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

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

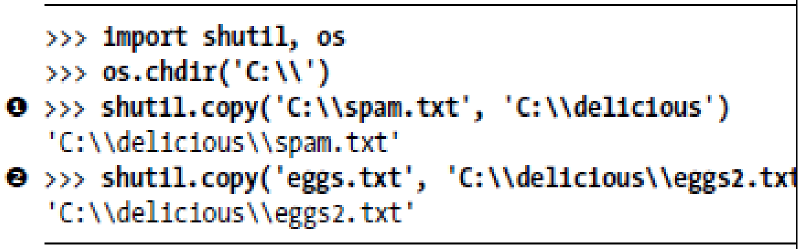
Internal Assessment Test III – Jan 2022

Sub:	Application Development Using Python			Sub Code:	18CS55	Branch:	ISE
Date:		Duration:	90 mins	Max Marks:	50	Sem/Sec:	V A,B & C	OBE

Answer any FIVE FULL Questions		MARKS	CO	RBT
1a)	Demonstrate the copy, move, rename and delete functions of shutil module with Python code snippet.	5	CO3	L3

Examples--- 2.5 marks Explanation 2.5 marks

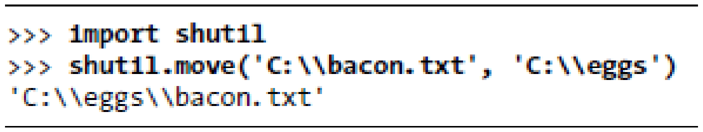
<u>Copy:</u> shutil.copy(source, destination) will copy the file from source path to destination path.



```
   >>> import shutil, os
   >>> os.chdir('C:\\')
❶ >>> shutil.copy('C:\\spam.txt', 'C:\\delicious')
   'C:\\delicious\\spam.txt'
❷ >>> shutil.copy('eggs.txt', 'C:\\delicious\\eggs2.txt
   'C:\\delicious\\eggs2.txt'
```

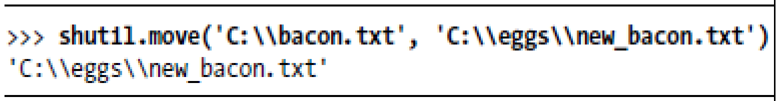
In the above example:
1 → Destination is a folder, hence the source file is copied to destination file.
2 → Destination is a folder with a filename, hence the source file copied will be renamed to the given filename.

<u>Move:</u> shutil.move(source, destination) will move the file or folder from source path to the destination path and will return a string of the absolute path of the new location.



```
>>> import shutil
>>> shutil.move('C:\\bacon.txt', 'C:\\eggs')
'C:\\eggs\\bacon.txt'
```

<u>Rename:</u> shutil.move(source, destination) can be used to move and rename also.



```
>>> shutil.move('C:\\bacon.txt', 'C:\\eggs\\new_bacon.txt')
'C:\\eggs\\new_bacon.txt'
```

<u>Delete:</u> We can delete a single file or a single empty folder with functions in the os module, whereas to delete a folder and all of its contents, we use the shutil module.

1. Calling os.unlink(path) will delete the file at path.
2. Calling os.rmdir(path) will delete the folder at path. This folder must be empty of any files or folders.
3. Calling shutil.rmtree(path) will remove the folder at path, and all files and folders it contains will also be deleted.

<u>Ex:</u> import os

```
                    for filename in in os.listdir( ):
if filename.endswith(‘.txt’): os.unlink(filename)
```

1b)	How do we specify and handle absolute, and relative paths?	5	CO3	L2

Examples methods--- 2.5 marks Explanation 2.5 marks

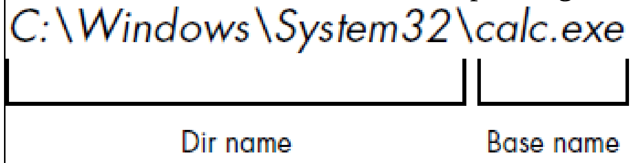
- ✓ There are two ways to specify a file path.
  1. An absolute path, which always begins with the root folder
  2. A relative path, which is relative to the program’s current working directory
- ✓ The os.path module provides functions for returning the absolute path of a relative path and for checking whether a given path is an absolute path.
  1. Calling os.path.abspath(path) will return a string of the absolute path of the argument. This is an easy way to convert a relative path into an absolute one.
  2. Calling os.path.isabs(path) will return True if the argument is an absolute path and False if it is a relative path.
  3. Calling os.path.relpath(path, start) will return a string of a relative path from the start path to path. If start is not provided, the current working directory is used as the start path.

```
>>> os.path.relpath('C:\\Windows', 'C:\\')
'Windows'
>>> os.path.relpath('C:\\Windows', 'C:\\spam\\eggs')
'..\\..\\Windows'
>>> os.getcwd()
'C:\\Python34'
```

