

www.Learnpython4cbse.com
Class-XII
Subject: Computer Science (083)
Answer Key

SECTION-A								
QN.	Answer of Question							
1.	Ans. False	1						
2.	Ans. (c) alter	1						
3.	Ans: (d) dict_student.update(dict_marks)	1						
4.	Ans. (b) True	1						
5.	Ans. (b) DELETE Command	1						
6.	Ans: (c) HomePage	1						
7.	Ans. (c) None	1						
8.	Ans. (a) Year . 0. at All the best	1						
9.	Ans. (b) Statement 4	1						
10.	Ans. (c) 512	1						
11.	Ans: (d) PAN	1						
12.	Ans. (b) W* B*	1						
13.	Ans. (a) Pickling	1						
14.	Ans. (b) DISTNICT	1						
15.	Ans. Topology	1						
16.	Ans. (a) Mycur.fetch()	1						
17.	Ans. (c) A is True but R is False	1						
18.	Ans. (c) A is True but R is False	1						
SECTION-B								
19.	(i) (a) IP-Internet Protocol (b) URL- Uniform Resource Locator (1/2 mark for each) (ii) VoIP is used to transfer audio (voice) and video over internet(1 mark) OR (i) Advantage: The network remains operational even if one of the nodes stopsworking. (1 mark for any ONE advantage) (ii) <table border="1" style="width: 100%; margin-top: 10px;"> <thead> <tr> <th style="width: 50%;">Hub</th> <th style="width: 50%;">Switch</th> </tr> </thead> <tbody> <tr> <td>Hub is a passive Device</td> <td>Switch is an active device</td> </tr> <tr> <td>Hub broadcasts messages to all nodes</td> <td>Switch sends the messages to intended node.</td> </tr> </tbody> </table> Or any other valid difference between the two. (1 mark for ANY ONE difference)	Hub	Switch	Hub is a passive Device	Switch is an active device	Hub broadcasts messages to all nodes	Switch sends the messages to intended node.	2
Hub	Switch							
Hub is a passive Device	Switch is an active device							
Hub broadcasts messages to all nodes	Switch sends the messages to intended node.							
20.	def reverse(num): rev = 0 while num > 0: rem == num %10 rev = rev*10 + rem num = num//10 return rev print(reverse(1234)) (½ Mark for each correction up to any 4 corrections)	1+1 =2						

21.	<pre>def INDEX_LIST(L): indexList=[] for i in range(len(L)): if L[i]!=0: indexList.append(i) return indexList</pre> <p><i>(½ mark for correct function header 1 mark for correct loop 1 mark for correct if statement ½ mark for return statement)</i></p> <p>Note: Any other relevant and correct code may be marked</p> <p style="text-align: center;">OR</p> <pre>def Count_How_Many(Data, item): count=0 for n in Data: if(n==item): count+=1 print(item, " found ", count, "times")</pre> <p>d=[101,102,107,105,102,103,104,102] i=102 Count_How_Many(d,i)</p> <p>or any other correct logic</p>	1+1= 2
22.	['H', 'A', 'P', 'P', 'Y'] ['B', 'I', 'R', 'T', 'H', 'D', 'A', 'Y']	2
23.	<p>(i) str="PYTHON@LANGUAGE" print(str[2: :]) (ii) d=dict()</p> <p style="text-align: center;">OR</p> <p>(i) s="LANGUAGE" l=list(s) (ii) t=tuple()</p>	2
24.	<p>COUNT(*) returns the count of all rows in the table, whereas COUNT (COLUMN_NAME) is used with Column_Name passed as argument and counts the number of non-NULL values in a column that is given as argument. Here discount column is having 4 rows with NULL values.</p> <p style="text-align: center;">OR</p> <p>Use KVS; (1/2 mark) Show Tables; (1/2 mark) Desc EMPLOYEE; (1/2 MARK) Select * from EMPLOYEE; (1/2 MARK)</p>	2
25.	{20: 3, 19: 3, 17: 2}	2
SECTION-C		
26.	<p>Vande O Bharat 9 Train 1 vANDE00bHARAT99tRAIN11 (3 marks for correct answer. Partial marks may be given for partially correct answer.)</p>	3
27.	(1 mark for each correct output)	1*3 =3

