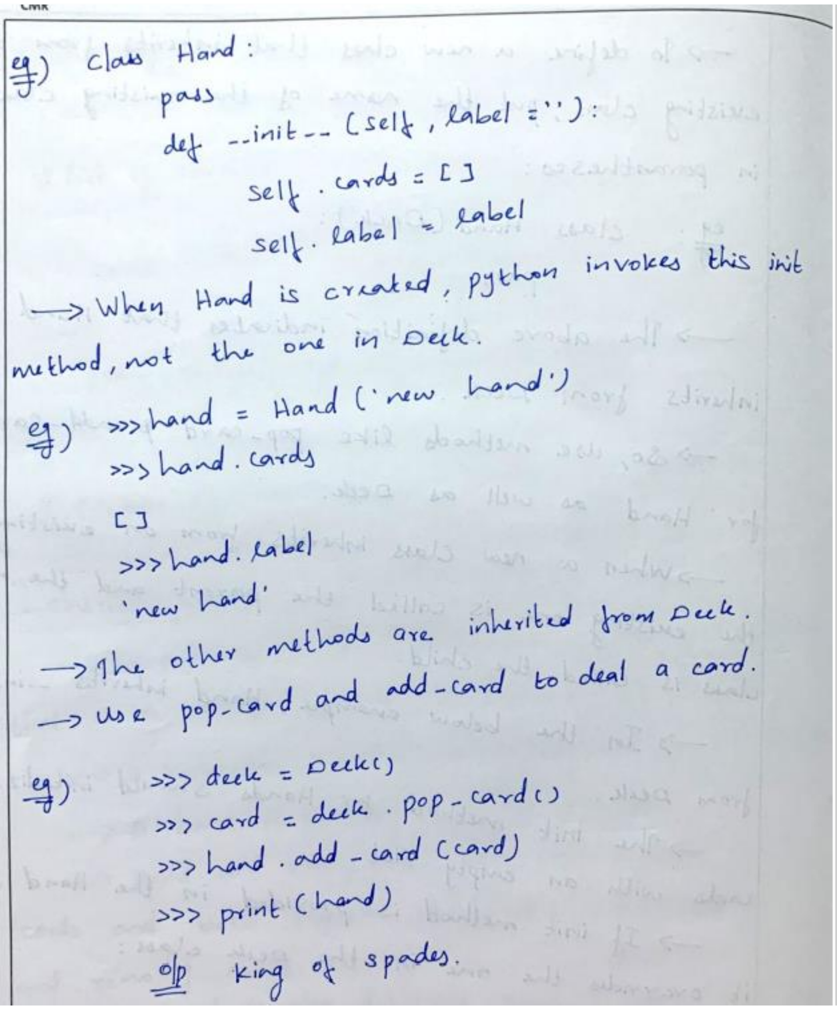
-> So, use methods like pop-card & add-card for Hand as well as Deck.

-> When a new class inherits from an existing one, the existing one is called the parent and the new class is called the child.

-> In the below example, Hand inherits --init-- from Deck.

-> The init method for Hands should initialize cards with an empty list.

-> If init method is provided in the Hand class, it overrides the one in the Deck class:



eg)
```
class Hand:
    pass
    def --init-- (self, label =''):
        self.cards = []
        self.label = label
```

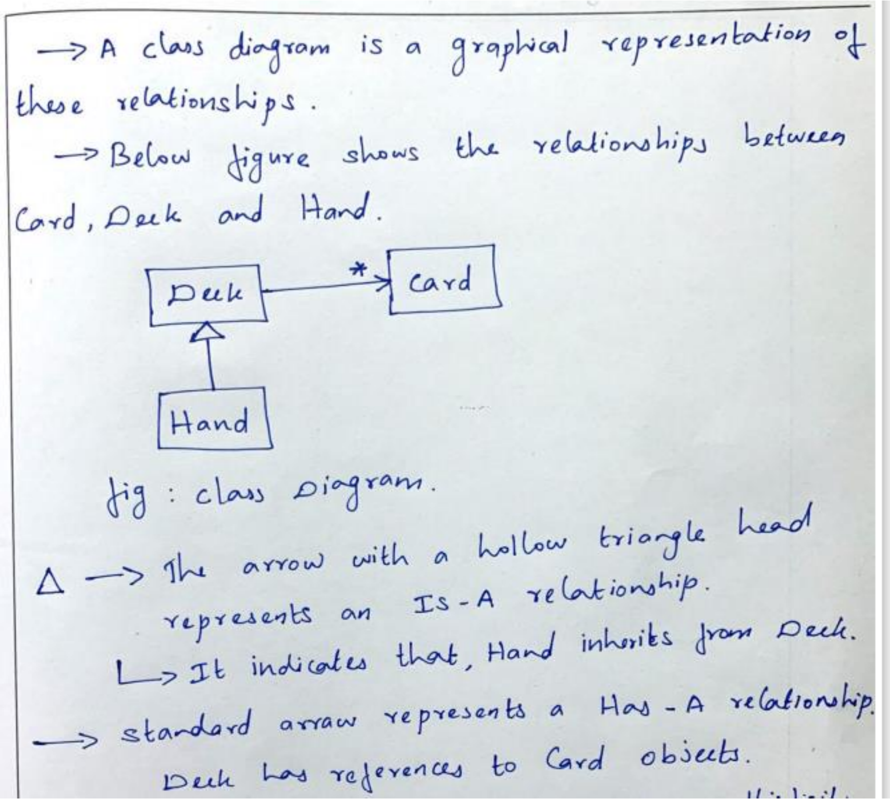
→ When Hand is created, python invokes this init method, not the one in Deck.

