|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **ERODE SENGUNTHAR ENGINEERING COLLEGE**  **(An Autonomous Institution , Affiliated to Anna University)**  **PERUNDURAI, ERODE -638 057** | | | | | | | | | | |  | | |
| **I Year - B.E. Degree (Common to CSE & ECE)** | | | | | | | | | | |
| **MODEL EXAMINATION** | | | | | | | | | | |
| **Credit: 3** | | **19ES101 - PYTHON PROGRAMMING** | | | | | | | | | | | **R-2019** | | |
| **Date : 18.03.2021** | | | | **SET – II**  **Answer Key** | | | **Maximum marks: 100** | | | | | | | | |
| **Time: 01.20 A.M.– 04.20 P.M.** | | | | **ANSWER ALL THE QUESTIONS** | | | **Duration: 3Hr** | | | | | | | | |
| **Q.No.** | **Part- A** **( 10 x 1 = 10 Marks**) | | | | | | | | | **Bloom’s Level** | | | | | **COs** |
| 1. | When an algorithm is written in the form of a programming language, it becomes a \_\_\_\_\_\_\_\_\_ **c) Pseudo code** | | | | | | | | | [RE] | | | | | CO1 |
| 2. | The operation represented by parallelograms.  **a) Input** | | | | | | | | | [RE] | | | | | CO1 |
| 3. | What is the output of this expression, 3\*1\*\*3?  **b) 3** | | | | | | | | | [AP] | | | | | CO2 |
| 4. | What is the output of the following code print(bool(0),bool(-3))  **b) False True** | | | | | | | | | [AP] | | | | | CO2 |
| 5. | What will be the output of below Python code? str1="Information" print(str1[2:8])  **a) format** | | | | | | | | | [AP] | | | | | CO3 |
| 6. | Which of the following will give output as [23,2,9,75]? If list1=[6,23,3,2,0,9,8,75]  **c) list1[1:8:2]** | | | | | | | | | [AP] | | | | | CO3 |
| 7. | Predict output d = {"john":40, "peter":45}, "john" in d  **a) True** | | | | | | | | | [AP] | | | | | CO4 |
| 8. | In which part of memory does the system stores the parameter and local variables of funtion call?  **b) stack** | | | | | | | | | [RE] | | | | | CO4 |
| 9. | A package is a folder containing one or more Python modules. One of the modules in a package must be called \_\_\_\_\_\_\_  **b) \_\_init\_\_.py** | | | | | | | | | [RE] | | | | | CO5 |
| 10. | To read two characters from a file object ob, we use \_\_\_\_\_\_  **a) ob.read(2)** | | | | | | | | | [UN] | | | | | CO5 |
| **Q.No** | **Part- B** **( 10 x 2 = 20 Marks**) | | | | | | | | | **Bloom’s Level** | | | | | **COs** |
| 11. | **Define algorithm. Write an algorithm to add two numbers.**   * An algorithm is a step-by-step procedure to solve a given problem. * An algorithm is a sequence of instructions for completing a task. * Algorithm is a sequence of finite, well-defined, unambiguous, and ordered instruction. * Algorithm is an English-like representation of a logic that is used to solve the problem.   **Algorithm for finding add of 2 numbers** Step 1: Start Step 2: Read 2-numbers A, B. Step 3: Add 2-numbers and store result in C. Step 4: Display the result “C” Step 5: Stop. | | | | | | | | | [RE] | | | | | CO1 |
| 12. | **Draw a flowchart to find given number is odd or even.** | | | | | | | | | [AP] | | | | | CO1 |
| 13. | **Solve the following expression. a=5, b=4, c=3, d=2 e=1.**  **((a\*b/(c\*a)\*\*e)-45)+50**  **Ans:**  =((a\*b/(c\*a)\*\*e)-45)+50  =((**5\*4**/(**3\*5**)\*\*1)-45)+50  =((20/**15\*\*1**)-45)+50  =((20/15)-45)+50  =(1.333-45)+50  =-43.66+50  **=6.33** | | | | | | | | | [AP] | | | | | CO2 |
| 14. | **Mentions the rules for defining the variables in Python.**   * Identifiers must begin with an alphabet. * Identifier cannot be a keyword. * In Python Programming identifier is case-sensitive   **Example:** Name ≠ name ≠ NAME   * Both upper case and lower case letters are permitted. * Special character underscore ( **\_** ) only can be used. No other special symbols are not allowed | | | | | | | | | [RE] | | | | | CO2 |
| 15. | **What would the following code print?**  **Mali = 5**  **print("Mali" + " is " + str(Mali))**  **Ans:**  Maliis5 | | | | | | | | | [AP] | | | | | CO3 |
| 16. | **Which of the following two Python codes will give same output? If tupl=(5,3,1,9,0)**  **(i) print(tupl[:-1]) (ii) print(tupl[0:5])**  **(iii) print(tupl[0:4]) (iv) print(tupl[-4:])**  Ans: Option (i) & (iii)  >>> tupl=(5,3,1,9,0)  >>> print(tupl[:-1])  (5, 3, 1, 9)  >>> print(tupl[0:5])  (5, 3, 1, 9, 0)  >>> print(tupl[0:4])  (5, 3, 1, 9)  >>> print(tupl[-4:])  (3, 1, 9, 0) | | | | | | | | | [AP] | | | | | CO3 |