	<p>(i)</p> <table border="1" data-bbox="321 92 586 317"> <tr><td>sports</td></tr> <tr><td>SOCCER</td></tr> <tr><td>TENNIS</td></tr> <tr><td>CRICKET</td></tr> <tr><td>ATHLETICS</td></tr> <tr><td>SNOOKER</td></tr> </table> <p>ii)</p> <table border="1" data-bbox="321 359 743 548"> <thead> <tr><th>Sports</th><th>MAX(salary)</th></tr> </thead> <tbody> <tr><td>SOCCER</td><td>50000</td></tr> <tr><td>TENNIS</td><td>20000</td></tr> <tr><td>CRICKET</td><td>15000</td></tr> <tr><td>ATHLETICS</td><td>12000</td></tr> </tbody> </table> <p>iii)</p> <table border="1" data-bbox="321 590 954 737"> <thead> <tr><th>pname</th><th>sports</th><th>salary</th></tr> </thead> <tbody> <tr><td>VIRAT</td><td>CRICKET</td><td>15000</td></tr> <tr><td>NEERAJ</td><td>ATHLETICS</td><td>12000</td></tr> <tr><td>SANIA</td><td>TENNIS</td><td>5000</td></tr> </tbody> </table>	sports	SOCCER	TENNIS	CRICKET	ATHLETICS	SNOOKER	Sports	MAX(salary)	SOCCER	50000	TENNIS	20000	CRICKET	15000	ATHLETICS	12000	pname	sports	salary	VIRAT	CRICKET	15000	NEERAJ	ATHLETICS	12000	SANIA	TENNIS	5000	
sports																														
SOCCER																														
TENNIS																														
CRICKET																														
ATHLETICS																														
SNOOKER																														
Sports	MAX(salary)																													
SOCCER	50000																													
TENNIS	20000																													
CRICKET	15000																													
ATHLETICS	12000																													
pname	sports	salary																												
VIRAT	CRICKET	15000																												
NEERAJ	ATHLETICS	12000																												
SANIA	TENNIS	5000																												
28.	<pre>def displaywords (): file = open('data.txt','r') st=file.read() lst=st.split() for k in lst: if len[k] >3: print(k, end=" ") file.close() displaywords ()# Call the displaywords</pre> <p>(½ mark for function header, 1 mark for opening file, 1 mark for correct for loop and condition,½ mark for closing file)</p> <p style="text-align: center;">OR</p> <pre>def count_lines(): f=open("student.txt",'r') rows=f.readlines() end_y=not_y=0 for rec in rows: if(rec[-1]=='\n'): end_y+=1 else: not_y+=1 print("The number of lines in file are", len(rows)) print("The number of lines ending with alphabet 'y' are:",end_y) print("The number of lines not ending with alphabet 'y' are:",not_y) count_lines() #call the function</pre> <p>(½ mark for function header, 1 mark for opening file, 1 mark for correct for loop and condition,½ mark for closing file)</p>	3																												
29.	<p>(i) SELECT EMP_NAME, BASIC+DA+HRA+NPS AS "GROSS SALARY" FROM SALARY;</p> <p>(ii) UPDATE SALARY SET DA=DA+0.03*BASIC;</p> <p>(iii) ALTER TABLE SALARY DROP COLUMN EMP_DESIG;</p>	1*3 =3																												

30.	<pre> data = [1,2,3,4,5,6,7,8] stack = [] def push(stack, data): for x in data: if x % 2 == 0: stack.append(x) def pop(stack): if len(stack)==0: return "stack empty" else: return stack.pop() push(stack, data) print(pop(stack)) </pre> <p>(½ mark should be deducted for all incorrect syntax. Full marks to be awarded for any other logic that produces the correct result.)</p>	3
SECTION-D		
31.	<p>i)SELECT SUM (PERIODS), SUBJECT FROM SCHOOL GROUP BY SUBJECT ; ii) SELECT MIN(EXPERIENCE), MAX(CODE) FROM SCHOOL; iii)SELECT TEACHERNAME, GENDER FROM SCHOOL, ADMIN WHERE DESIGNATION = 'COORDINATOR' AND SCHOOL.CODE=ADMIN.CODE; iv)SELECT COUNT(DISTINCT SUBJECT) FROM SCHOOL;</p> <p>(1 mark for each correct query)</p>	1*4 =4
32.	<pre> import csv def Add_New(): fout=open("playerdata.csv ","a",newline='\n') wr=csv.writer(fout) P_id=int(input("Enter Player Id :: ")) P_name=input("Enter Player name :: ") P_runs=int(input("Enter price :: ")) playerlist=[P-id,P_name,P_runs] wr.writerow(playerlist) fout.close() def Display_Record(): fin=open("playerdata.csv ","r") data=csv.reader(fin) found=False print("The Player Records are: ") for Rec in data: if int(rec[2])>5000: found=True print(rec[0],rec[1],rec[2]) if found==False: print("Such Record not found") Add_New(): Display_Record(): </pre> <p>(½ mark for importing csv module) (1 ½marks each for correct definition of Add_New() and Display_Record ()) (½ mark for function call statements)</p>	2+2= 4