eg)
```
>>>hand = Hand('new hand')
>>>hand.cards

[]
>>>hand.label
'new hand'
```

→ The other methods are inherited from Deck.

→ use pop-card and add-card to deal a card.

eg)
```
>>> deck = Deck()
>>> card = deck.pop-card()
>>> hand.add-card(card)
>>> print(hand)
o/p   King of spades.
```



→ A class diagram is a graphical representation of these relationships.

→ Below figure shows the relationships between Card, Deck and Hand.

Deck
*
Card
Hand

fig : class Diagram.

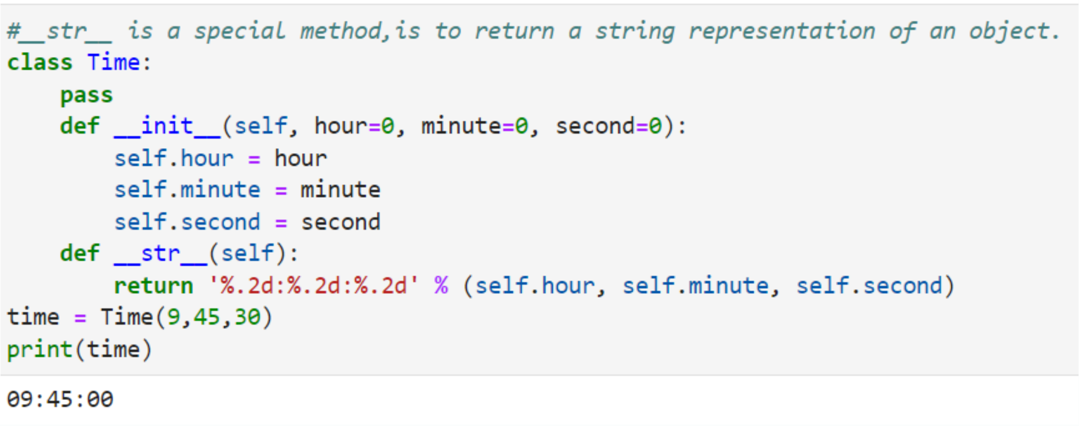
△ → The arrow with a hollow triangle head represents an IS-A relationship.

└→ It indicates that, Hand inherits from Deck.

→ standard arraw represents a Has-A relationship.

Deck has references to Card objects.

