Internal Assessment Test 2 – Dec 2021

Sub:	Application Development using Python					Sub Code:	18CS55	Branch:	CSE
Date:	20-12-2021	Duration:	90 mins	Max Marks:	50	Sem/Sec:	A, B & C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	Discuss keys(), values(), items(), get() and setdefault() dictionary methods with examples	[10]	CO2	L2

► To check if key exists before accessing the value.
► get() : takes 2 arguments
► Key of the value to retrieve
► Fallback value to return if the key does not exist.

```
>>> items={'pen':5, 'stapler':1, 'marker': 3}
>>>print("I have",items.get('pen',0), 'pens')
I have 5 pens
>>>print("I have",items.get('pencil',0), 'pencils')
I have 0 pencils
```

Keys(), values(), items()

► Returns list-like values
► These lists returned are not true lists: They cannot be modified and do not have an append() method.
► Their data types are dict_keys, dict_values, dict_items
► Can be used in for loops. Create a dictionary of items. Ask user to enter the name of item and the number of it. If item exists, add the number to the existing count. If item does not exist, create new item with the number

```
items={'pen':5, 'stapler':1, 'marker': 3}
while True:
print("Enter item(blank to quit):")
item = input()
if item==":
break
number = int(input("Number:"))
items[item]=items.get(item,0)+number
print(items)
```

Sample output

```
#items={'pen':5, 'stapler':1, 'marker': 3}
python items_get.py
Enter item(blank to quit):
stapler Number:3
Enter item(blank to quit): eraser Number:4
Enter item(blank to quit): {'pen': 5, 'stapler': 4, 'marker': 3, 'eraser': 4}
>>> codes = {'red':'#FF0000','sky-blue':'#87CEEB','coral':'#FF7F50'}
>>>codes.values() dict_values(['#FF0000', '#87CEEB', '#FF7F50'])
>>>codes.keys() dict_keys(['red', 'sky-blue', 'coral'])
>>> type(codes.values())
>> type(codes.keys())
>>> type(codes.items())
>>>codes.items()
dict_items([('red', '#FF0000'), ('sky-blue', '#87CEEB'), ('coral', '#FF7F50')])
>>> list(codes.keys()) ['red', 'sky-blue', 'coral']
```

#use multiple assignment
>>> for color,hexa in codes.items():
print('Color:',color,'Value:',hexa) Color: red Value:
#FF0000 Color: sky-bule Value:
#87CEEB Color: coral Value:
#FF7F50

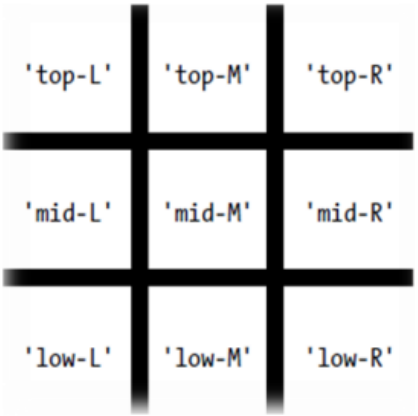
2 (a)	Develop a program to accept a sentence from the user and to display the longest word of the sentence along with its length.	[05]	CO2	L3

```
sent=input('Enter sentence')
long=max(sent.split(),key=len)
print('The longest word is:',long)
print('The length of longest word is: ',len(long))
```

```
Enter sentence I study at cmrit college
The longest word is: college
The length of longest word is:  7
```

