Roll No.				

Internal Assessment Test 1 – Nov. 2023

Sub:	Introduction to Python Programming--**Solutions and Scheme**					Sub Code:	BPLCK105B	Branch:	Chemistry Cycle	
Date:	6-12-2023	Duration:	90 min's	Max Marks:	50	Sem / Sec:	I / Chemistry Cycle			OBE

	Answer any FIVE FULL QUESTIONS	MARKS	CO	RBT
1 (a)	**What is list? Explain append( ), insert( ), remove() and sort( ) methods with Example.** **Correct definition/description [2 marks]** **Example [1*4= 4 marks]** A list is a value that contains multiple values in an ordered sequence. List begins with an opening square bracket and ends with a closing square bracket, [ ]. Values inside the list are also called items. Items are separated with commas. Spam= ['cat', 'bat', 'rat', 'elephant'] Print(Spam) o/p: ['cat', 'bat', 'rat', 'elephant'] **APPEND**: To add new values to a list, use the append() and insert() methods. append() method call adds the argument to the end of the list. E.g. spam = ['cat', 'dog', 'bat'] spam.append('moose') print(spam) **output:** **['cat', 'dog', 'bat', 'moose']** **INSERT**: The insert() method can insert a value at any index in the list. The first argument to insert() is the index for the new value, and the second argument is the new value to be inserted. E.g. spam = ['cat', 'dog', 'bat'] spam.insert(1, 'chicken') print(spam) **output**: **['cat', 'chicken', 'dog', 'bat']** **REMOVE**: The remove() method removes a value from the list. spam = ['cat', 'bat', 'rat', 'elephant'] spam.remove('bat') print(spam) **output**:	[6]	CO2	L2

	['cat', 'rat', 'elephant'] Attempting to delete a value that does not exist in the list will result in a ValueError error. **SORT**: Lists of number values or lists of strings can be sorted with the sort() method. For example spam = ['ants', 'cats', 'dogs', 'badgers', 'elephants'] spam.sort() print(spam) **output**: **['ants', 'badgers', 'cats', 'dogs', 'elephants']**			

(b)	**Explain the use of in and not in operators in list with suitable examples.**
	
	□ Explanation [2 marks]
	□ Example with output [2 marks]
	
	The in and not in operators are used to check whether a value is or isn't in a list. Like other operators, in and not in are used in expressions and connect two values: a value to look for in a list and the list where it may be found. These expressions will evaluate to a Boolean value
	'howdy' in ['hello', 'hi', 'howdy', 'heyas']
	True
	spam = ['hello', 'hi', 'howdy', 'heyas']
	'cat' in spam
	False
	'howdy' not in spam
	False
	'cat' not in spam
	True
	
	EXAMPLE PROGRAM:
	
	myPets = ['Tom', 'Heyas', 'Fat-tail']
	print('Enter a pet name:')
	name = input()
	if name not in myPets:
	print('I do not have a pet named ' + name)
	else:
	print(name + ' is my pet.')
	Output:
	Enter a pet name:
	Footfoot
	I do not have a pet named Footfoot
		[4]	CO2	L2

2 (a)	**Read N numbers from the console and create a list. Develop a program to compute Mean, Variance and Standard Deviation with suitable messages.** **□ Correct logic [3 marks]** **□ Correct syntax [2 marks]** n = int(input("Enter the range of value to be read :")) list = [] for i in range(0,n): print("Enter number",i) a=int(input())	[5]	CO2	L3

```
    list.append(a)
mean = sum(list)/n
print("The mean of all the list number is:”,mean)
list1 = []
for i in list:
     b= (i-mean)**2
     list1.append(b)
variance= sum(list1)/n
print("The variance of all the list number is:”,variance)
print("The standard deviation of all the list number is:”,variance**0.5 )
```

output:

```
Enter range of values to be read: 5
Enter number 1: 1
Enter number 2: 2
Enter number 3: 3
Enter number 4: 4
Enter number 5: 5
The mean of all the list number is: 3
The variance of all the list number is: 2
The standard deviation of all the list number is: 1.414
```

(b)	**How is tuple different from list and which function is used to convert list to tuple? Explain in detail.**	[5]	CO2	L2

- **Difference between tuple and list [3 marks]**
- **converting list to tuple with example [2 marks]**

Tuple datatype, is an immutable form of the list data type. The tuple data type is almost identical to the list data type, except in two ways. First, tuples are typed with parentheses, ( ), instead of square brackets, [ ].