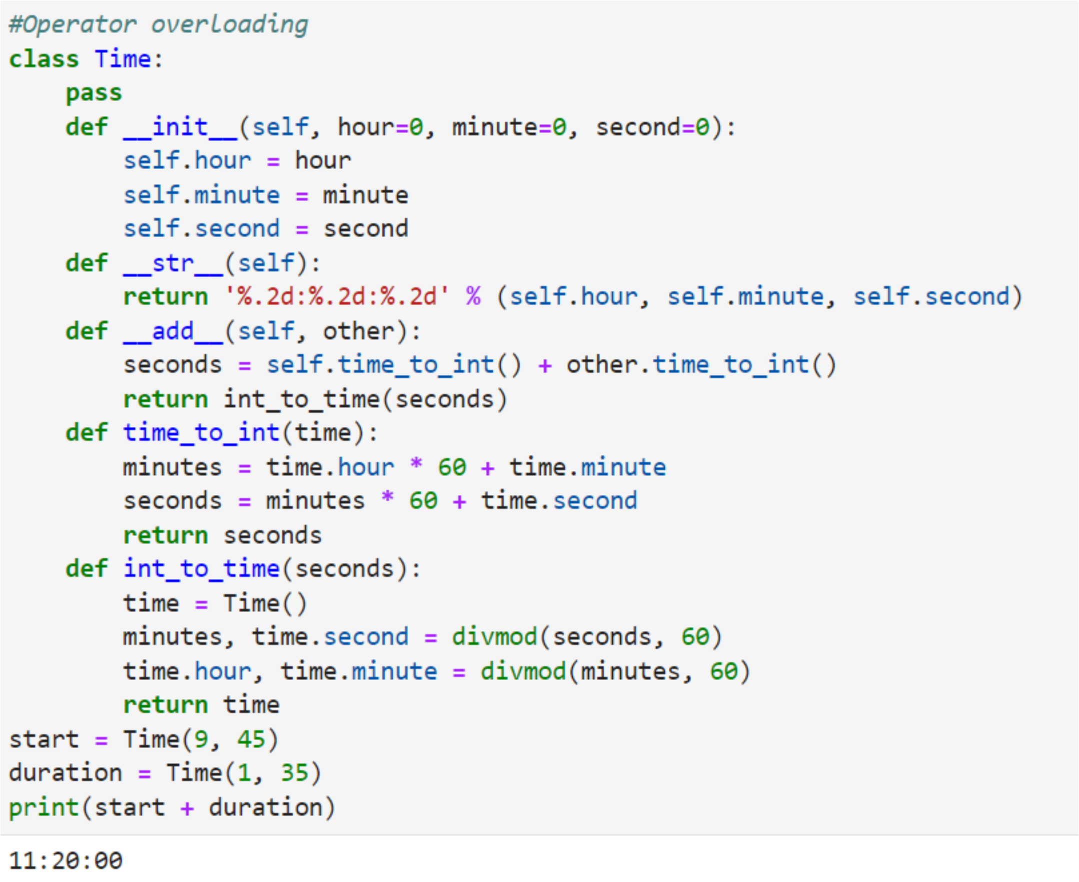
3 (a) Implement a Time class with methods to display time in HH:MM:SS with __str__() method. [05] CO4 L3



```
#__str__ is a special method,is to return a string representation of an object.
class Time:
    pass
    def __init__(self, hour=0, minute=0, second=0):
        self.hour = hour
        self.minute = minute
        self.second = second
    def __str__(self):
        return '%.2d:%.2d:%.2d' % (self.hour, self.minute, self.second)
time = Time(9,45,30)
print(time)
```

```
09:45:00
```

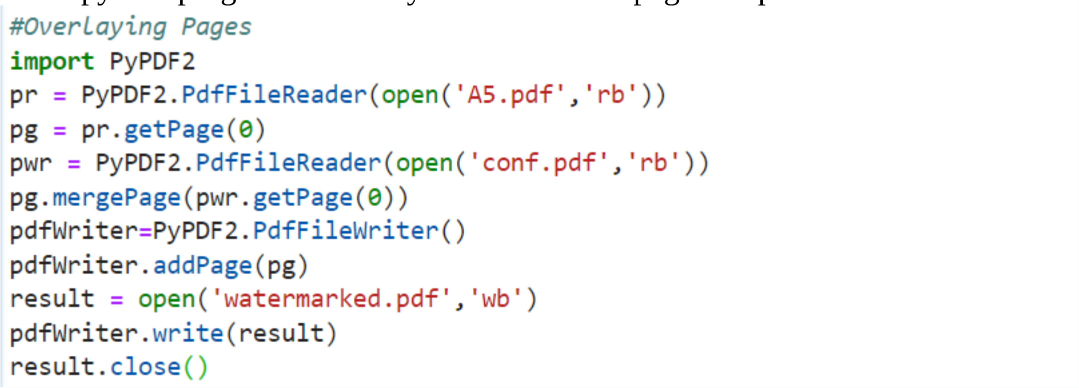
(b) Implement a Time class with methods to add two time objects using operator overloading. [05] CO4 L3



