Internal Assessment Test 2: Solution – May 2024

Sub:	**Introduction to Python Programming**					Sub code:	BPLCK205B	Branch:	Chemistry Cycle
Date:	23-05-2024	Duration:	90 min's	Max Marks:	50	Sem / Sec:	II / Chemistry Cycle		OBE

		MARKS	CO	RBT
1 (a)	What is a list? Explain the following with the help of example: Append( ), insert( ), remove( ) and sort( ) methods	[6]	CO2	L2

ans:

append(): The append() method call adds the argument to the end of the list.

```
>>> spam = ['cat', 'dog', 'bat']
>>> spam.append('moose')
>>> spam
  ['cat', 'dog', 'bat', 'moose']
```

insert(): this 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.

```
>>> spam = ['cat', 'dog', 'bat']
>>> spam.insert(1, 'chicken')
>>> spam
['cat', 'chicken', 'dog', 'bat']
```

remove(): The remove() method is passed the value to be removed from the list it is called on.

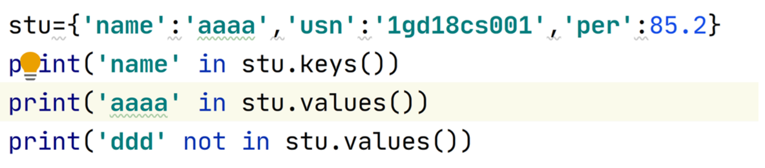
```
>>> spam = ['cat', 'bat', 'rat', 'elephant']
>>> spam.remove('bat')
>>> spam
  ['cat', 'rat', 'elephant']
```

sort(): There are three things you should note about the sort() method.
1. the sort() method sorts the list in place; don't try to return value by writing code like spam = spam.sort().
2. we cannot sort lists that have both number values and string values in them.
3. sort() uses "ASCIIbetical order(upper case)" rather than actual alphabetical order(lower case) for sorting strings.

```
>>> spam = [2, 5, 3.14, 1, -7]
>>> spam.sort()
```

	>>> spam
	[-7, 1, 2, 3.14, 5]
	>>> spam = ['ants', 'cats', 'dogs', 'badgers', 'elephants']
	>>> spam.sort()
	>>> spam
	['ants', 'badgers', 'cats', 'dogs', 'elephants']
	>>> spam.sort(reverse=True)
	>>> spam
	['elephants', 'dogs', 'cats', 'badgers', 'ants']			

(b)
 	
ans:
	How is tuple different from list and which function is used to convert list to tuple? Explain in detail.
	The tuple data type is almost identical to the list data type, except in two ways.
	1. tuples are typed with parentheses, (and), instead of square brackets, [and].
	2. Benefit of using tuples instead of lists is that, because they are immutable and their contents don’t change. Tuples cannot have their values modified, appended, or removed.
	If you have only one value in your tuple, you can indicate this by placing a trailing comma after the value inside the parentheses.
	
	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']
		[4]	CO2	L2

2 (a)
ans:
	Explain the use of in and not in operator in list with suitable examples
	We can determine whether a value is or isn’t in a list with the in and not in 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 and these expressions will evaluate to a Boolean value.
	
	The following program lets the user type in a pet name and then checks to see whether the name is in a list of pets.
	
	
	stu={'name':'aaaa','usn':'1gd18cs001','per':85.2}
	print('name' in stu.keys())
	print('aaaa' in stu.values())
	print('ddd' not in stu.values())
		[5]	CO2	L2

(b)	Write a program to print prime numbers in a given range.	[5]	CO2	L3

ans:

```
num = 11
# If given number is greater than 1
if num > 1:
    # Iterate from 2 to n // 2
    for i in range(2, (num//2)+1):
        # If num is divisible by any number between
        # 2 and n / 2, it is not prime
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

3 (a)	What is a dictionary in Python? Explain get( ) and setdefault( ) methods with example	[5]	CO2	L2

ans:

get():

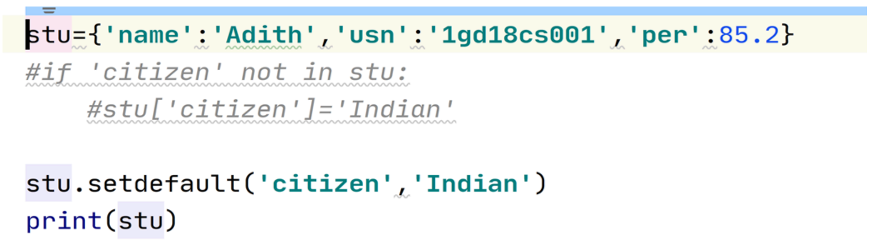
The `get()` method returns the value of the item with the specified key.

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = car.get("model")
print(x)
```