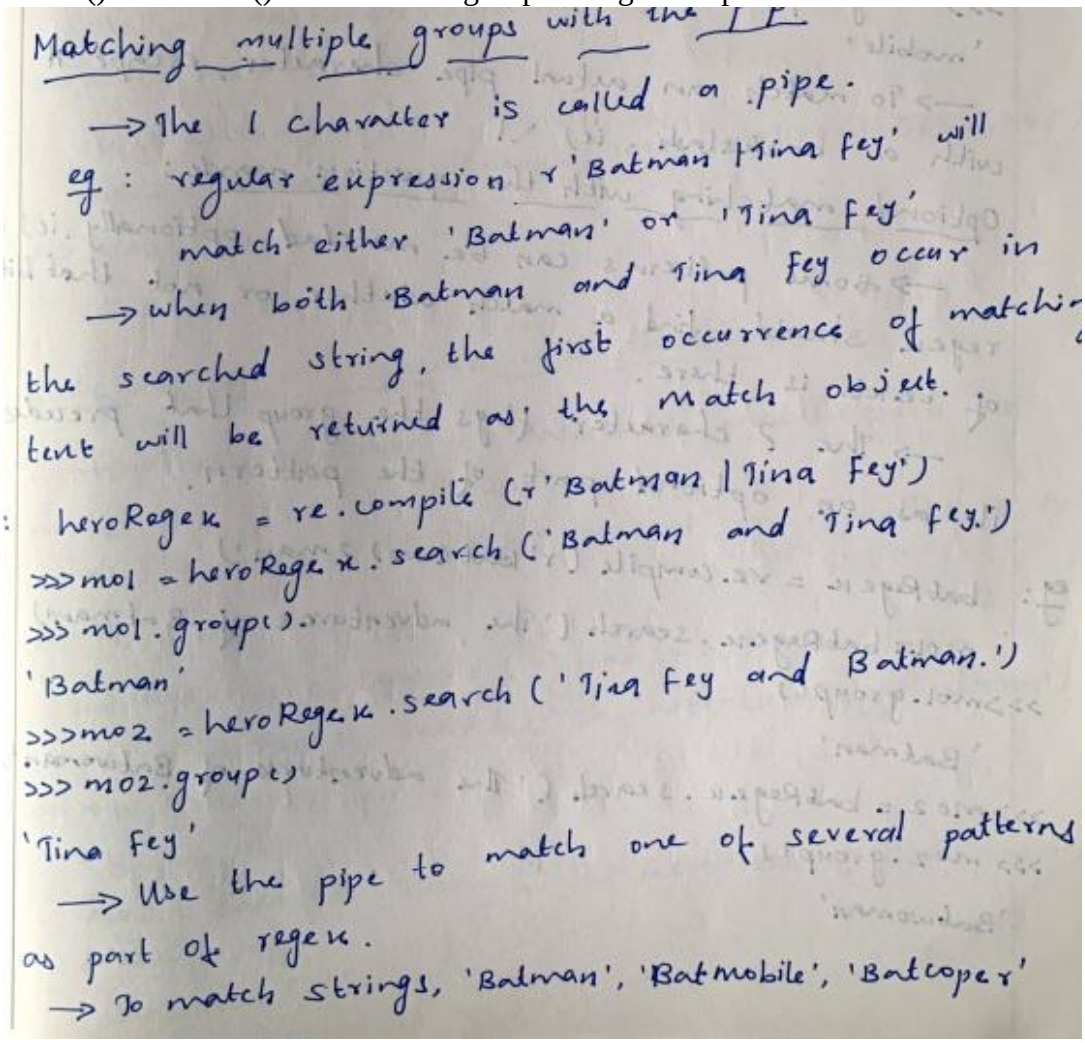
(b)	Create a function to print out a blank tic-tac-toe board. ► looks like a large hash symbol (#) ► nine slots ► Each can contain an X, an O, or a blank. ► To represent using a dictionary, each slot can be given a string-value key.	[05]	CO2	L3

```
theBoard = {'top-L': ' ', 'top-M': ' ', 'top-R': ' ', 'mid-L': ' ',
            'mid-M': ' ', 'mid-R': ' ', 'low-L': ' ', 'low-M': ' ', 'low-R': ' '}
def printBoard(board):
 print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
 print('-+-+-')
 print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
 print('-+-+-')
 print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])
printBoard(theBoard)
```



3 (a)	What is the use of | (pipe) in regular expression? What is difference between	[05]	CO3	L3

search() and findall() methods with groups of regular expression?



Matching multiple groups with the |

→ The | character is called a pipe.