| 17. | **What is local and global variable? Find the output of the following code**  **x=10**  **def cg():**  **global x**  **x=x+1**  **cg()**  **print(int(x))**  **Ans:**  **Local Scope**   * A local variable is a variable defined within a function. * Function defined inside the function is said to have local scope * Variable is visible only inside that function. * Variable is not visible outside that function   **ii) Global Scope**   * Global variable is a variable defined in the main program i.e., outside the functions. * Such variables are visible throughout the program. * Variable inside the function can be declared global using the keyword “global”   Output:  11 | | | | | | | | | [AP] | | | | | CO4 |
| 18. | **List out the properties of dictionary.**   * **Unordered Set** * A Dictionary is an unordered set of key: value pairs. * The order of the dictionary element is not fixed, it may of any order. * **Not a Sequence** * Sequences are indexed by a range of ordinal numbers * Dictionary is not a Sequence, it is an unordered set of elements * **Indexed by Keys, Not numbers** * Dictionaries are indexed by keys. * Keys can be of any non-mutable type (string, number, tuple) * **Keys must be Unique** * Each of the keys within a dictionary must be unique. * Since keys are used to identify values in a dictionary, there cannot be duplicate keys in the dictionary. | | | | | | | | | [UN] | | | | | CO4 |
| 19. | **How to import the random and math modules in the python code? List any two functions in each module.**  To import random module:  >>>import random  To import math module:  >>>import math  Random functions: random.randrange(), randint(), random() etc.,  Math functions: math.sqrt(), ceil(), floor(), sin(), cos() etc., | | | | | | | | | [UN] | | | | | CO5 |
| 20. | **What is absolute path? Write a python statement to open the file “sample.txt” in the path “ E:\PYTHON\FILEHANDLE”**   * Absolute path **specifies complete path** the file where it resides. * An *absolute path*, which always begins with the root folder * Example of Absolute path (“input.txt” resides in directory (D:\\TAMIL\\FILE\\PROGRAMS)   **Python statement to open the file**  **fp=open("E:\\PYTHON** [**\\FILEHANDLE\\sample.txt","r**](file://\\FILEHANDLE\\sample.txt%22,%22r)**")**  a=fp.read()  print(a)  fp.close() | | | | | | | | | [AP] | | | | | CO5 |
| **Part- C (5 x 14 = 70 Marks)** | | | | | | | | | | | | | | | |
| 21. | **a) Write the algorithm, flowchart and pseudo code to find the greatest of three numbers.**  Step 1: Start Step 2: Read 3 numbers A, B and C. Step 3: Compare A and B. If A is greater perform Step 4 else perform Step 5 Step 4: Compare A and C. If A is greater, Print “A is great” else Print “C is great”. Step 5: Compare B and C, If B is greater, Print “B is great” else Print “C is great”. Step 6: Stop. **Pseudo code to find greatest of 3-number**   |  | | --- | | READ A,B,C IF (A>B):      IF (A>C):            DISPLAY "A is great" ELIF (B>C):      PRINT "B is great" ELSE:      PRINT "C is great" STOP |   **Flowchart to find greatest of 3-numbers** | | | | | | | | (14) | | | [AP] | | CO1 | |
| **OR** | | | | | | | |  | | |  | |  | |
| **b) What is flowchart? Tabulate the various symbols used to represent flow chart along with its purpose.**  **FLOWCHART**   * A flowchart is a graphical/pictorial representation of an algorithm to solve a given problem. * A flowchart is drawn using boxes of different shapes with lines connecting them to show the flow of control. * The purpose of drawing a flowchart is to make the logic of the program clearly in a visual form.   **Need for flowchart**   * Flowchart makes the logic clear. * Useful in coding * Used for effective analysis.     **Guidelines for preparing a flowchart**   * Flowchart should have start and end. * Flowchart must have direction of flow from top to bottom; left to right properly indicated using flow lines. * Flowchart may be simple or complex. * Standard symbol should be used for the respective action.   **Flowchart Symbols**   * A flowchart is drawn using different kinds of symbols. * A symbol used in a flowchart is for a specific purpose. * Their shapes and functions are predefined in nature.       **Advantage / Benefits**   * A flowchart helps to clarify how things are currently working and how they are improved. * It helps to remove repeated and misplaced steps.   **Limitations / Disadvantage**   * Complex when the program is very large * It is difficult to modify.   Costly and take more time | | | | | | | | (14) | | | [UN] | | CO1 | |