Setdefault() takes 2 arguments:

o The first argument is the key to check for, and

o 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.



```
stu={'name':'Adith','usn':'1gd18cs001','per':85.2}
#if 'citizen' not in stu:
    #stu['citizen']='Indian'

stu.setdefault('citizen','Indian')
print(stu)
```

(b)	Explain the concept of file path. And also explain the Difference between absolute and relative path.	[5]	CO2	L2

ans:

•The os.path.join() function is helpful if you need to create strings for filenames.

•<u>Example:</u>

```
myFiles = [‘accounts.txt’,’details.csv’, ‘invite.docx’]

for filename in myfiles:

  print(os.path.join(“C:\\Users\\asweigrt”, filename))
```

•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 (.) à folder name/ this directory

•dot-dot (..) folders à parent folder

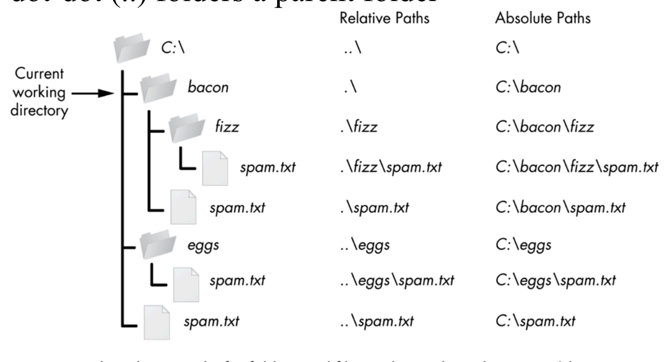


Figure 8-2: The relative paths for folders and files in the working directory C:\bacon

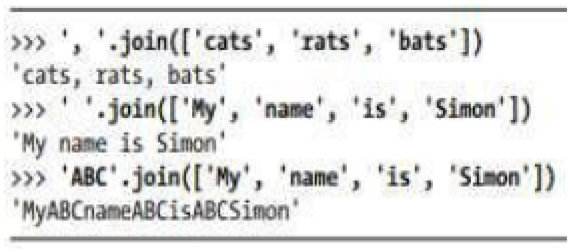
4 (a) Explain join() and split() methods with examples. [5] CO3 L2

ans:

Join()

➢ The join() method is useful when we have a 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'
>>> ' '.join(['My', 'name', 'is', 'Simon'])
'My name is Simon'
>>> 'ABC'.join(['My', 'name', 'is', 'Simon'])
'MyABCnameABCisABCSimon'
```

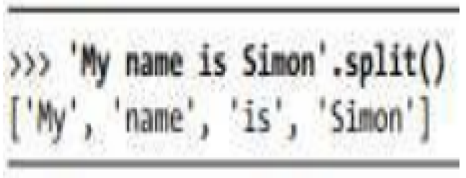
➢ string join() calls on is inserted between each string of the list argument.

o Ex: when join(['cats', 'rats', 'bats']) is called on the ', ' string, the returned string is 'cats, rats, bats'.

o join() is called on a string value and is passed a list value.

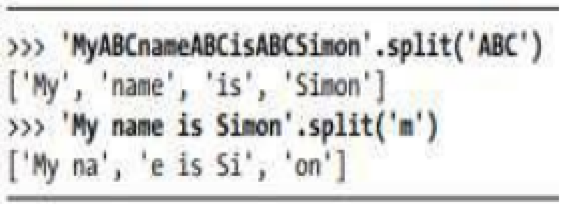
Split()

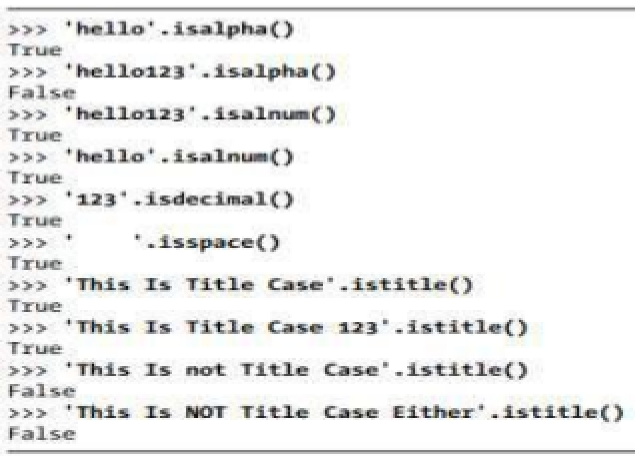
➢ The split() method is called on a string value and returns a list of strings.