- ✓ Since C:\Python34 was the working directory when os.path.abspath() was called, the “single-dot”

folder represents the absolute path 'C:\\Python34'.

- ✓ Calling os.path.dirname(path) will return a string of everything that comes before the last slash in the path argument.
- ✓ Calling os.path.basename(path) will return a string of everything that comes after the last slash in the path argument.



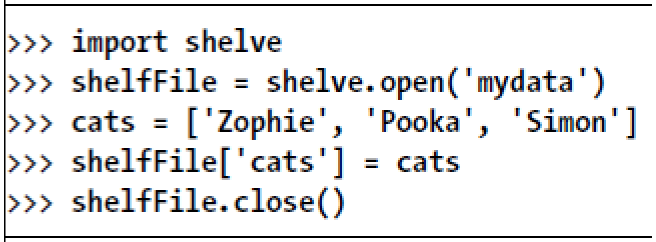
Ex:

```
>>> path = 'C:\\Windows\\System32\\calc.exe'
>>> os.path.basename(path)
'calc.exe'
>>> os.path.dirname(path)
'C:\\Windows\\System32'
```

2a)	Explain saving of variables using shelve module.	5	CO3	L2

Program : 4 marks Explanation- 1 Mark

- ✓ The variables can be saved in Python programs to binary shelf files using the shelve module.
- ✓ This helps the program to restore data to variables from the hard drive.
- ✓ The shelve module will let us add Save and Open features to your program.
- ✓ Example:



```
>>> import shelve
>>> shelfFile = shelve.open('mydata')
>>> cats = ['Zophie', 'Pooka', 'Simon']
>>> shelfFile['cats'] = cats
>>> shelfFile.close()
```

2b)	Write a Program that creates a class called Time with attributes hour, minute and second. Write the following functions. i)A function to read the attributes ii)A function to add two time objects and print the time in format	5	CO4	L3

Program- 5 marks(2.5 *2=5 Marks)

As another example of a user-defined type, we'll define a class called `Time` that records the time of day. The class definition looks like this:

```
class Time(object):
    """Represents the time of day.

    attributes: hour, minute, second
    """
```

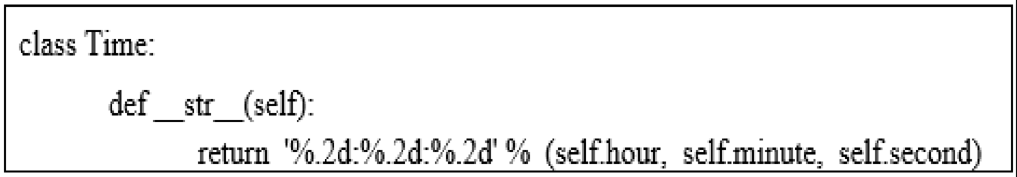
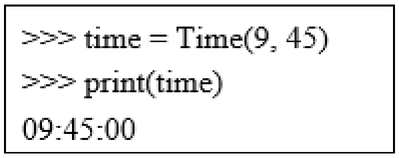
We can create a new `Time` object and assign attributes for hours, minutes, and seconds:

```
time = Time()
time.hour = 11
time.minute = 59
time.second = 30

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

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

    if sum.second >= 60:
        sum.second -= 60
        sum.minute += 1

    if sum.minute >= 60:
        sum.minute -= 60
```

	`        sum.hour += 1`
	
	`    return sum`			

3(a)	Illustrate the concept of modifier function with Python code.
	Explanation: 1 Mark Program-4 marks
	Modifiers: It modifies the objects passed to it as arguments and it has effect.
	