| 22. | **a) Explain in detail about python operators with suitable example. Write a python program to demonstrate various operators used in python.**  **Operators**   * The symbol that performs the operation/action on data are called **Operators** * The data on which the operation is being carried out are referred as **Operands**   ***Based on operands, operators can be divided into two types***   * **Unary operator** * The operators that act on ***one operand*** are referred to as **Unary operator.** * Example: * Unary + * +5, +7.8 which represents positive number * Unary - * -3, -6.8 which represents negative number * In the above examples, operator works only one operand * **Binary operator** * The operator that act upon ***two operands*** are referred to as **Binary Operator.** * **Example:** * **3+5** * **5\*7** * **70/4** * In the above examples, operator works on two operand, hence it is called "Bi" means 2, Binary Operator   **Based on operation or action, operators can be classified as follows:** **Types of operators:**   * There are six types of operator used in Python   Arithmetic operator, Relational operator  Logical operator, Bitwise operator  Identity operator, Membership operator  Assignment operator  **(1) Arithmetic operator**   * Python provides operators for basic calculations such as addition, subtraction, multiplication, division etc., * Arithmetic operators are,​​​​​  |  |  |  | | --- | --- | --- | | **Binary Operator** | **Meaning** | **Example** | | + | Addition | 5 + 2 = 7 | | - | Subtraction | 5 – 2 = 3 | | \* | Multiplication | 5 \* 2 = 10 | | / | Division | 5 / 2 = 2.5 | | // | Floor Division | 5 // 2 = 2 | | % | Modulo / Remainder | 5 % 2 = 1 | | \*\* | Exponentiation (Power) | 5 \*\* 2 = 25 |   **(2) Relational operators**                Relational operator is used to compare two values. Relational operators are symbols that are used to test the relationship between two values/variables.   |  |  |  |  | | --- | --- | --- | --- | | **Relational Operator** | **Meaning** | **Example** | **Result** | | < | Less Than | 5 < 6 | True | | 10 < 2 | False | | > | Greater Than | 5 > 6 | False | | 10 > 2 | True | | <= | Less Than Equal To | 5 <= 6 | True | | 6 <= 6 | True | | 10 <= 2 | False | | >= | Greater Than Equal To | 5 >= 6 | False | | 6 >= 6 | True | | 10 >= 2 | True | | = = | Equal To | 5 = = 6 | False | | 6 = = 6 | True | | != | Not Equal To | 5! = 6 | True | | 6! = 6 | False |   **(3) Logical Operator**  Operator which are used to combine two or more relational operator are called logical operator. These operators are used to test more than one condition at a time.   * Result of logical AND operation is True, only if both given expressions are True. * The result of logical OR operations will be True, if either of the expression is True or both expressions are True. * The logical NOT operator reverses the value of the expression.   Logical Operators are,   * and , or      , not    Consider the table given below to understand the logical operator results. Here 1 represents value is True and 0 represents value is False.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **a** | **b** | **a and b** | **a or b** | **not a** | | 0 | 0 | 0 | 0 | 1 | | 0 | 1 | 0 | 1 | 1 | | 1 | 0 | 0 | 1 | 0 | | 1 | 1 | 1 | 1 | 0 |   **Example 1**   |  |  |  | | --- | --- | --- | | **Problem** | **Intermediate result** | **Final Result** | | 5 > 3 and 5 < 10 | True and True | True | | 5 > 3 and 5 > 10 | True and False | False | | 10 < 5 and 6 > 10 | False and False | False |   **Example 2**   |  |  | | --- | --- | | **Operation** | **Result** | | 5 and 0 | 0 | | 5 and 6 | 6 | | 5 or 0 | 5 | | 5 or 6 | 5 |   **(5) Identity Operator**   * Identity operators are used to check if both reference the same object memory. * It is used to compare the memory location of two objects. * There are two identity operators in Python  |  |  |  | | --- | --- | --- | | **Operator** | **Example** | **Description** | | is | a is b | Returns **True**, is both operands are pointing to **same object**.  