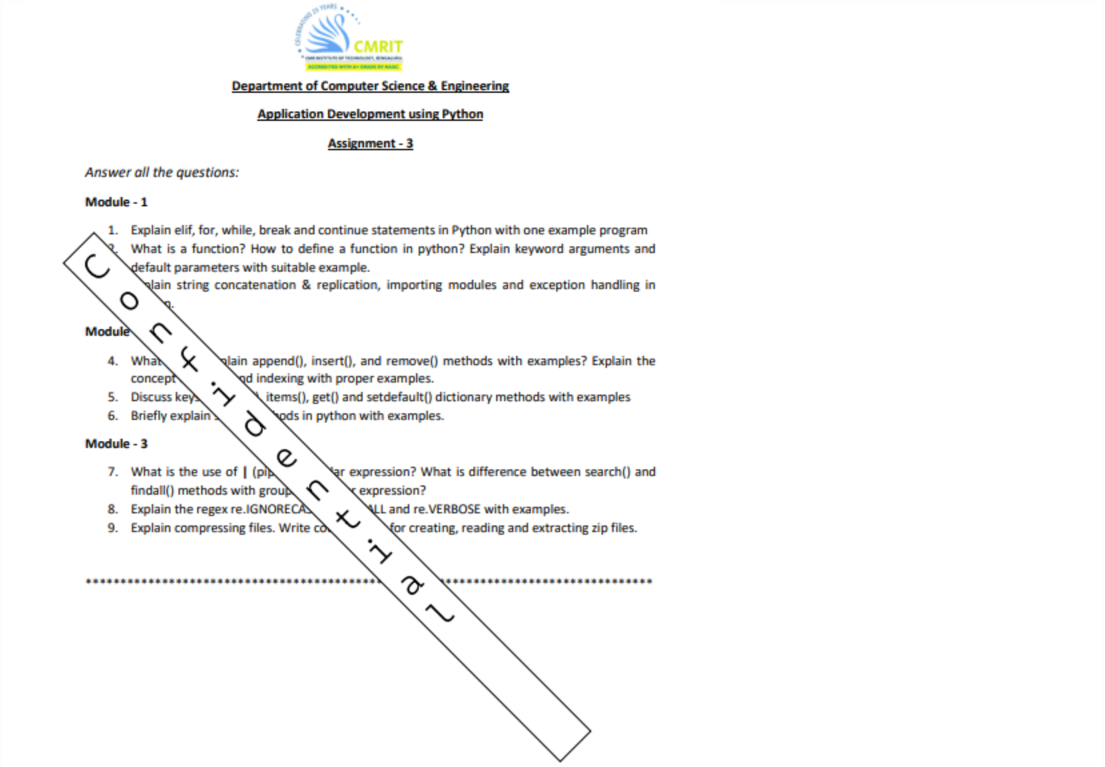
```
#Operator overloading
class Time:
    pass
    def __init__(self, hour=0, minute=0, second=0):
        self.hour = hour
        self.minute = minute
        self.second = second
    def __str__(self):
        return '%.2d:%.2d:%.2d' % (self.hour, self.minute, self.second)
    def __add__(self, other):
        seconds = self.time_to_int() + other.time_to_int()
        return int_to_time(seconds)
    def time_to_int(time):
        minutes = time.hour * 60 + time.minute
        seconds = minutes * 60 + time.second
        return seconds
    def int_to_time(seconds):
        time = Time()
        minutes, time.second = divmod(seconds, 60)
        time.hour, time.minute = divmod(minutes, 60)
        return time
start = Time(9, 45)
duration = Time(1, 35)
print(start + duration)
```

```
11:20:00
```

4 (a) Write python program to overlay contents of one page of a pdf file over another file. [05] CO5 L3



```
#Overlaying Pages
import PyPDF2
pr = PyPDF2.PdfFileReader(open('A5.pdf','rb'))
pg = pr.getPage(0)
pwr = PyPDF2.PdfFileReader(open('conf.pdf','rb'))
pg.mergePage(pwr.getPage(0))
pdfWriter=PyPDF2.PdfFileWriter()
pdfWriter.addPage(pg)
result = open('watermarked.pdf','wb')
pdfWriter.write(result)
result.close()
```



Department of Computer Science & Engineering

Application Development using Python

Assignment - 3

Answer all the questions:

Module - 1

1. Explain elif, for, while, break and continue statements in Python with one example program
2. What is a function? How to define a function in python? Explain keyword arguments and default parameters with suitable example.
3. [illegible] plain string concatenation & replication, importing modules and exception handling in [illegible] n.

Module [illegible]

4. What [illegible] plain append(), insert(), and remove() methods with examples? Explain the concept [illegible] nd indexing with proper examples.
5. Discuss keys [illegible] items(), get() and setdefault() dictionary methods with examples
6. Briefly explain [illegible] ods in python with examples.

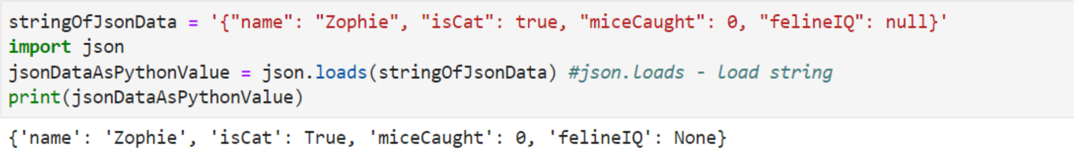
Module - 3

7. What is the use of | (pip [illegible] lar expression? What is difference between search() and findall() methods with group [illegible] r expression?
8. Explain the regex re.IGNORECA [illegible] LL and re.VERBOSE with examples.
9. Explain compressing files. Write co [illegible] for creating, reading and extracting zip files.