For example,

```
eggs = ('hello', 42, 0.5)
eggs[0]
'hello'
eggs[1:3]
(42, 0.5)
len(eggs)
3
```

But the main way that **tuples are different from lists** is that tuples, like strings, are immutable. Tuples cannot have their values modified, appended, or removed.

```
eggs = ('hello', 42, 0.5)
eggs[1] = 99
```

output

TypeError: 'tuple' object does not support item assignment

If we have only one value in tuple, we can indicate this by placing a trailing comma after the value inside the parentheses. Otherwise, Python will think we have just typed a value inside regular parentheses. The comma is what lets Python know this is a tuple value.

```
type(('hello',))
<class 'tuple'>
type(('hello'))
<class 'str'>
```

Converting Types with the list() and tuple() Functions:

The functions list() and tuple() will return list and tuple versions of the values passed to them.

	tuple(['cat', 'dog', 5])
	('cat', 'dog', 5)
	list(('cat', 'dog', 5))
	['cat', 'dog', 5]
	list('hello')
	['h', 'e', 'l', 'l', 'o']			

3 (a)	**What is dictionary in Python? Explain get( ) and setdefault( ) methods with example.** **Correct definition/description [2 marks]** **get() and setdefault() Explanation [3 marks]** **The dictionary data type:** A dictionary is a collection of many values. But unlike indexes for lists, indexes for dictionaries can use many different data types, not just integers. Indexes for dictionaries are called keys, and a key with its associated value is called a key-value pair. In code, a dictionary is typed with braces, {} e.g. myCat = {'size': 'fat', 'color': 'gray', 'disposition': 'loud'} This assigns a dictionary to the myCat variable. This dictionary's keys are 'size', 'color', and 'disposition'. The values for these keys are 'fat', 'gray', and 'loud', respectively. access these values through their keys: myCat['size'] **'fat'** 'My cat has ' + myCat['color'] + ' fur.' **'My cat has gray fur.'** **The get() Method** Dictionaries have a get() method that takes two arguments: the key of the value to retrieve and a fallback value to return if that key does not exist. E.G., picnicItems = {'apples': 5, 'cups': 2} 'I am bringing ' + str(picnicItems.get('cups', 0)) + ' cups.' **'I am bringing 2 cups.'** 'I am bringing ' + str(picnicItems.get('eggs', 0)) + ' eggs.' **'I am bringing 0 eggs.'** Because there is no 'eggs' key in the picnicItems dictionary, the default value 0 is returned by the get() method. **The setdefault() Method** The first argument passed to the method is the key to check for, and the second argument is the value to set at that key if the key does not exist. If the key does exist, the setdefault() method returns the key's value. spam = {'name': 'Tom', 'age': 5} spam.setdefault('color', 'black') print(spam) **{'color': 'black', 'age': 5, 'name': 'Tom'}** spam.setdefault('color', 'white') print(spam) **{'color': 'black', 'age': 5, 'name': 'Tom'}** The first time setdefault() is called, the dictionary in spam changes to {'color': 'black', 'age': 5, 'name': 'Tom'}. The method returns the value 'black' because this is now the value set for the key 'color'. When spam.setdefault('color', 'white') is called next, the value for that key is not changed to 'white' because spam already has a key named 'color'.	[5]	CO2	L2

(b)	**Develop a program to print frequency of each digit with suitable message**.
	