SECTION-E

33.	<p>i) ADM Block Justification- It has maximum number of computers. Reduce traffic.</p> <p>ii) wired medium is ethernet cables. Following bus (cable cost efficient) or star with ADM as centre (network traffic efficient)</p> <div data-bbox="321 241 1177 451" style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <pre> graph TD DEV[DEVELOPMENT] --- HR[HUMANRESOURCE] HR --- ADM[ADM] LOG[LOGISTICS] --- ADM </pre> </div> <p>iii) (a) Switches in all the blocks since the computers need to be connected to the network. (b) Repeaters between ADM and HUMANRESOURCE block & ADM and Logistics block. The reason being the distance is more than 100m. iv) Modem should be placed in the Server building v) Optical Fiber cable connection</p>	1*5=5
34.	<p>(i) Full form of CSV is Coma Separated Value. pickle module is used for Binary files and csv module is used for importing csv files. (1 + ½ + ½)</p> <p>ii) import pickle</p> <pre> def Trace_Book(): fopen=open("library.dat","r") data=pickle.load(fopen) found=False print("The Book Records are: ") for Rec in data: if (rec[2])<1000: found=True print(rec[0],rec[1],rec[2]) if found==False: print("Such Record not found") Trace_Book(): </pre> <p style="text-align: center;">OR</p> <p>(i) (1 mark for each difference between text file and binary file)</p> <p>(ii) import pickle</p> <pre> def Get_Stud(): Total = 0 Count_rec = 0 Count_age = 0 with open("STUDENT.DAT", "rb") as F: while True: try: R=pickle.load(f) Count_rec = Count_rec+1 Total = Total+R[2] if R[2] > 18: print (R[1],"is of Age :",R[2]) Count_age += 1 except: break if Count_age == 0 : print("There is no student who is greater than 18 year") </pre>	2+3=5

Get_Stud()								
35.	(i)Any one difference:	1+4=5						
	<table border="1"> <thead> <tr> <th>CANDIDATE KEY</th> <th>ALTERNATE KEY</th> </tr> </thead> <tbody> <tr> <td>All attributes in a relation thathave potential to become a Primary key</td> <td>All the leftover candidate keys after selecting the primary key</td> </tr> </tbody> </table>		CANDIDATE KEY	ALTERNATE KEY	All attributes in a relation thathave potential to become a Primary key	All the leftover candidate keys after selecting the primary key		
CANDIDATE KEY	ALTERNATE KEY							
All attributes in a relation thathave potential to become a Primary key	All the leftover candidate keys after selecting the primary key							
<p>(ii)</p> <pre>import mysql.connector as BD def Emp_Database(): con=BD.connect(host="localhost", user="root", password="bharat", database="TOUR") BDcursor=con.cursor() print("Travels at Hilly Area and the distance more than 1000 KM.:") BDcursor.execute("select * from TRAVELS WHERE Geo_Cond ='hilly area" AND Distance<1000) TravelRec= BDcursor.fetchall() for rec in TravelRec: print(rec)</pre> <p style="text-align: center;">OR</p> <p>(i)Any one difference:</p> <table border="1"> <thead> <tr> <th>PRIMARY KEY</th> <th>UNIQUE KEY</th> </tr> </thead> <tbody> <tr> <td>There can be only one primary key in a table</td> <td>There can be more than one unique keys in a table</td> </tr> <tr> <td>The primary key cannot have null values</td> <td>Unique can have null values</td> </tr> </tbody> </table> <p>(ii)</p> <pre>import mysql.connector as cnt def Emp_Database(): con=cnt.connect(host="localhost", user="root", password="tiger", database="company") mycursor= con.cursor() print("Display Employee whose age is more than 55 years:") mycursor.execute("select * from Emp where age>55") EmpRec= mycursor.fetchall() for rec in EmpRec: print(rec)</pre>			PRIMARY KEY	UNIQUE KEY	There can be only one primary key in a table	There can be more than one unique keys in a table	The primary key cannot have null values	Unique can have null values
PRIMARY KEY	UNIQUE KEY							
There can be only one primary key in a table	There can be more than one unique keys in a table							
The primary key cannot have null values	Unique can have null values							