****

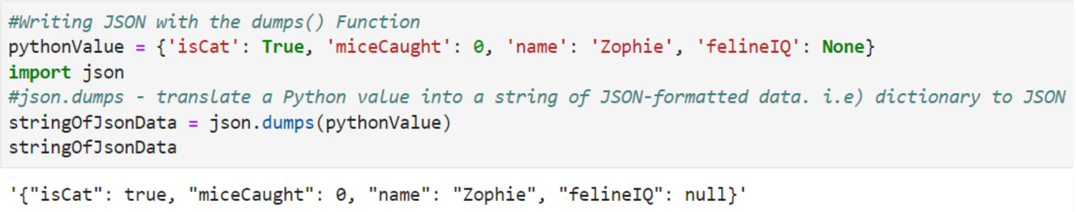
(b)	Explain JSON module with its methods loads() and dumps() with examples.	[05]	CO5	L2

Python’s json module handles all the details of translating between a string with JSON data and Python values for the json.loads() and json.dumps() functions. JSON can’t store every kind of Python value. It can contain values of only the following data types: strings, integers, floats, Booleans, lists, dictionaries, and NoneType. JSON cannot represent Python-specific objects, such as File objects, CSV Reader or Writer objects, Regex objects, or Selenium WebElement objects. The json.dumps() function (which means “dump string,” not “dumps”) will translate a Python value into a string of JSON-formatted data.



```
stringOfJsonData = '{"name": "Zophie", "isCat": true, "miceCaught": 0, "felineIQ": null}'
import json
jsonDataAsPythonValue = json.loads(stringOfJsonData) #json.loads - load string
print(jsonDataAsPythonValue)
```

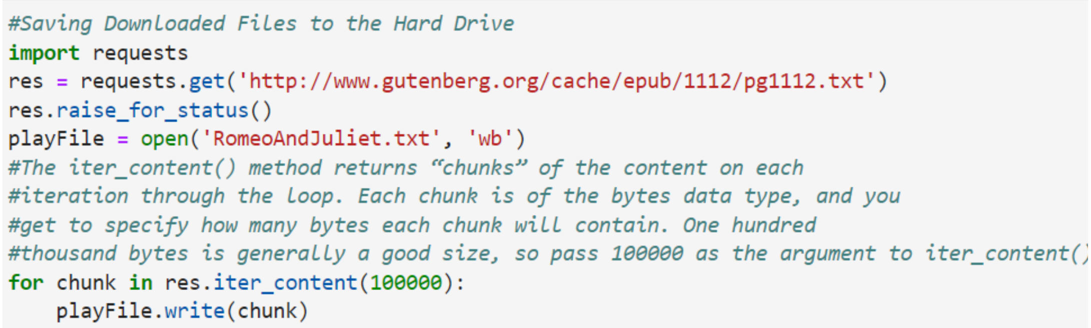
```
{'name': 'Zophie', 'isCat': True, 'miceCaught': 0, 'felineIQ': None}
```



```
#Writing JSON with the dumps() Function
pythonValue = {'isCat': True, 'miceCaught': 0, 'name': 'Zophie', 'felineIQ': None}
import json
#json.dumps - translate a Python value into a string of JSON-formatted data. i.e) dictionary to JSON
stringOfJsonData = json.dumps(pythonValue)
stringOfJsonData
```

```
'{"isCat": true, "miceCaught": 0, "name": "Zophie", "felineIQ": null}'
```

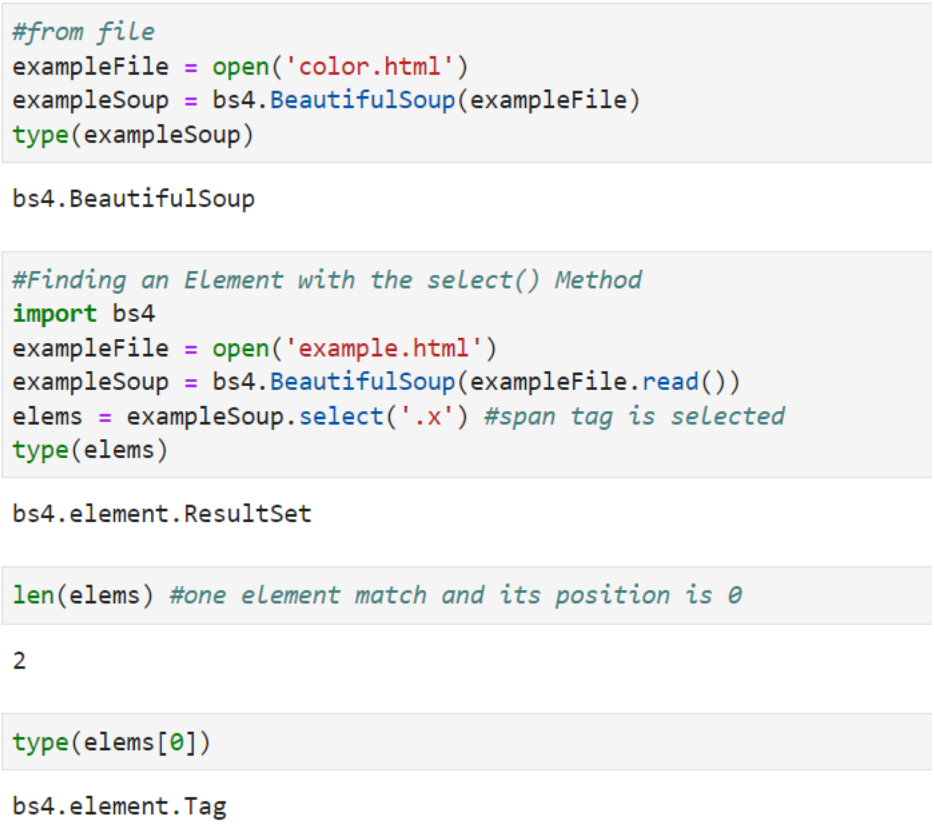
5 (a)	Write a python program to download files from a URL to hard drive using requests module.	[05]	CO5	L3



```
#Saving Downloaded Files to the Hard Drive
import requests
res = requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt')
res.raise_for_status()
playFile = open('RomeoAndJuliet.txt', 'wb')
#The iter_content() method returns “chunks” of the content on each
#iteration through the loop. Each chunk is of the bytes data type, and you
#get to specify how many bytes each chunk will contain. One hundred
#thousand bytes is generally a good size, so pass 100000 as the argument to iter_content(
for chunk in res.iter_content(100000):
    playFile.write(chunk)
```