	 Correct logic [3 marks]
	 Correct syntax [2 marks]
	
	`str1=input(“Enter a multidigit number”)`
	`for i in range(0,10):`
	`    if str1.count(str(i))!=0:`
	`          print(“Number of count of”,i,”is”,str1.count(str(i)))`
	output:
	1256236
	Number of count of 1 is 1
	Number of count of 2 is 2
	Number of count of 3 is 1
	Number of count of 5 is 1
	Number of count of 6 is 2
		[5]	CO2	L3

4 (a)	**Write a Python program to find the total size of the text files in the folder ‘C:\\Windows\\System32’**
	
	 Correct logic [3 marks]
	 Correct syntax [2 marks]
	
	`totalSize = 0`
	`for filename in os.listdir('C:\\Windows\\System32'):`
	`    totalSize = totalSize + os.path.getsize(os.path.join('C:\\Windows\\System32', filename))`
	`print(totalSize)`
	output:
	1117846456
		[5]	CO3	L3

(b)	**Explain Join and split methods with Examples.** **Explanation (3 Marks)** **Example(each 1*2= 2 Marks)** The join() method is useful when list of strings that need to be joined together into a single string value. The join() method is called on a string, gets passed a list of strings, and returns a string. The returned string is the concatenation of each string in the passed-in list. ', '.join(['cats', 'rats', 'bats']) **'cats, rats, bats'** 'ABC'.join(['My', 'name', 'is', 'Simon']) **'MyABCnameABCisABCSimon'** join() is called on a string value and is passed a list value. The split() method does the opposite: It’s called on a string value and returns a list of strings. 'My name is Simon'.split() **['My', 'name', 'is', 'Simon']** Pass a delimiter string to the split() method to specify a different string to split upon. 'MyABCnameABCisABCSimon'.split('ABC') **['My', 'name', 'is', 'Simon']**	[5]	CO3	L2
5(a)	**Explain the following with Example code snippet:**	[5]	CO3	L2

	i) isalpha( ) ii) isalnum( ) iii) isdecimal( ) iv) isspace( ) v) istitle( ) **Explanation with Example [1 *5 = 5 marks]** These methods return a Boolean value that describes the nature of the string. Here are some common isX string methods: **isalpha()** returns True if the string consists only of letters and is not blank. ‘Hello’.isalpha() **True** ‘Hello’.isalpha() **False** **isalnum()** returns True if the string consists only of letters and numbers and is not blank. ‘Hello123’.isalnum() **True** ‘Hello 123’.isalnum() **False** **isdecimal()** returns True if the string consists only of numeric characters and is not blank. ‘1234’.isdecimal() **True** **isspace()**returns True if the string consists only of spaces, tabs, and new- lines and is not blank. ‘ ‘.isspace() **True** **istitle()** returns True if the string consists only of words that begin with an uppercase letter followed by only lowercase letters. ‘Hello World’.istitle() **True**			

(b)	**Develop a program to check whether the given number is Armstrong number or not. [Hint: Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself.**
	
	 Correct logic [3 marks]
	 Correct syntax [2 marks]
	num = int(input("Enter a number: "))
	sum = 0
	temp = num
	while temp > 0:
	    digit = temp % 10
	    sum += digit ** 3
	    temp //= 10
	if num == sum:
	    print(num,"is an Armstrong number")
	else:
	    print(num,"is not an Armstrong number")
	OUTPUT:
	Enter a number: 153
	153 is an Armstrong number
		[5]	CO3	L3

