**METHODIST COLLEGE OF ENGINEERING & TECHNOLOGY (An Autonomous Institution)**

**B.E. (ECE) III-Semester (AICTE) (Regular) Examination, Feb/March -2023**

**Subject: PYTHON PROGRAMMING**

**Time: 3 hours Max.Marks:60**

**Note: Missing data, if any, maybe suitably assumed.**

**PART-A**

**Answer All the questions.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q.No.** | **Questions** | **Marks** | **CO** | **BTL** |
| **1. a** | a=31  b=12  c=2022  print(a,b,c,sep="-") | **2** |  |  |
| **b** |  | **2** |  |  |
| **c** | Iteration 1  Iteration 2  Iteration 3  While loop terminated | **2** |  |  |
| **d** |  | **2** |  |  |
| **e** | **Numbers**– They include integers, floating-point numbers, and complex numbers. eg. 1, 7.9,3+4i  **List**– An ordered sequence of items is called a list. The elements of a list may belong to different data types. Eg. [5,’market’,2.4]  **Tuple**– It is also an ordered sequence of elements. Unlike lists , tuples are immutable, which means they can’t be changed. Eg. (3,’tool’,1)  **String**– A sequence of characters is called a string. They are declared within single or double-quotes. Eg. “Sana”, ‘She is going to the market’, etc.  **Set**– Sets are a collection of unique items that are not in order. Eg. {7,6,8}  **Dictionary**– A dictionary stores values in key and value pairs where each value can be accessed through its key. The order of items is not important. Eg. {1:’apple’,2:’mango}  **Boolean**– There are 2 boolean values- **True** and **False**. | **2** |  |  |
| **f** | {'brand': 'Ford', 'model': 'Mustang', 'year': 2020} | **2** |  |  |
| **g** | A module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.  There are many built in modules in Python. Some of them are as follows:math, random , threading , collections , os , mailbox , string , time , tkinter etc | **2** |  |  |
| **h** | def f():      print("Inside Function", s)    # Global scope  s = "Hello"  f()  print("Outside Function", s)  Output:  Inside Function Hello  Outside Function Hello | **2** |  |  |
| **i** | The rename() Method  The rename() method takes two arguments, the current filename and the new filename.  Syntax  os.rename(current\_file\_name, new\_file\_name)  The remove() Method  You can use the remove() method to delete files by supplying the name of the file to be deleted as the argument.  Syntax  os.remove(file\_name) | **2** |  |  |
| **j** | f =open("ex88.txt",'w') f.write("hello world") f.close() | **2** |  |  |