(b) Write short notes on beautiful soup and selenium module. [05] CO5 L2

Beautiful Soup is a module for extracting information from an HTML page (and is much better for this purpose than regular expressions). The BeautifulSoup module's name is bs4 (for Beautiful Soup, version 4). Here parsing means the BeautifulSoup can analyze and identify the parts of) an HTML file on the hard drive. We can open a new file editor window in IDLE, enter the following, and save it as example.html.



```
#from file
exampleFile = open('color.html')
exampleSoup = bs4.BeautifulSoup(exampleFile)
type(exampleSoup)
```

```
bs4.BeautifulSoup
```

```
#Finding an Element with the select() Method
import bs4
exampleFile = open('example.html')
exampleSoup = bs4.BeautifulSoup(exampleFile.read())
elems = exampleSoup.select('.x') #span tag is selected
type(elems)
```

```
bs4.element.ResultSet
```

```
len(elems) #one element match and its position is 0
```

```
2
```

```
type(elems[0])
```

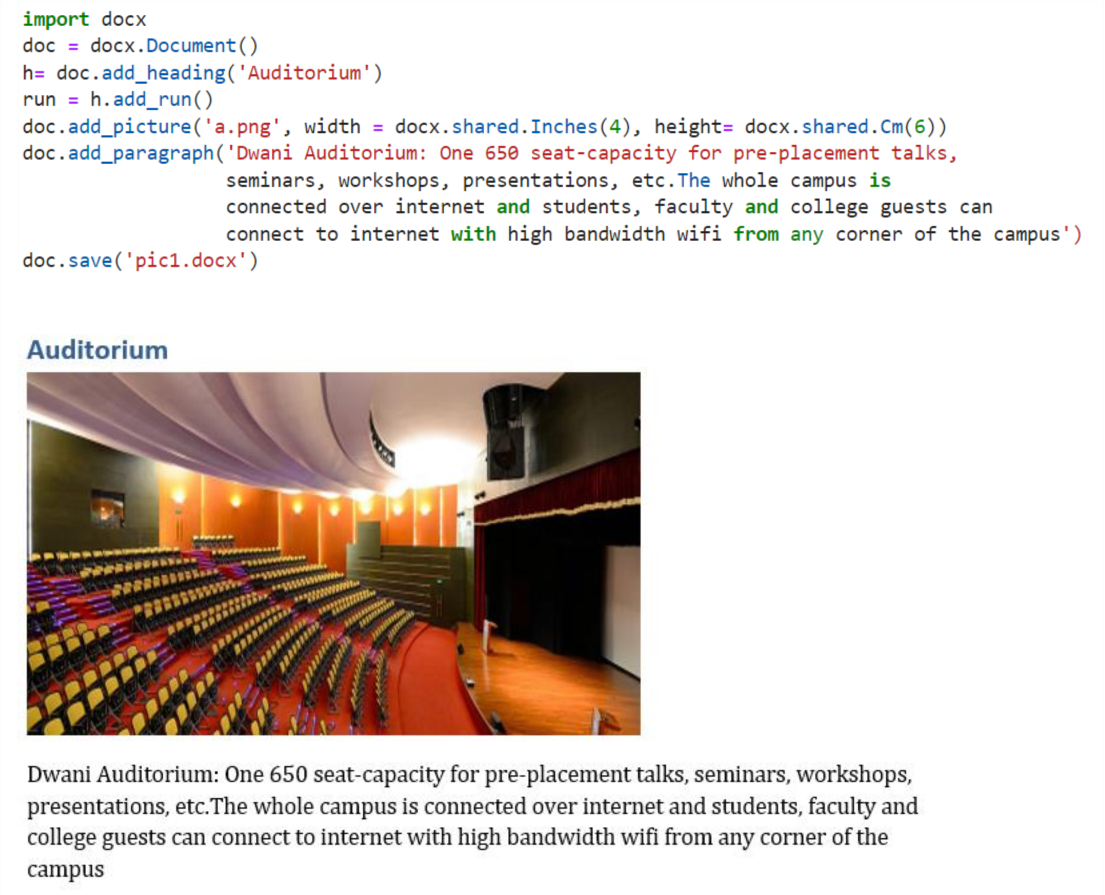
```
bs4.element.Tag
```

Actions class is an ability provided by Selenium for handling keyboard and mouse events. In Selenium WebDriver, handling these events includes operations such as drag and drop, clicking on multiple elements with the control key, among others. These operations are performed using the advanced user interactions API. It mainly consists of Actions that are needed while performing these operations. Action class is defined and invoked using the following syntax:

Actions action = new Actions(driver);
action.moveToElement(element).click().perform();
driver.get("cmrit.ac.in"); Actions action = new Actions(driver);
element = driver.findElement(By.linkText("Academic"));