Returns **False**, is both operands are pointing to **different object.** | | is not | a is not b | Returns **True**, is both operands are pointing to **different object**.  Returns **False**, is both operands are pointing to **same object.** |  * **Example**     **(6) Membership Operator**   * Identity operators are used to check whether given element found in list/string/tuple * There are two identity operators in Python  |  |  | | --- | --- | | **Operator** | **Description** | | in | If the given object found, returns True  If the given object is not found, returns False | | not in | If the given object is not found, returns True  If the given object is found, returns False |  * **Example**     **(7) Augmented Assignment Operator**   * Python has an Assignment operator (=) which assigns the value specified on RHS (Right Hand Side) to the variable/object on the LHS (Left Hand Side). * Python also offers augmented assignment operator which combines the impact of an arithmetic operator  |  |  |  | | --- | --- | --- | | **Operator** | **Operation** | **Description** | | += | a+=b | a=a+b | | -= | a-=b | a=a-b | | \*= | a\*=b | a=a\*b | | /= | a/=b | a=a/b | | //= | a//=b | a=a//b | | \*\*= | a\*\*=b | a=a\*\*b | | %= | a%=b | a=a%b | | | | | | | | | (14) | | | [AP] | | CO2 | |
| **OR** | | | | | | | |  | | |  | |  | |
| **b) i) Write Python program to find a given number is a prime number or not.**  **Program**  limit=int(input("Enter the limit:"))  print("Prime numbers upto limit:",limit)  for i in range(1,limit+1):  count=0  for j in range(1,i+1):  if i%j == 0:  count+=1  if count==2:  print(i, end='\t') | | | | | | | | (7) | | | [AP] | | CO2 | |
| **ii) Write a python code to find the sum of the given digit. Example: If input is 153. Answer will be 1+5+3 = 9**  **Program**  num=int(input("Enter the number:"))  sum=0  while num>0:  digit=num%10  sum=sum+digit  num=num//10  print("Sum of digit=",sum) | | | | | | | | (7) | | | [AP] | | CO2 | |
| 23. | **a) Compare and contrast string, list and tuples. Tabulate minimum seven points of the each data types with clear examples.** | | | | | | | | (14) | | | [UN] | | CO3 | |
| **OR** | | | | | | | |  | | |  | |  | |
| **b) i) Write Python program to search the given elements in the list and display its index position. Display the message “Element not found” if the searching element does not occurs in the input list.**  **Program**  a=[]  n=int(input("Enter number of elements:"))  for i in range(n):  b=int(input("Enter element:"))  a.append(b)  X=int(input("Enter element to be searched:"))  flag = 0  for i in range(0,n):  if X == a[i]:  flag = 1  break  if flag == 1:  print("Element found at",i+1)  else:  print("Element not found") | | | | | | | | (6) | | | [AP] | | CO3 | |
| **ii) Write about any four functions used in python list data type. Give suitable example for each function.**  **1. len( )**   * Used to find the number of elements in the list * **Syntax** * len(list\_name) * **Example**     **2. max( )**   * Used to find the maximum value in the list * **Syntax** * max(list\_name) * **Example**     **3. min( )**   * Used to find the minimum value in the list * **Syntax** * min(list\_name) * **Example**     **4. sum( )**   * Used to find the sum of the elements in the list * **Syntax** * sum(list\_name) * **Example** | | | | | | | | (8) | | | [AP] | | CO3 | |
| 24. | **a) i) Explain in detail about the actual and formal parameter with suitable example.**   * **Arguments** * Python refers to the value being **passed** as Arguments * Arguments used during **function call / Main program** are called Actual Arguments. * It is also called as **Actual Argument / Actual Parameter** * **Parameters** * Python refers to the value being **received** as Parameters * Parameters used during the function definition /in **function header** are called Formal Arguments. * It is also called as **Formal Argument / Formal Parameter**       **Identify the actual and formal argument of the given code**  **Example 1**    **Actual Arguments – x**  **Formal Arguments – a & b** | | | | | | | | (6) | | | [UN] | | CO4 | |