```
>>> 'My name is Simon'.split()
['My', 'name', 'is', 'Simon']
```

➢ We can pass a delimiter string to the split() method to specify a different string to split upon.

	
	`>>> 'MyABCnameABCisABCSimon'.split('ABC')`
	`['My', 'name', 'is', 'Simon']`
	`>>> 'My name is Simon'.split('m')`
	`['My na', 'e is Si', 'on']`
	➢ common use of split() is to split a multiline string along the newline characters.
	● Passing split() the argument '\n' lets us split the multiline string stored in spam along the newlines and return a list in which each item corresponds to one line of the string.			

(b)
ans:
	Explain the isX String methods with Example code snippet
	
	1. **isalpha()** returns True if the string consists only of letters and is not blank.
	2. **isalnum()** returns True if the string consists only of letters and numbers and is not blank.
	3. **isdecimal()** returns True if the string consists only of numeric characters and is not blank.
	4. **isspace()** returns True if the string consists only of spaces, tabs, and newlines and is not blank.
	5. **istitle()** returns True if the string consists only of words that begin with an uppercase letter followed by only lowercase letters.
	
	`>>> 'hello'.isalpha()`
	`True`
	`>>> 'hello123'.isalpha()`
	`False`
	`>>> 'hello123'.isalnum()`
	`True`
	`>>> 'hello'.isalnum()`
	`True`
	`>>> '123'.isdecimal()`
	`True`
	`>>> '     '.isspace()`
	`True`
	`>>> 'This Is Title Case'.istitle()`
	`True`
	`>>> 'This Is Title Case 123'.istitle()`
	`True`
	`>>> 'This Is not Title Case'.istitle()`
	`False`
	`>>> 'This Is NOT Title Case Either'.istitle()`
	`False`
		[5]	CO3	L2

5(a) ans:	Explain the following methods with example: i) key( ) ii) values( ) iii) items( )	[6]	CO2	L2
(b) ans:	Explain the following with help of examples i) file size and folder contents ii) checking the path validity I) File size and folder contents: •Once you have ways of handling file paths, you can then start gathering information about specific files and folders. •The os.path module provides functions for finding the size of a file in bytes and the files and folders inside a given folder. • Calling os.path.getsize(*path*) will return the size in bytes of the file in the *path* argument. • Calling os.listdir(*path*) will return a list of filename strings for each file in the *path* argument. (Note that this function is in the os module, not os.path.)	[4]	CO3	L2

```
>>>import os
        os.path.getsize('C:\\Windows\\System32\\calc.exe')
>>>import os
        os.listdir('C:\\Windows\\System32')
>>>import os
        totalSize = 0
        for filename in os.listdir('C:\\Windows\\System32'):
  totalSize = totalSize + os.path.getsize(os.path.join('C:\\Windows\\System32', filename))
        print(totalSize)
```

II) Check Path Validity:

•Many Python functions will crash with an error if you supply them with a path that does not exist.

•The os.path module provides functions to check whether a given path exists and whether it is a file or folder.

• Calling os.path.exists(*path*) will return True if the file or folder referred to in the argument exists and will return False if it does not exist.

• Calling os.path.isfile(*path*) will return True if the path argument exists and is a file and will return False otherwise.

• Calling os.path.isdir(*path*) will return True if the path argument exists and is a folder and will return False otherwise.