action.moveToElement(element).click();

Mouse Actions in Selenium:
doubleClick(): Performs double click on the element
clickAndHold(): Performs long click on the mouse without releasing it
dragAndDrop(): Drags the element from one point and drops to another
moveToElement(): Shifts the mouse pointer to the center of the element
contextClick(): Performs right-click on the mouse Keyboard Actions in Selenium:
sendKeys(): Sends a series of keys to the element
keyUp(): Performs key release keyDown(): Performs keypress without release

6 (a) Write a python program to create a word document and add an image, heading of the image and description of the image as paragraph. [05] CO5 L3

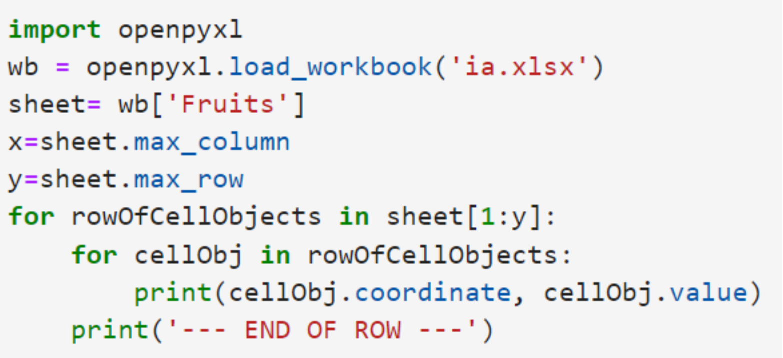


```python
import docx
doc = docx.Document()
h= doc.add_heading('Auditorium')
run = h.add_run()
doc.add_picture('a.png', width = docx.shared.Inches(4), height= docx.shared.Cm(6))
doc.add_paragraph('Dwani Auditorium: One 650 seat-capacity for pre-placement talks,
                  seminars, workshops, presentations, etc.The whole campus is
                  connected over internet and students, faculty and college guests can
                  connect to internet with high bandwidth wifi from any corner of the campus')
doc.save('pic1.docx')
```

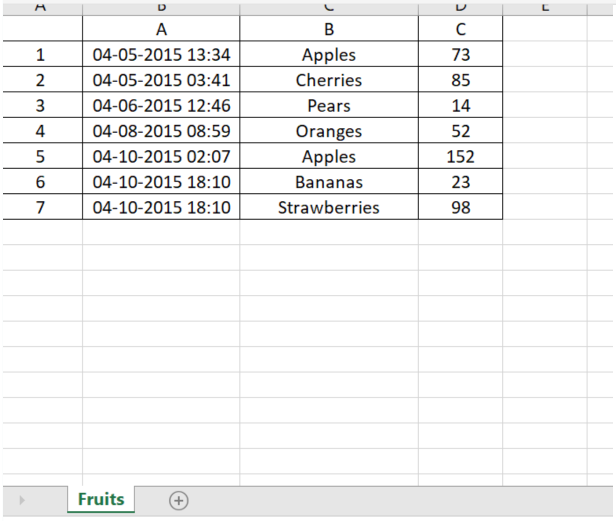
Auditorium

Dwani Auditorium: One 650 seat-capacity for pre-placement talks, seminars, workshops, presentations, etc.The whole campus is connected over internet and students, faculty and college guests can connect to internet with high bandwidth wifi from any corner of the campus

(b) Write a program to print row values of active sheet in separate line by importing openpyxl module. [05] CO5 L3