	Ex: In the below example the object start has been passed as parameters and it has been changed by adding seconds to it. Hence, it is a modifier.
	`def increment(time, seconds):`
	`    time.second += seconds`
	
	`    if time.second >= 60: time.second -= 60:`
	`        time.minute += 1`
	
	`    if time.minute >= 60: time.minute -= 60:`
	`        time.hour += 1`
	
	`>>> start  = Time()`
	`>>> start.hour  = 9`
	`>>> start.minute  = 45`
	`>>> start.second  =   0`
	
	`>>> done = increment(start, 10)`
	`>>> print_time(done)`
	`9:45:10`
		5	CO4	L3

3(b)	Explain __init__ and __str__ method with an example Python Program.
	2.5*2 = 5 Marks
	Explanation : 1 marks program 1.5 marks 2.5*2=5 marks
	
	__init_: The init method (short for “initialization”) is a special method that gets invoked when an object is instantiated. The parameters of__init to have the same names as the attributes.
	Ex:
	`class Time:`
	`    def  __init__(self, hour=0, minute=0, second=0):`
	`        self.hour = hour`
	`        self.minute = minute`
	`        self.second = second`
	
	`class Time:`
	`    def print_time(self):`
	`        print('%.2d:%.2d:%.2d'  %  (self.hour,  self.minute,  self.second))`
	
	If function is called with three arguments then, all the values will be overridden.
		5	CO4	L2

	__str_: It is a special method which returns a string
	
	representation of an object. Ex:
	
	`class Time:`
	`    def __str__(self):`
	`        return '%.2d:%.2d:%.2d' % (self.hour, self.minute, self.second)`
	
	When we print an object, Python invokes the str method:
	
	`>>> time = Time(9, 45)`
	`>>> print(time)`
	`09:45:00`			

4 (a)	What is class? How do we define a class in python? How to instantiate the class and how class members are accessed? **Explanation 2.5 Marks Full Program 2.5 marks** ✓ Class is a abstract data type which can be defined as a template or blueprint that describes the behavior / state that the object of its type support. We define class as follows: ✓ The header indicates that the new class is called Point. ✓ The body is a docstring that explains what the class is for. You can define variables and methods inside a class *definition*. ✓ The process of creating this object is called instantiation ✓ Class can be used to create new object instances (instantiation) of that class. The class members are accessed using dot operator as shown below:	5	CO4	L3

```
class Point:
    """ This is a class Point representing
     a coordinate point
    """

def read_point(p):
    p.x=float(input("x coordinate:"))
    p.y=float(input("y coordinate:"))

def print_point(p):
    print("(%g,%g)"%(p.x, p.y))

def distance(p1,p2):
    d=math.sqrt((p1.x-p2.x)**2+(p1.y-p2.y)**2)
    return d

p1=Point()                    #create first object
print("Enter First point:")
read_point(p1)                #read x and y for p1

p2=Point()                    #create second object
print("Enter Second point:")
read_point(p2)                #read x and y for p2

dist=distance(p1,p2)          #compute distance
print("First point is:")
print_point(p1)               #print p1
print("Second point is:")
print_point(p2)               #print p2

print("Distance is: %g" %(distance(p1,p2)))    #print d
```

(b)	Explain operator overloading with example.	5	CO2	L2

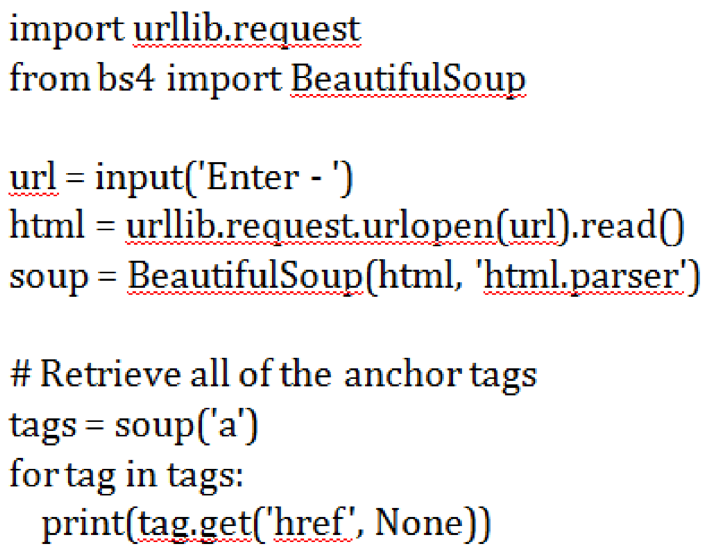
Explanation 1.5 Marks Program 3.5 Marks

<u>Operator Overloading:</u> Ability of an existing to work on user defined data type. It is a polymorphic nature of any object oriented programming. Basic operators like +, -, * can b overloaded. Here, the behavior of an operator is changed like + so it works with a user defined type.