| **ii) Elucidate in detail positional and keyword argument. Mention the rules to be followed while using both positional and keyword argument in the same function.**  **Positional arguments**   * When the function call statement must match the number and order of arguments as defined in the function definition is called Positional arguments. * Cannot omit the number of parameters * Cannot the change the order of passing * Must pass the argument exactly as defined during function definition. * Also called as Required arguments or Mandatory Arguments * **Example 1**       **Keyword arguments**   * Keyword arguments are the **named arguments** with assigned values being passed in the **function call statement**. * During the function call, name of the argument will be passed * **Order of passing the value can be changed** * It can be used with default argument. * Positional argument must not follow after keyword argument * **Example 1**      * **Example 2**     Positional argument must not follow after keyword argument)    ***Note: Order of parameter***  ***Position arguments, non-default arguments, default arguments*** | | | | | | | | (8) | | | [AP] | | CO4 | |
| **OR** | | | | | | | |  | | |  | |  | |
| **b) i) Explain about any four functions of the dictionary with syntax and example.**  **items( ) method**   * Display all the items of the Dictionary. * Returns the Dictionary element in any order. No particular sequential order maintained in the dictionary. * Dictionary is an unordered sequence. * **Syntax:** * dict\_name.items() * **Example:**     **keys( ) method**   * Display all the keys in the Dictionary. * Returns the Dictionary keys in any order. No particular sequential order maintained in the dictionary. * **Syntax:** * dict\_name.keys() * **Example:**     **values( ) method**   * Display all the values in the Dictionary. * **Syntax:** * dict\_name.values() * **Example:**     **pop( ) method**   * Removes and returns the dictionary element associated with the passed key * **Syntax:** * **dict\_name.pop(key,[errorvalue/message])** * **Example:** | | | | | | | | (8) | | | [UN] | | CO4 | |
| **ii) Write a python program to display the dictionary using pretty printing.**  **Pretty printing a Dictionary**   * Dictionary elements can be printed in a certain way. * Pretty printing requires "json" module. * Steps for pretty-printing * import json module * use function dumps( ) for prettyprinting * **Syntax** * **print(json.dumps(dict\_name,indent=)**   **Example** | | | | | | | | (6) | | | [AP] | | CO4 | |
| 25. | **a. i) Write a program to get roll number, names and marks of the students of a class (get from user), write these details in a file called “mark.txt” and read file content.**  n=int(input("No.of students:"))  fp=open("mark.txt","w")  fp.write("RollNo\tName\tMark1\tMark2\tMark3\tMark4\tMark5\tTotal\tAverage  \tRemark\n")  for i in range(n):  print("Enter Student ",i+1,"details")  rollno=int(input("Roll Number:"))  name=input("Name:")  m1=float(input("Mark1:"))  m2=float(input("Mark2:"))  m3=float(input("Mark3:"))  m4=float(input("Mark4:"))  m5=float(input("Mark5:"))  total=m1+m2+m3+m4+m5  average=total/5  if average>=90:  remark="Very Good"  elif average>=70:  remark="Good"  elif average>=50:  remark="Average"  elif average<50:  remark="Improve"  record=str(rollno)+"\t"+name+"\t"+str(m1)+"\t"+str(m2)+"\t"+str(m3)+"\t"+str(m4)+"\t"+str(m5)+"\t"+str(total)+"\t"+str(average)+"\t"+remark+"\n"  fp.write(record)  fp.close()  fp1=open("mark.txt","r")  line=" "  while line:  line=fp1.readline()  print(line)  fp1.close() | | | | | | | | (10) | | | [AP] | | CO5 | |
| **ii) Explain various forms of import statement in python with example.**  **i) import Statement**  User can use any Python source file as a module by executing an import statement in some other Python source file.  **Syntax**   |  | | --- | | import module1[, module2[,... moduleN] |   **Example**   |  | | --- | | # Import module math  **import math** |   A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.  **ii) from import Statement**  Python's ***from statement*** lets the user to import **specific attributes/functions** from a module into the current namespace.  **Syntax**   |  | | --- | | from modname import name1[, name2[, ... nameN]] |   **Example**   |  | | --- | | from math import sqrt |   This statement **does not import the entire module** math into the current namespace; it just introduces the function **sqrt** from the module math into the current namespace.  **iii) from import \* Statement:**  It is also possible to import all names from a module into the current namespace by using the following import statement  **Syntax**   |  | | --- | | from modname import \* |   **Example**   |  | | --- | | from math import \* |   This provides an easy way to import all the items from a module into the current namespace | | | | | | | | (4) | | | [UN] | | CO5 | |