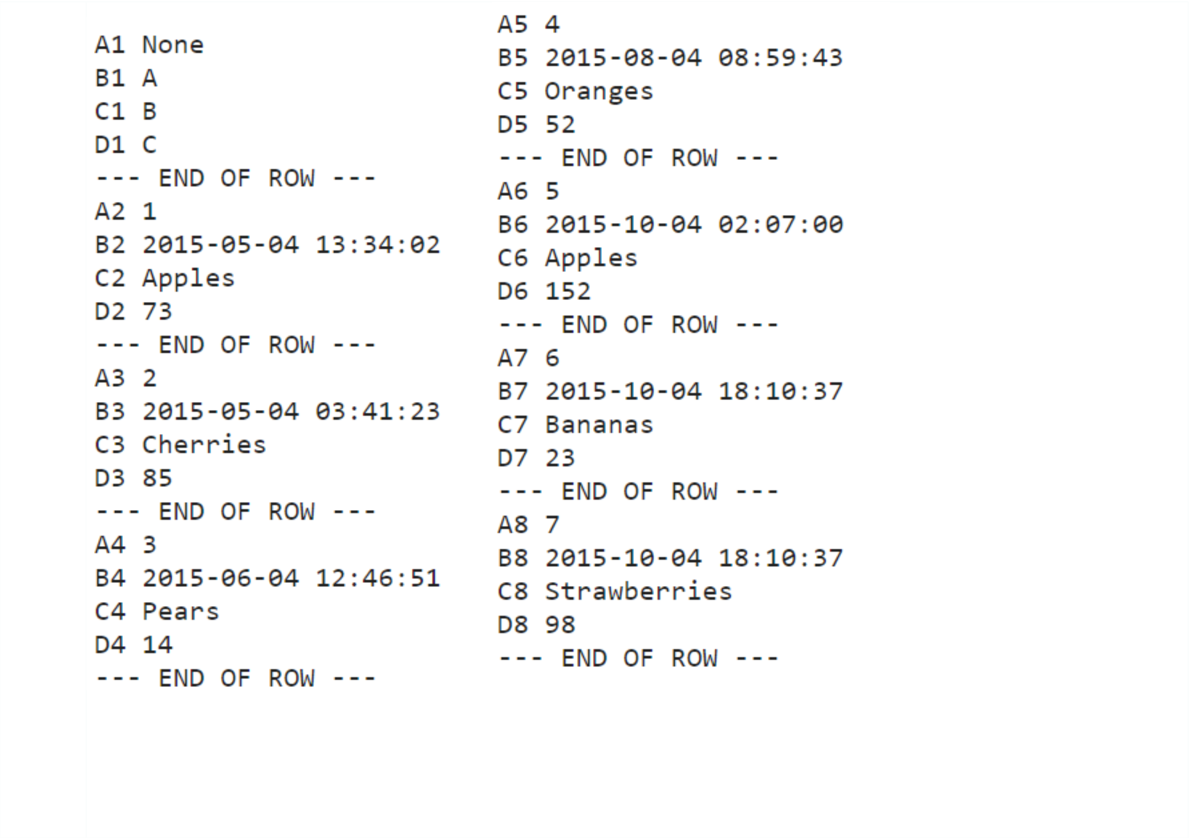


```python
import openpyxl
wb = openpyxl.load_workbook('ia.xlsx')
sheet= wb['Fruits']
x=sheet.max_column
y=sheet.max_row
for rowOfCellObjects in sheet[1:y]:
    for cellObj in rowOfCellObjects:
        print(cellObj.coordinate, cellObj.value)
    print('--- END OF ROW ---')
```



	A	B	C
1	04-05-2015 13:34	Apples	73
2	04-05-2015 03:41	Cherries	85
3	04-06-2015 12:46	Pears	14
4	04-08-2015 08:59	Oranges	52
5	04-10-2015 02:07	Apples	152
6	04-10-2015 18:10	Bananas	23
7	04-10-2015 18:10	Strawberries	98

Output:



```
A1 None
B1 A
C1 B
D1 C
--- END OF ROW ---
A2 1
B2 2015-05-04 13:34:02
C2 Apples
D2 73
--- END OF ROW ---
A3 2
B3 2015-05-04 03:41:23
C3 Cherries
D3 85
--- END OF ROW ---
A4 3
B4 2015-06-04 12:46:51
C4 Pears
D4 14
--- END OF ROW ---
A5 4
B5 2015-08-04 08:59:43
C5 Oranges
D5 52
--- END OF ROW ---
A6 5
B6 2015-10-04 02:07:00
C6 Apples
D6 152
--- END OF ROW ---
A7 6
B7 2015-10-04 18:10:37
C7 Bananas
D7 23
--- END OF ROW ---
A8 7
B8 2015-10-04 18:10:37
C8 Strawberries
D8 98
--- END OF ROW ---
```