USN										

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

Internal Assessment Test II – Dec 2021

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

Answer any FIVE FULL Questions		MARKS	CO	RBT
1a)	List any six methods associated with string and explain each of them with example. (each 1M) i) upper( ): This method is used to convert lower case characters into upper case characters. Ex: x = 'Python' x = x.upper( ) PYTHON ii) lower( ): This method is used to convert upper case characters into lower case characters. Ex: x = 'Python' x = x.lower( ) python iii) isupper( ): This method is used to check whether a string has at least one letter or complete string is upper or not. It returns Boolean value. Ex: x = 'Python' x = x.isupper( ) TRUE Ex: y = 'python' y = y.isupper( ) FALSE iv) islower( ): This method is used to check whether a string has at least one letter or complete string is lower or not. It returns Boolean value. Ex: x = 'Python' x = x.islower( ) TRUE Ex: y = 'PYTHON' y = y.isupper( ) FALSE	6*1=6	CO2	L2

	v) isspace( ): Returns True if the string consists only of spaces, tabs, and newlines and is not blank. Ex: ' '.isspace( ) TRUE vi) isalnum( ): Returns True if the string consists only of letters and numbers and is not blank. Ex: 'hello123'.isalnum( ) TRUE Ex: ' '.isalnum( ) FALSE			

b)	Write a Python program to swap cases of a given string. Program(4M)
	Input: python
	Output: PYTHON
	Solution 1: Using inbuilt function
	print("Enter a String")
	string = input()
	print(string.swapcase())
	Solution 2: Without using inbuilt function
	def swapcase(string):
	result_str = ""
	for item in string:
	if item.isupper():
	result_str += item.lower()
	else:
	result_str += item.upper()
	return result_str
	string = input("Enter a String")
	print(swapcase(string))
		4	CO2	L2

2a)	Describe the following with suitable Python code snippet. (i) Greedy and Non Greedy Pattern Matching(4M) (ii) findall() method of Regex object.(2M) (i) Greedy and Non Greedy Pattern Matching: Greedy pattern matching means matching with the longest possible string. By default, in python greedy matching is followed. Non-greedy pattern matching means matching with the shortest possible string. It should be represented explicitly using a question mark after the curly brackets.	4+2=6	CO3	L2

	Ex (ii) findall() method of Regex object. ⎱ search() method will return a Match object of the first matched text in the searched string. ⎱ findall() method will return the strings of every match in the searched string in the form of list of strings—as long as there are no groups in the regular expression.			

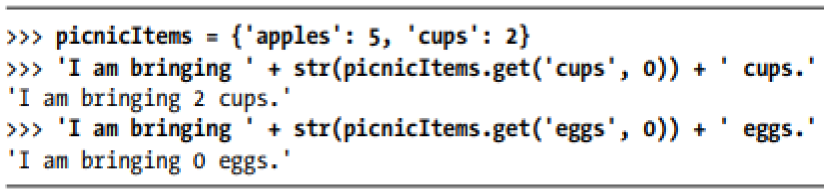
(b)	Write a python program to extract phone numbers and email addresses using regex. Use input from the clipboard and use relevant module imported. (2+2M)
	import pyperclip, re
	phoneRegex = re.compile(r'''(
	(\d{3}|\(\d{3}\))? # area code
	(\s|-|\.)? # separator
	(\d{3}) # first 3 digits
	(\s|-|\.) # separator
	(\d{4}) # last 4 digits
	(\s*(ext|x|ext.)\s*(\d{2,5}))? # extension
	)''', re.VERBOSE)
	# Create email regex.
	emailRegex = re.compile(r'''([a-zA-Z0-9._%+-]+
	[a-zA-Z0-9.-]+ # domain name
	(\.[a-zA-Z]{2,4}){1,2} # dot-something
	)''', re.VERBOSE)
		4	CO3	L2

3(a)	You are creating a fantasy video game. The data structure to model the player's inventory will be a nested dictionary. The keys are name of the players and values are items represented in dictionary again. Keys in the inner dictionary String values describing the item in the inventory and the value is an integer value detailing how many of that item the player has. For example, the dictionary value {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12} means the player has 1 rope, 6 torches, 42 gold coins, and so on. Similarly have items for different players. Write a function named displayInventory () that would take any possible "inventory" and display the total count of Inventory item given as a input. Program(5M)
	Use Nested Dict.
	stuff = {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12}
	def displayInventory(inventory):
	print("Inventory:")
	item_total = 0
	for k, v in inventory.items():
		5	CO2	L3

```
item_total = item_total + v
print(str(stuff.get(k, 0)) + ' ' + k)
print("Total number of items: " + str(item_total))
displayInventory(stuff)
```

(b) **Explain the use of get() and setdefault() methods in dictionary with suitable code snippet. The get() Method (Each 2.5M)** | 5 | CO2 | L2

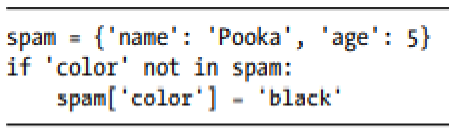
- Dictionaries have a get() method that takes two arguments:
    - o The key of the value to retrieve and
    - o A fallback value to return if that key does not exist.



```
>>> 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.'
```