<u>Ex:</u> + → _ _ add_ _ - → _ _sub_ _ * → _ _ mul_ _

```
class A:
    def init (self, a):
        self.a = a

    def __add_(self, o):
        return self.a + o.a

    def __sub_(self, o):
        return self.a - o.a

    def __mul_(self, o):
        return self.a * o.a

ob1 = A(1) ob2 = A(2)
ob3 = A("Python")
ob4 = A("Applications")
print(ob1 + ob2)
print(ob1 - ob2)
print(ob1 * ob2)
print(ob3 + ob4)
```

	Output:
	`  3`
	` -1`
	`  2`
	PythonApplications			

5 (a)	Write a python program to extract all the hyperlinks from the given URL using BeautifulSoup and urllib request Module.
	Program 5 Marks
	
	`import urllib.request`
	`from bs4 import BeautifulSoup`
	
	`url = input('Enter - ')`
	`html = urllib.request.urlopen(url).read()`
	`soup = BeautifulSoup(html, 'html.parser')`
	
	`# Retrieve all of the anchor tags`
	`tags = soup('a')`
	`for tag in tags:`
	`   print(tag.get('href', None))`
		5	CO4	L3

(b)	Explain how Exceptions can be raised with example code. Also write the difference between Exceptions and Assertions.
	Exceptions are raised with a raise statement. In code, a raise statement consists of the following: **[2.5 marks= 1 mark explanation Program 1.5 Marks]**
	• The raise keyword
	• A call to the Exception() function
	• A string with a helpful error message passed to the Exception() function
	• `>>> raise Exception('This is the error message.')`
	`>>> import traceback`
	`>>> try:`
	`...        raise Exception('This is the error message.')`
	`except:`
	`...        errorFile = open('errorInfo.txt', 'w')`
	`...        errorFile.write(traceback.format_exc())`
	`...        errorFile.close()`
	`...        print('The traceback info was written to errorInfo.txt.')`
		5	CO3	L2

[2.5 marks= 1 mark explanation Program 1.5 Marks]

- ✓ **<u>Assertion:</u>** An assertion is a sanity check to make sure the code isn't doing something obviously wrong. This is to verify that logic is impemented properly or not.
- ✓ If the sanity check fails, then an AssertionError exception is raised.
- ✓ In python, an assert statement consists of the following:
  1. The assert keyword
  2. A condition (that is, an expression that evaluates to True or False)
  3. A comma
  4. A string to display when the condition is False
- Assertions are for programmer errors, not user errors. Assertions should only fail while the program is under development; a user should never see an assertion error in a finished program.

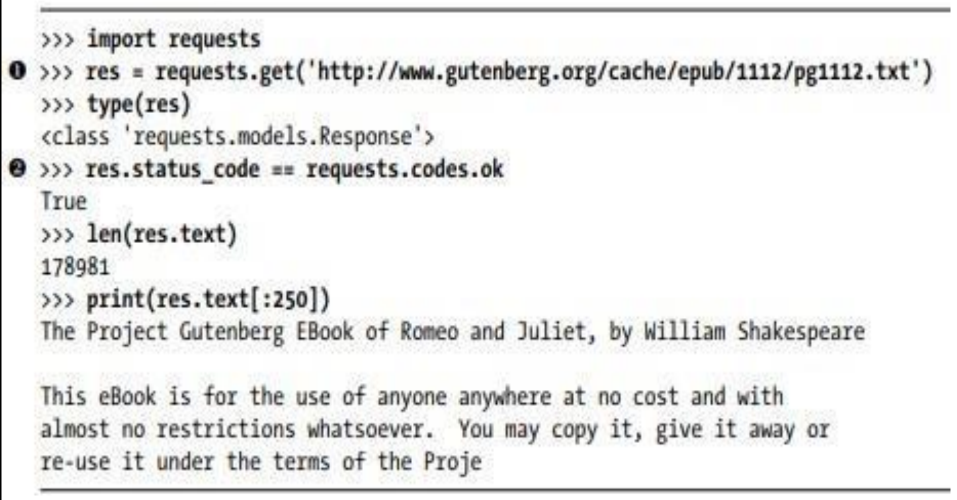
```
>>> ages = [26, 57, 92, 54, 22, 15, 17, 80, 47, 73]
>>> ages.reverse()   # instead of sort reverse method is called
>>> ages
[73, 47, 80, 17, 15, 22, 54, 92, 57, 26]
>>> assert ages[0] <= ages[-1] # Assert that the first age is <= the last age.
```