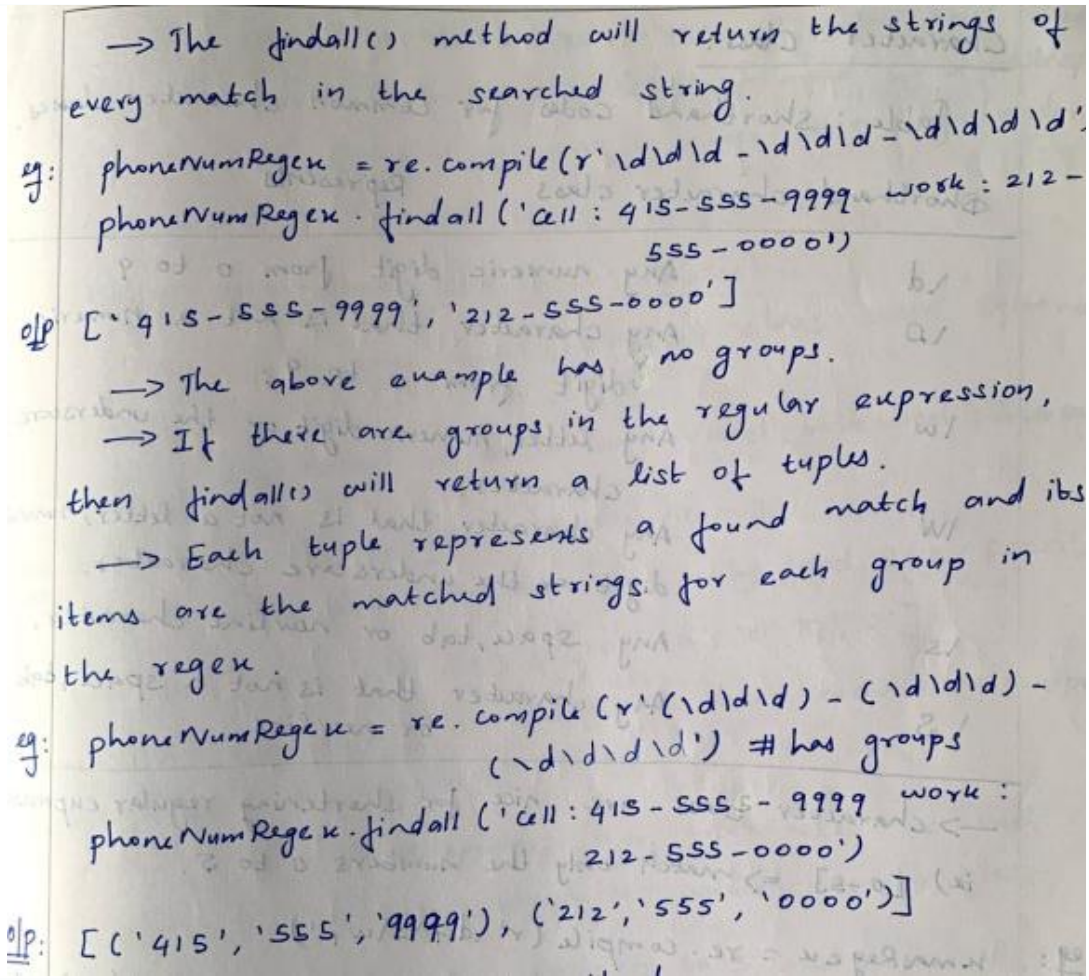
eg : regular expression r'Batman|Tina Fey' will match either 'Batman' or 'Tina Fey'

→ when both Batman and Tina Fey occur in the searched string, the first occurrence of matching text will be returned as the match object.