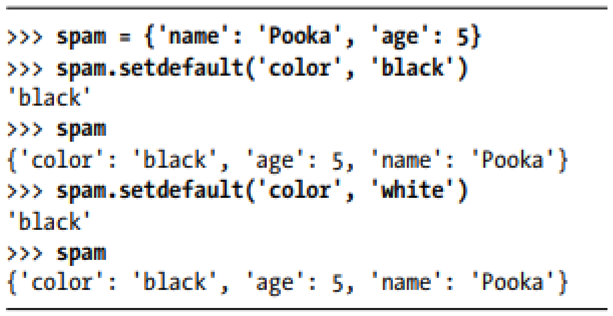
<u>The setdefault() Method</u>

- To set a value in a dictionary for a certain key only if that key does not already have a value.



```
spam = {'name': 'Pooka', 'age': 5}
if 'color' not in spam:
    spam['color'] = 'black'
```

- The setdefault() method offers a way to do this in one line of code.
- Setdeafault() takes 2 arguments:
    - ○ The first argument 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()



```
>>> spam = {'name': 'Pooka', 'age': 5}
>>> spam.setdefault('color', 'black')
'black'
>>> spam
{'color': 'black', 'age': 5, 'name': 'Pooka'}
>>> spam.setdefault('color', 'white')
'black'
>>> spam
{'color': 'black', 'age': 5, 'name': 'Pooka'}
```

method returns the key's value.

- The first time setdefault() is called, the dictionary in spam changes to {'color': 'black', 'age': 5, 'name': 'Pooka'}. 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'.			
4 (a)	Explain the various string methods for the following operations with examples. (i) Removing whitespace characters from the beginning, end or both sides of a string. (2M) (ii) To right-justify, left-justify, and center a string.(3M) • Removing whitespace characters from the beginning, end or both sides of a string. The strip() string method will return a new string without any whitespace characters at the beginning or end. The lstrip() and rstrip() methods will remove whitespace characters from the left and right ends, respectively • To right-justify, left-justify, and center a string. The rjust() and ljust() string methods return a padded version of the string they are called on, with spaces inserted to justify the text. The first argument to both methods is an integer length for the justified string.	5	CO2	L2

(b)	Write a program that reads a 10 strings as input. Display all the strings that starts with 'a' and ends with 'z'. program (5M)
	The caret symbol (^) at the start of a regex is used to indicate that a match must occur at the beginning of the searched text.
	Search.py
	import re
	regex= re.compile(r'^a... z$')
	str = input()
	if regex.search(str):
	print("Search Successful")
	else:
	print("Search unsuccessful")
	Sample output 1
	> python Search.py
	Hello
	Search unsuccessful
		5	CO3	L2

	Sample output 2 > python Search.py abyzz Search Successful Sample output 3 >python Search.py abeizz Search unsuccessful			

5 (a)	What are regular expressions? Describe question mark(?), star(*), plus(+) and dot(.) regex symbols with suitable Python code snippets.(2+3M)
	
	Regular expressions are used for pattern matching. They have special characters that are interpreted for the purpose of matching patterns in text.
	1. Import the regex module with import re.
	2. Create a Regex object with the re.compile() function.
	(Remember to use a raw string.)
	3. Pass the string you want to search into the Regex object's search() method.This returns a Match object.
	4. Call the Match object's group() method to return a string of the actual matched text.
	Optional matching with ?
	• a pattern to match only optionally
	? character flags the group that precedes it as an optional part
	(wo)? : pattern wo is an optional group
	Match has zero instances or one instance text that of wo in it
	>>> spider_re = re.compile(r'Spider(wo)?man')
	>>> mo1 = spider_re.search('The Amazing Spiderman')
	>>> mo1.group()
	Spiderman
	>>> mo2 = spider_re.search('The all new Spiderwoman')
	>>> mo2.group()
	Spiderwoman
	Here the 'wo' is matched optionally.
	Matching 0 or more with the star
	* (called the star or asterisk) means "match zero or more".
	Group that precedes the star can occur any number of times in the text can be completely absent
	Or repeated over and over again
	'Spiderman' (wo)* part of the regex matches zero instances
	'Spiderwoman' , the (wo)* matches one instance of wo
		5	CO3	L2

(b)	Write a Python program to reverse words in a given String in Python. [Input : str = "python quiz practice code." Output : str = "code. practice	5	CO2	L3

	quiz python"] Program 5M
	s = input()
	words = s.split(' ')
	string =[]
	for word in words:
	    string.insert(0, word)
	print("Reversed String:")
	print(" ".join(string))			

6 a)	In the regex created from the following code,
	import re
	phoneRegex = re.compile(r'(\d*)?-(\d{3})-(\d{3,5})')
	mo=phoneRegex.search('333-444-55555') Each 1M
	mo.group(0)
	mo.group(1)
	mo.group(2)
	mo.gorup(3)
	What is the output of this program?
	333-444-55555
	333
	444
	55555
		4	CO3	L2

(b)	How would you write a regex that matches a number with commas for every three digits from right to left? It must match the following: '42' '1,234' '6,368,745' but not the following: '12,34,567' (which has only two digits between the commas) '1234' (which lacks commas) Program: 6M Program: import re pattern =r"(?<!\d,)(?<!\d)[1-9][0-9]{0,2}(?:,\d{3})*(?!,?\d)" string = '42 1,234 6,368,745 12,34,567 1234' a = re.findall(pattern,string) print(a) # => ['42', '1,234', '6,368,745'] or `r'(?<![,\d])[1-9]\d{,2}(?:,\d{3})*(?![,\d])'` **Regex details**	6	CO3	L3

	• (?- no digit or digit +,` allowed immediately to the left of the current location • [1-9][0-9]{0,2} - a non-zero digit followed with any zero, one or two digits • (?:,\d{3})* - 0 or more occurrences of a comma and then any three digits • (?!,?\d) - no , or , + digit allowed immediately to the right of the current location.			