6 (a)	**Illustrate with Example function of Opening of a file, reading the contents of file, writing to files.** **Explanation (3 Marks)** **Example(2 Marks)** **Opening Files with the open() Function** To open a file with the open() function, pass it a string path indicating the file wants to open; it can be either an absolute or relative path. The open() function returns a File object. **helloFile = open('C:\\\\Users\\\\your_home_folder\\\\hello.txt')** Read mode is the default mode for files opened in Python. We can explicitly specify the mode by passing the string value 'r' as a second argument to open(). So open ('/Users/Al/hello.txt', 'r') and open('/Users/Al/hello.txt') do the same thing. **Reading the Contents of Files** To read the entire contents of a file as a string value, use the File object's read() method. helloContent = helloFile.read() helloContent **'Hello, world!'** Alternatively, use the **readlines()** method to get a list of string values from the file, one string for each line of text. **Writing to Files** Python allows us to write content to a file in a way similar to how the print() function "writes" strings to the screen. We can't write to a file that opened in read mode, though. Write mode will overwrite the existing file and start from scratch. Pass 'w' as the second argument to open() to open the file in write mode. Append mode, on the other hand, will append text to the end of the existing file. Pass 'a' as the second argument to open() to open the file in append mode. If the filename passed to open() does not exist, both write and append mode will create a new, blank file. After reading or writing a file, call the close() method before opening the file again. baconFile = open('bacon.txt', 'w') baconFile.write('Hello, world!\n') **13** baconFile.close() baconFile = open('bacon.txt', 'a') baconFile.write('Bacon is not a vegetable.') **25** baconFile.close() baconFile = open('bacon.txt') content = baconFile.read() baconFile.close() print(content) **Hello, world!** **Bacon is not a vegetable.**	[5]	CO3	L3

(b)	**Write a python code to determine whether a given string is Palindrome or not?**
	
	Correct logic [3 marks]
	Correct syntax [2 marks]
	
	str1=input("Enter a string: ")
	str2=str1[::-1]
	if(str1.lower()==str2.lower()):
	    print("Entered string is a palindrome")
	else:
	    print("Entered string is not a palindrome")
	Output:
	Enter a string: Madam
	Entered string is a palindrome
		[5]	CO3	L3

7 (a)	**Explain the concept of file path. And also Explain the Difference between absolute and relative path.** **Definition (3 Marks)** **Difference between absolute and relative path(2 marks)** A file has two key properties: a filename and a path. The path specifies the location of a file on the computer. For example, there is a file on my Windows laptop with the filename project.docx in the path C:\Users\Al\Documents. The part of the filename after the last period is called the **file's extension** and a file's type. The filename project.docx is a Word document, and Users, Al, and Documents all refer to folders (also called directories). Folders can contain files and other folders. The C:\ part of the path is the root folder, which contains all other folders. On Windows, the root folder is named C:\ and is also called the C: drive. **Absolute vs. Relative Paths** There are two ways to specify a file path: An absolute path, which always begins with the root folder. A relative path, which is relative to the program's current working Directory. There are also the dot (.) and dot-dot (..) folders. These are not real folders but special names that can be used in a path. A single period ("dot") for a folder name is shorthand for "this directory." Two periods ("dot-dot") means "the parent folder."	[5]	CO4	L2
(b)	**Explain saving variables with Shelve Module.** **Explanation (3 Marks)** **with Example (2 Marks)** Save variables in Python programs to binary shelf files using the shelve module. This way, program can restore data to variables from the hard drive. The shelve module will lets us add Save and Open features. import shelve shelfFile = shelve.open('mydata') cats = ['Tom', 'Simon'] shelfFile['cats'] = cats shelfFile.close()	[5]	CO3	L2

To read and write data using the shelve module, first import shelve. Call shelve.open() and pass it a filename, and then store the returned shelf value in a variable. When program is done, call close() on the shelf value. Here, our shelf value is stored in shelfFile.
After running the previous code, three new files in the current working directory: mydata.bak, mydata.dat, and mydata.dir are created.
These binary files contain the data stored in shelf.

we can use the shelve module to later reopen and retrieve the data from these shelf files.

```
shelfFile = shelve.open('mydata')
shelfFile['cats']
['Tom', 'Simon']
shelfFile.close()
```

Just like dictionaries, shelf values have keys() and values() methods that will return list-like values of the keys and values in the shelf. Since these methods return list-like values instead of true lists, pass them to the list() function to get them in list form.

```
shelfFile = shelve.open('mydata')
list(shelfFile.keys())
['cats']
list(shelfFile.values())
[['Tom', 'Simon']]
shelfFile.close()
```

CI **CCI** **HOD**