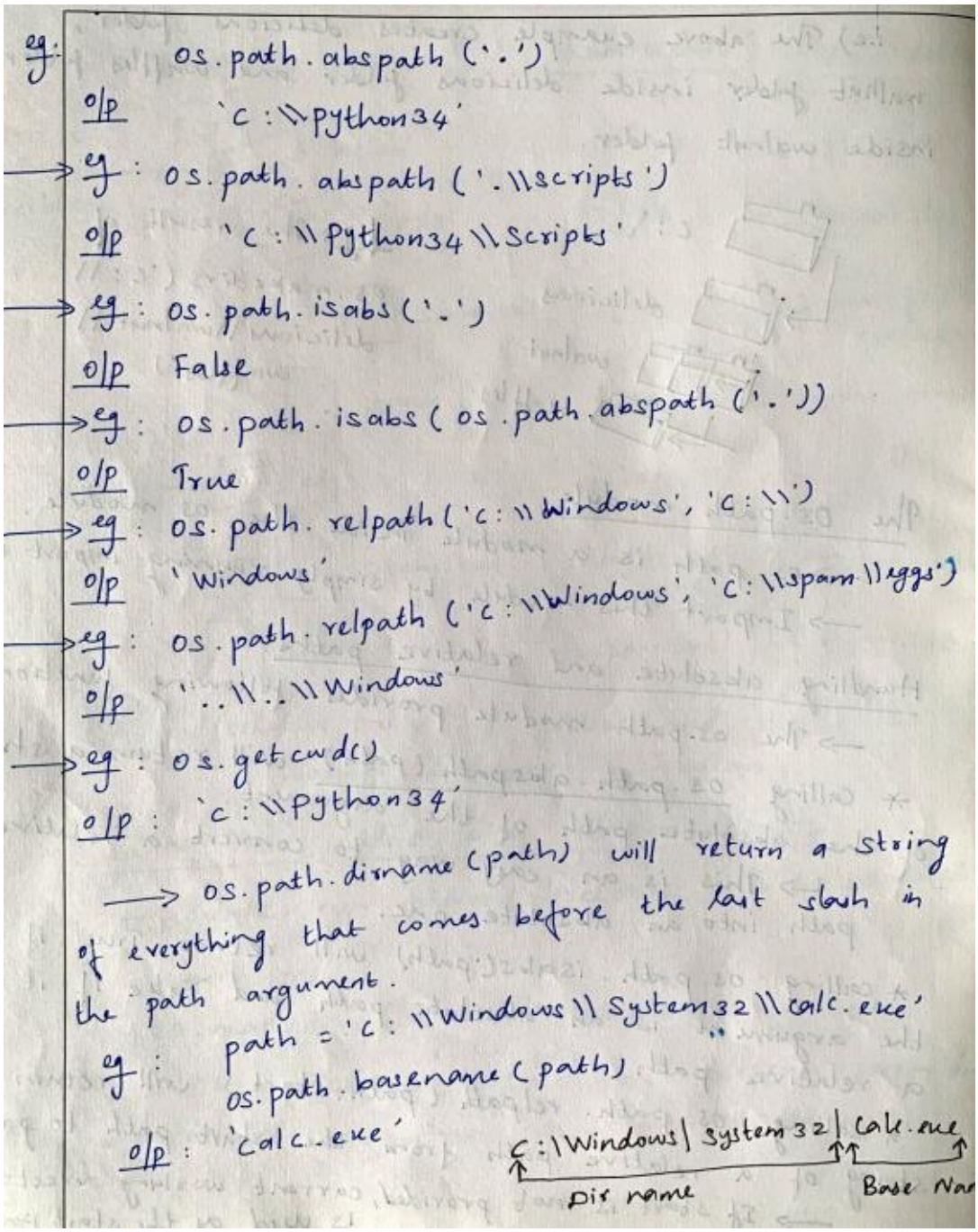
```
heroRegex = re.compile(r'Batman|Tina Fey')
>>> mo1 = heroRegex.search('Batman and Tina Fey.')
>>> mo1.group()
'Batman'
>>> mo2 = heroRegex.search('Tina Fey and Batman.')
>>> mo2.group()
'Tina Fey'
```

→ Use the pipe to match one of several patterns as part of regex.

→ To match strings, 'Batman', 'Batmobile', 'Batcopter'



(b)	Explain abspath, basename, relpath, getsize and exists of OS path Module with examples.	[05]	CO3	L2	



4 (a) Write a python program to create a folder PYTHON and under the hierarchy 3 files file1,file2 and file3.write the content in file1 as "VTU" and in file2 as "UNIVERSITY" and file3 content should be by opening and merge of file1 and file2. Check out the necessary condition before write file3. [05] CO3 L3

```
import os
os.mkdir("Python")
os.chdir("Python")
myfile1=open('file1','w')
myfile2=open('file2','w')
myfile1.write("VTU")
myfile2.write("UNIVERSITY")
myfile1.close()
myfile2.close()
```

```
myfile3=open('file3','a') #opening file3 in append mode
myfile1=open('file1').read()
myfile2=open('file2').read()
myfile3.write(myfile1)
myfile3.write(myfile2)# appending contents of file2 (not overwriting)
myfile3.close() #answer is completed
contents_file3=open('file3').read()
print(contents_file3)
  #Expected output: VTUUNIVERSITY"
```

(b)	Explain the regex re.IGNORECASE, re.DOTALL and re.VERBOSE with examples.	[05]	CO3	L2

To use re.VERBOSE to write comments in your regular expression but also want to use re.IGNORECASE to ignore capitalization? Unfortunately, the re.compile() function takes only a single value as its second argument. You can get around this limitation by combining the re.IGNORECASE, re.DOTALL, and re.VERBOSE variables using the pipe character (|), which in this context is known as the bitwise or operator.