```
import os
os.path.exists('C:\\Windows')
```
```
True
```
```
import os
os.path.exists('C:\\some_made_up_folder')
```
```
False
```
```
import os
os.path.isdir('C:\\Windows\\System32')
```
```
True
```
```
import os
os.path.isfile('C:\\Windows\\System32')
```
```
False
```
```
import os
os.path.isdir('C:\\Windows\\System32\\calc.exe')
```
```
False
```
```
import os
os.path.isfile('C:\\Windows\\System32\\calc.exe')
```
```
True
```

6 (a)	Illustrate with Example function of Opening of a file , reading the contents of files, writing to files	[6]	CO3	L2

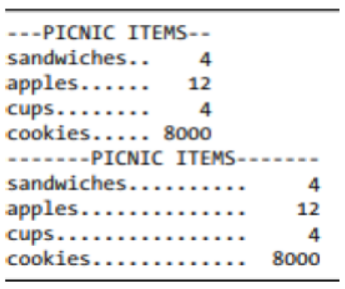
ans:

1. Opening of a file

•To open a file with the open() function, you pass it a string path indicating the file you want to open;

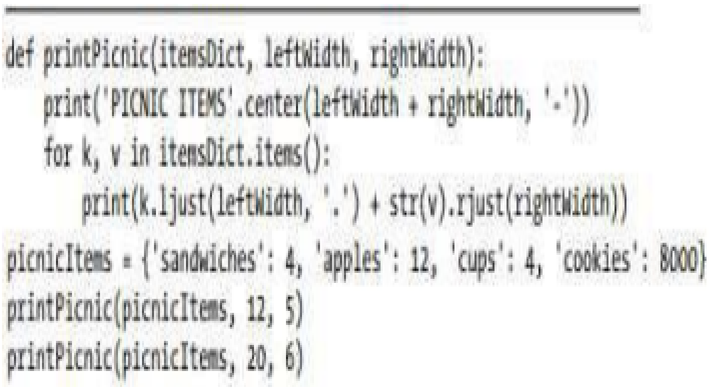
	• it can be either an absolute or relative path.			
	➢ The open() function returns a File object.			
	➢ Example: a text file named *hello.txt* using Notepad or TextEdit.			
	>>> **helloFile = open('C:\\\\Users\\*your_home_folder*\\\\hello.txt')**			
	2. Reading a file:			
	If you want to read the entire contents of a file as a string value, use the File object's read() method			
	>>> helloContent = helloFile.read()			
	>>> helloContent			
	'Hello world!'			
	➢ Alternatively, you can use the readlines() method to get a *list* of string values from the file, one string for each line of text.			
	➢ For example, create a file named *sonnet29.txt* in the same directory as *hello.txt* and write the following text in it:			
	➢			
	3. Writing a file:			
	•If the filename passed to open() does not exist, both write and append mode will create a new, blank file.			
	•➢ Example:			
	>>> baconFile = open('bacon.txt', 'w')			
	>>> baconFile.write('Hello world!\n')			
	o/p: 13			
	>>> baconFile.close()			
	>>> baconFile = open('bacon.txt', 'a')			
	>>> baconFile.write('Bacon is not a vegetable.')			
	o/p: 25			
	>>> baconFile.close()			
(b)	Develop a python program to swap cases of a given string:	[4]	CO3	L3
	Input: PyThon			
	Output: pYtHON			
ans:				
	a=input("Enter the string:")			
	for x in a:			
	if x.isupper():			
	print(x.lower(), end="")			
	elif x.islower():			
	print(x.upper(), end="")			
7 (a)	Develop a program to sort the contents of a text file and write the sorted contents in a separate text file	[6]	CO3	L3
ans:				

```
fp = open('file_name.txt', 'r+')
word_lst = []
for line in fp.readlines():
  word_lst += line.split()
word_dst = {}
for wd in set(word_lst):
  word_dst[wd] = word_lst.count(wd)
# Creating dictionary with word and count.
# sortedkey = sorted(dst,key=dst.get, reverse=True)
sortedkey = sorted(word_dst,key=word_dst.get, reverse=True)
# sortedkey is a list of keys in decreasing order.
count = 0
for y in sortedkey:
  print(count+1," Frequency of word ", y,'=', word_dst[y])
  count+=1
  if(count == 10):
    break
```

(b)	Develop a python program to get the following output using strings methods: 	[4]	CO3	L3

```
---PICNIC ITEMS--
sandwiches..    4
apples......   12
cups........    4
cookies..... 8000
-------PICNIC ITEMS-------
sandwiches..........     4
apples..............    12
cups................     4
cookies.............  8000
```

Program



```
def printPicnic(itemsDict, leftWidth, rightWidth):
    print('PICNIC ITEMS'.center(leftWidth + rightWidth, '-'))
    for k, v in itemsDict.items():
        print(k.ljust(leftWidth, '.') + str(v).rjust(rightWidth))
picnicItems = {'sandwiches': 4, 'apples': 12, 'cups': 4, 'cookies': 8000}
printPicnic(picnicItems, 12, 5)
printPicnic(picnicItems, 20, 6)
```

CI **CCI** **HOD**