**PTO**

**PART-B**

**Answer Any Five questions**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Q.No.** |  | **Questions** | **Marks** | **CO** | **BTL** |
| **2.** | **a** | Memory Management in python  Memory in Python is managed by Python private heap space. All Python objects and data structures are located in a private heap. This private heap is taken care of by Python Interpreter itself, and a programmer doesn’t have access to this private heap.  Python memory manager takes care of the allocation of Python private heap space.  Memory for Python private heap space is made available by Python’s in-built garbage collector, which recycles and frees up all the unused memory.  A comment in a computer program is text that is intended only for the human reader — it is completely ignored by  the interpreter.  **Comments starts with a #, and Python will ignore them**  #This is a comment  print("Hello, World!")  **To add a multiline comment you could insert a # for each line:**  **Example**  #This is a comment #written in #more than just one line print("Hello, World!")  **Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:**  """  This is a comment  written in  more than just one line  """  print("Hello, World!") | **4** |  |  |
| **b** | Notations: | **4** |  |  |
| **3.** | **a** | Arithmetic Operators  The operators +, -, and \* perform addition, subtraction, and multiplication, as in the  following examples:  >>> 40 + 2  42  >>> 43 - 1  42  >>> 6 \* 7  42  The operator / performs division:  >>> 84 / 2  42.0  You might wonder why the result is 42.0 instead of 42. I’ll explain in the next section.  Finally, the operator \*\* performs exponentiation; that is, it raises a number to a  power:  >>> 6\*\*2 + 6  42  **Logical Operators**  There are three logical operators, and, or, and not, that allow us to build more complex Boolean expressions from  simpler Boolean expressions. The semantics (meaning) of these operators is similar to their meaning in English. For  example, x > 0 and x < 10 produces True only if x is greater than 0 and at the same time, x is less than 10.  n % 2 == 0 or n % 3 == 0 is True if either of the conditions is True, that is, if the number n is divisible  by 2 or it is divisible by 3.  Finally, the not operator negates a Boolean value, so not (x > y) is True if (x > y) is False, that is, if x is less than or equal to y. In other words: not True is False, and not False is True.  The expression on the left of the or operator is evaluated first: if the result is True, Python does not (and need not) evaluate the expression on the right — this is called short-circuit evaluation. Similarly, for the and operator, if the  expression on the left yields False, Python does not evaluate the expression on the right. | **4** |  |  |
| **b** | # Python program to find the largest number among the three input numbers  num1 = float(input("Enter first number: "))  num2 = float(input("Enter second number: "))  num3 = float(input("Enter third number: "))  if (num1 >= num2) and (num1 >= num3):  largest = num1  elif (num2 >= num1) and (num2 >= num3):  largest = num2  else:  largest = num3  print("The largest number is", largest) | **4** |  |  |
| **4.** | **a** | A tuple is a sequence of values. The values can be any type, and they are indexed byintegers, so in that respect tuples are a lot like lists. The important difference is that tuples are immutable.  # Creating a Tuple  # with the use of string  Tuple1 = ('Geeks', 'For')  print("\nTuple with the use of String: ")  print(Tuple1)  # Creating a Tuple with  # the use of list  list1 = [1, 2, 4, 5, 6]  print("\nTuple using List: ")  print(tuple(list1))  #output  Tuple with the use of String: ('Geeks', 'For')  Tuple using List: (1, 2, 4, 5, 6) | **4** |  |  |
| **b** | you can create a new dictionary with three items:  >>> eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}  The order of the key-value pairs might not be the same.  The len function works on dictionaries; it returns the number of key-value pairs:  >>> len(eng2sp)  3  The in operator works on dictionaries, too; it tells you whether something appears as  a *key* in the dictionary (appearing as a value is not good enough).  >>> 'one' in eng2sp  True  >>> 'uno' in eng2sp  False  To see whether something appears as a value in a dictionary, you can use the method  values, which returns a collection of values, and then use the in operator:  >>> vals = eng2sp.values()  >>> 'uno' in vals  True | **4** |  |  |
| **5.** | **a** | A module is a file containing Python definitions and statements intended for use in other Python programs. There  are many Python modules that come with Python as part of the standard library.  import random  def make\_random\_ints(num, lower\_bound, upper\_bound):  4"""  Generate a list containing num random ints between lower\_bound  and upper\_bound. upper\_bound is an open bound.  """  rng = random.Random() # Create a random number generator  result = []  for i in range(num):  result.append(rng.randrange(lower\_bound, upper\_bound))  return result  output:  >>> make\_random\_ints(5, 1, 13) # Pick 5 random month numbers  [8, 1, 8, 5, 6] | **4** |  |  |
| **b** | n=int(input('Enter value of n : ')) ''' Accept values from user '''  r=int(input('Enter value of r: '))  ''' calculate nPr '''  c=n  r1=n-r  for i in range(1,n,1):  n=n\*i  for i in range(1,r1,1):  r1=r1\*i  print('nPr = ',n/r1)  ''' calculate nCr '''  for i in range(1,r,1):  r=r\*i  print('nCr = ',n/r1\*r) |  |  |  |
| **6.** | **a** | A text file is a file that contains printable characters and whitespace, organized into lines separated by newline characters. To demonstrate, we‘ll create a text file with three lines of text separated by newlines: >>> f = open("test.dat","w") >>> f.write("line one\nline two\nline three\n") >>> f.close()  The read method reads data from the file. With no arguments, it reads the entir>>> f.write("Now is the time") >>> f.write("to close the file") Closing the file tells the system that we are done writing and makes the file available for reading: >>> f.close()e contents of the file: >>> text = f.read() >>> print text | **4** |  |  |
| **b** | An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.  dividing by zero creates an exception:  >>> print 55/0  ZeroDivisionError: integer division or modulo So does accessing a nonexistent  list item: >>> a = []  >>> print a[5] |  |  |  |
| **7.** | **a** | Pseudocode is a technique used to describe the distinct steps of an algorithm in a manner that’s easy to understand for anyone with basic programming knowledge.  At its core pseudocode is the ability to represent six programming constructs (always written in uppercase): SEQUENCE, CASE, WHILE, REPEAT-UNTIL, FOR, and IF-THEN-ELSE.  SEQUENCE represents linear tasks sequentially performed one after the other.  WHILE a loop with a condition at its beginning.  REPEAT-UNTIL a loop with a condition at the bottom.  FOR another way of looping.  IF-THEN-ELSE a conditional statement changing the flow of the algorithm.  CASE the generalization form of IF-THEN-ELSE. | **4** |  |  |
| **b** | num = int(input("Enter a number: "))  # If number is greater than 1  if num > 1:  # Check if factor exist  for i in range(2,num):  if (num % i) == 0:  print(num,"is not a prime number")  break  else:  print(num,"is a prime number")    # Else if the input number is less than or equal to 1  else:  print(num,"is not a prime number") |  |  |  |
| **8.** | **a** | str\_1 = input (“Enter the string to check if it is a palindrome: “)  str\_1 = str\_1.casefold ()  rev\_str = reversed (str\_1)  if (list str\_1) == list (rev\_str):  print (“The string is a palindrome.”)  else:  print (“The string is not a palindrome.”) | **4** |  |  |
| **b** | # Python program to find the  # area and perimeter of circle in python  # Initialising the value of PI  PI = 3.14  # Getting input from user  R = float(input("Enter radius of the circle: "))  # Finding the area and perimeter of the circle  area = (PI\*R\*R)  perimeter = (2\*PI\*R)  # Printing the area and perimeter of the circle  print("The area of circle is", area)  print("The perimeter of circle is", perimeter) |  |  |  |
| **9.** | **a** | Read Only (‘r’) : Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exists, raises the I/O error. This is also the default mode in which a file is opened.  Read and Write (‘r+’): Open the file for reading and writing. The handle is positioned at the beginning of the file. Raises I/O error if the file does not exist.  Write Only (‘w’) : Open the file for writing. For the existing files, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist.  Write and Read (‘w+’) : Open the file for reading and writing. For an existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.  Append Only (‘a’): Open the file for writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.  Append and Read (‘a+’) : Open the file for reading and writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data. | **4** |  |  |
| **b** | A substring of a string is obtained by taking a slice. The operator [n:m] returns the part of the string from the nth character to the mth character, including the first but excluding the last. >>>book=‘Problem Solving and Python Programming‘ >>>print(book[0:7]) Problem >>>print(book[21:27]) python  The comparison operator works on string to check if two strings are equal. >>>word=‘VRB Publishers‘ >>>if word==‘VRB Publishers‘ Print(‗Both are Equal‘) Both are Equal |  |  |  |

**\*\*\*\*\*\***