6 a)	How do we download a file in the web page and save it to hard drive using request module? Explain with program.	5	CO5	L3

Program [5 marks]

<u>Downloading files from the Web with the requests module:</u>

- ➢ The requests module lets us easily download files from the Web without having to worry about complicated issues such as network errors, connection problems, and data compression.
- ➢ **Ex:**



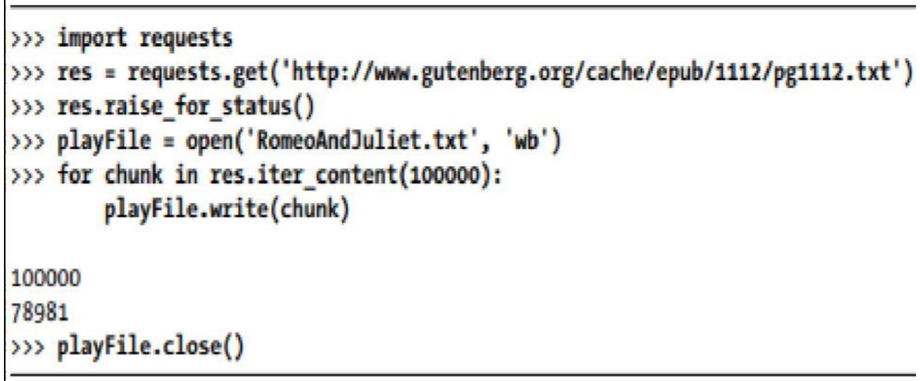
```
>>> import requests
❶ >>> res = requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt')
>>> type(res)
<class 'requests.models.Response'>
❷ >>> res.status_code == requests.codes.ok
True
>>> len(res.text)
178981
>>> print(res.text[:250])
The Project Gutenberg EBook of Romeo and Juliet, by William Shakespeare

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Proje
```

<u>Saving downloaded files to the hard drive</u>

- ➢ We can save the web page to a file on our hard drive with the standard open() function and write() method.

- We must open the file in write binary mode by passing the string 'wb' as the second argument to open().
- Even if the page is in plaintext, we need to write binary data instead of text data in order to maintain the Unicode encoding of the text.

Ex:



```
>>> import requests
>>> res = requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt')
>>> res.raise_for_status()
>>> playFile = open('RomeoAndJuliet.txt', 'wb')
>>> for chunk in res.iter_content(100000):
        playFile.write(chunk)

100000
78981
>>> playFile.close()
```

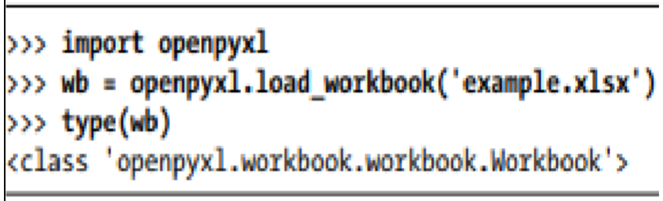
(b) Write a note on the following by demonstrating with code snippet. 5 CO5 L3

i)Opening Excel documents with openpyxl.
ii) Getting Sheets from the Workbook.
iii) Getting Cells, Rows and Columns from the Sheets.