So if you want a regular expression that's case-insensitive and includes newlines to match the dot character, you would form your re.compile() call like this:

```
>>>someRegexValue = re.compile('foo', re.IGNORECASE | re.DOTALL)
```

All three options for the second argument will look like this:

```
>>>someRegexValue = re.compile('foo', re.IGNORECASE | re.DOTALL | re.VERBOSE)

phoneRegex = re.compile(r'''( (\d{3}|\(\d{3}\))?
# area code (\s|-|\.)?
# separator \d{3}
# first 3 digits (\s|-|\.)
# separator \d{4}
# last 4 digits (\s*(ext|x|ext.)\s*\d{2,5})?
# extension )''', re.VERBOSE | re.IGNORECASE | re.DOTALL |)
```

5 (a)	Explain functions of shutil module with examples.	[05]	CO3	L2

► Used to save variables in your Python programs to binary shelf files
► program can restore data to variables from the hard drive.
► Shelve module : allows you to Save and Open features to your program.
► To read and write data, call shelve.open() and pass file name.
► Notice that 3 files, 'mydata.bak', 'mydata.dat', 'mydata.dir' get created in the current working directory.

(On OS X, mydata.db is created)

```
>>> import shelve
>>>shelfFile = shelve.open('mydata')
>>> topics=['read files','writefiles','append']
>>>shelfFile['topics']=topics
>>>shelfFile.close()
>>>os.listdir() mydata.bak mydata.dat mydata.dir
>>> sf= shelve.open('mydata')
>>> list(sf.keys()) ['topics']
>>> list(sf.values()) ['read files','writefiles','append']
```

(b)	Write a program to find five characters which starts with 'a' and ends with 'z'. Print	[05]	CO3	L3

successful if pattern matches the string.

```
import re
while True:
    str1=input("Enter string : (nomore to end) ")
    if str1 == 'nomore':
        break
    if len(str1)== 5 and re.findall("^a.*z$", str1):
        print("Yes, the string has 5 characters a...z :",str1)
    else:
        print("Not taken")
```

```
Enter string : (nomore to end)  asdfz
Yes, the string has 5 characters a...z : asdfz
Enter string : (nomore to end)  nomore
```

6 (a) Explain compressing files. Write code snippet for creating, reading and extracting zip files. [05] CO3 L2

► Compressing a file reduces its size, which is useful when transferring it over the Internet.
► namelist() method that returns a list of strings for all the files and folders contained in the ZIP file.
► Can be passed to getinfo() of ZipFile
► To retrieve information about the particular file Reading ZIP files
>>> import zipfile, os
>>>os.chdir('C:\\Users\\Admin\\Pictures')
>>>zp = zipfile.ZipFile('sc.zip')
>>>zp.namelist()
>>>spInfo = zp.getinfo('sc/world.png')
>>>spInfo.file_size 6546
>>>spInfo.compress_size 3096 ZipFile : holds information about entire object ZipInfo : holds information about particular file. Extracting from ZIP files
► extractall() method for ZipFile objects extracts all the files and folders onto current working directory.
► Pass a folder name to extract files into a folder other that current working directory. ► If folder is not found, it will be created.
► extract() : extracts a single file from the ZIP file.
► Optional second argument : folder.
>>>zp = zipfile.ZipFile('sc.zip')
>>>zp.extractall()
>>>zp.extractall('test')
>>>zp.namelist() [….] # lists all the names of the files in a list
>>>zp.extract('sc/world.png') # extracts only world.png
>>>zp.extract('sc/world.png','test_single') Creating and adding ZipFiles
► open the ZipFile object in write mode by passing 'w'
► Similar to opening file to write
► pass a path to the write() method of ZipFile object
► 2 nd argument is the compression type
► The algorithm to be used for compression
► zipfile.ZIP_DEFLATED works well for all file types
► To add files to an existing zip file pass the 'a' mode.
>>>newZip = zipfile.ZipFile('sc_new.zip','w')
>>>newZip.write('download.jpg',compress_type = zipfile.ZIP_DEFLATED)

```
>>>newZip.write('code1.png',compress_type = zipfile.ZIP_DEFLATED)
>>>newZip.close()
```

(b) Explain logging module and its basicConfig() functions [05] | CO3 | L2

Python has a built-in module logging which allows writing status messages to a file or any other output streams. Logging is a means of tracking events that happen when some software runs. The software's developer adds logging calls to their code to indicate that certain events have occurred.

```
#importing module
import logging

#Create and configure logger
logging.basicConfig(filename="logfile.log",format='%(asctime)s %(message)s',filemode='w')

#Creating an object
logger=logging.getLogger()

#Setting the threshold of logger to DEBUG
logger.setLevel(logging.DEBUG)

#Test messages
logger.debug("Harmless debug Message")
logger.info("Just an information")
logger.warning("Its a Warning")
logger.error("Did you try to divide by zero")
logger.critical("Internet is down")
```