| **OR** | | | | | | | |  | | |  | |  | |
| **b) i) Explain various modes used in python to open a text file.**  **open operation**   * Python has a built-in function *open()* to open a file. * This function returns a ***file object,*** also called a handle, as it is **used to read or modify the file accordingly**. * This function creates a **file** object, which would be utilized to call other support methods associated with it. * Syntax  |  | | --- | | file\_object = open(file\_name [, access\_mode]) |  * file\_object object which control further actions in the file   (like read, write, close etc.,)   * file\_name Name of the file. * access\_mode It is optional. The access\_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc.   **Modes**   |  |  | | --- | --- | | **Mode** | **Description** | | **'r'** | Open a file to perform **read only** operation. (*default*) | | **‘r+’** | Open a file to perform **read and write** operations.  File pointer will be at beginning of the file.  Will **overwrite** the content of existing file. | | **'w'** | Open a file to perform **write only** operation.  Creates a new file if it does not exist or truncates the file if it exists. | | **‘w+’** | Open a file to perform **write and read** operations.  Will **truncated** the content of existing file.  File pointer will be at beginning/end of the file (since no content). | | **'a'** | Open a file to perform **append(write) only** operation at the end of the file **without truncating** it.  Creates a new file if it does not exist. | | **‘a+’** | Open a file to perform **append(write) and read** operations.  File pointer will be at **end** of the file.  Do not truncated the content of existing file, it will **append new content** to the existing content. | | **'x'** | Open a new(exclusive) file to perform **write only** operation.  If the file already exists, the operation fails/will raise error. | | **‘x+’** | Open a new(exclusive) file to perform **write and read** operation.  If the file already exists, the operation fails/will raise error. | | **'b'** | Open in binary mode.  br, bw, ba, rb+/r+b, wb+/w+b, ab+/w+b | | | | | | | | | (8) | | | [UN] | | CO5 | |
| **ii) Develop a python program to count number of words ending with vowels. Assume the file name is “sample.txt”.**  **Example: Assume that the input file contains the string as given,**  **“Do you know file handling in python language” Output: 4**  **Output:**  fp=open("sample.txt","r")  a=fp.read()  b=a.split()  count=0  for i in b:  if i[-1] in ‘AEIOUaeiou’:  count+=1  print("count=",count) | | | | | | | | (6) | | | [AP] | | CO5 | |
| The following course outcome are assessed:- | | | | | | | | | | | | | | | |
| **COURSE OUTCOME** | | | | | | | | | | | **MARKS** | | | | |
| **CO1** | Apply problem solving techniques to real world problems. | | | | | | | | | | 34 | | | | |
| **CO2** | To recognize and construct common programming idioms: variables, loop, branch and input/output | | | | | | | | | | 34 | | | | |
| **CO3** | To design, code, and test python programs using List,Tuples and Strings | | | | | | | | | | 34 | | | | |
| **CO4** | To write code using dictionaries and functions | | | | | | | | | | 34 | | | | |
| **CO5** | To read and write data from/to files in Python Programs. | | | | | | | | | | 34 | | | | |
| **Prepared by**  S.Tamil Selvan  AP / CSE | | | **Bloom’s level** | | **Keyword** | **Marks** | | **Approved by**  **Dr.G.Sivakumar**  **HoD/CSE** | | | | | | | |
| Remembering | | RE | 08 | |
| Understanding | | UN | 59 | |
| Applying | | AP | 103 | |
| **Scrutinized by**  Dr.M.P.Thiruvenkatasuresh, Prof./CSE | | | Analyzing | | AN | - | |
| Evaluating | | EV | - | |
| Creating | | CR | - | |
| **TOTAL** | | | **170**  (Including Choices) | |