Explanation 2 Marks+ code snippet for each subdivision 1 mark= 5 marks

<u>Opening Excel documents with openpyxl.</u>

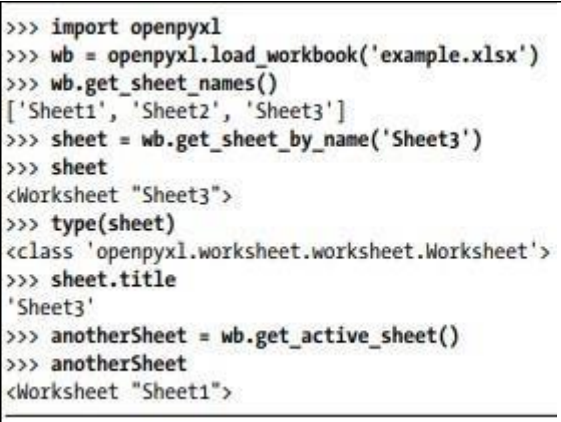
- ✓ Once we've imported the openpyxl module, we'll be able to use the openpyxl .load_workbook() function.
- ✓ The openpyxl.load_workbook() function takes in the filename and returns a value of the workbook data type.



```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('example.xlsx')
>>> type(wb)
<class 'openpyxl.workbook.workbook.Workbook'>
```

<u>Getting Sheets from the Workbook.</u>

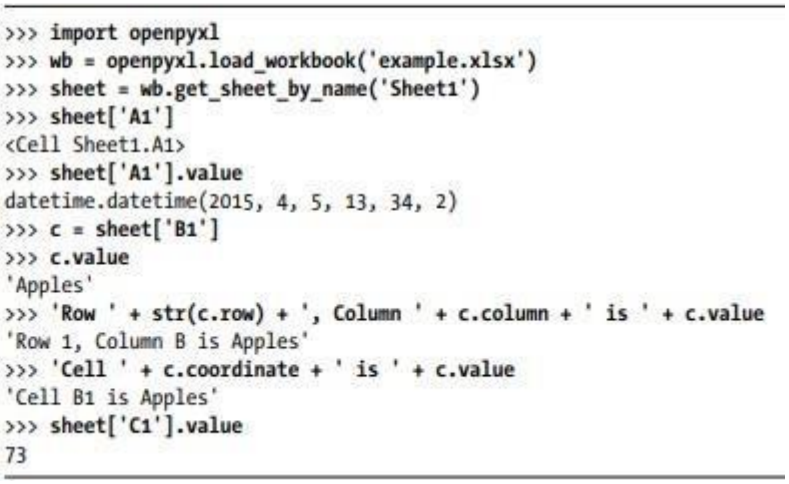
- ✓ We can get a list of all the sheet names in the workbook by calling the get_sheet_names() method.
- ✓ Each sheet is represented by a Worksheet object, which we can obtain by passing the sheet name string to the get_sheet_by_name() workbook method.
- ✓ We can call the get_active_sheet() method of a Workbook object to get the workbook's active sheet.



```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('example.xlsx')
>>> wb.get_sheet_names()
['Sheet1', 'Sheet2', 'Sheet3']
>>> sheet = wb.get_sheet_by_name('Sheet3')
>>> sheet
<Worksheet "Sheet3">
>>> type(sheet)
<class 'openpyxl.worksheet.worksheet.Worksheet'>
>>> sheet.title
'Sheet3'
>>> anotherSheet = wb.get_active_sheet()
>>> anotherSheet
<Worksheet "Sheet1">
```

Getting Cells, Rows and Columns from the Sheets.

- ✓ Once we have a Worksheet object, we can access a Cell object by its name.
- ✓ Cell objects also have row, column, and coordinate attributes that provide location information for the cell.
- ✓ The row attribute gives us the integer 1, the column attribute gives us 'B', and the coordinate attribute gives us 'B1'.
- ✓ **Ex:**



```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('example.xlsx')
>>> sheet = wb.get_sheet_by_name('Sheet1')
>>> sheet['A1']
<Cell Sheet1.A1>
>>> sheet['A1'].value
datetime.datetime(2015, 4, 5, 13, 34, 2)
>>> c = sheet['B1']
>>> c.value
'Apples'
>>> 'Row ' + str(c.row) + ', Column ' + c.column + ' is ' + c.value
'Row 1, Column B is Apples'
>>> 'Cell ' + c.coordinate + ' is ' + c.value
'Cell B1 is Apples'
>>> sheet['C1'].value
73
```

HoD Signature　　　　　　CCI signature　　　　　　Course Instructor