SAMPLE PAPER – 3 MARKING SCHEME
SUBJECT: COMPUTER SCIENCE

SuperNova-LearnPython, a YouTube channel dedicated to helping students to learn Python and computer science concepts.

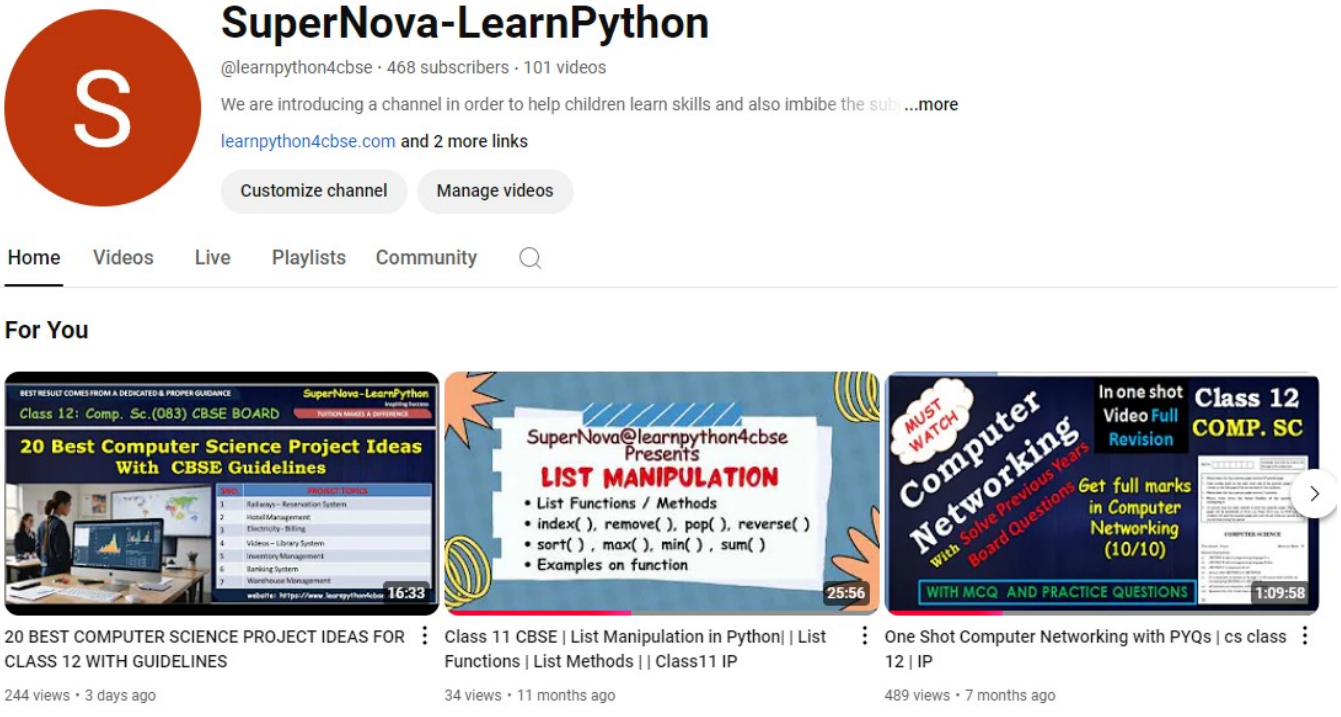
The channel covers various topics related to computer science, including ***Python programming, data file handling, computer networking***, ***SQL*** and many more.

If you're looking for video descriptions, notes, assignments, and previous years' question papers related to Python and computer science for class 11 and 12, I recommend checking out the **SuperNova-LearnPython** channel on YouTube.

You can find the channel here[1].

Happy learning! .

Please like, Subscribe and share the Channel



SAMPLE PAPER – 3 – MARKING SCHEME

Code: Learnpython4cbse-3/5(25CBSE01)

COMPUTER SCIENCE 12TH (CODE 083)

General Instructions:

Time: 3 Hrs. **Max. Marks: 70**

1. This question paper contains 37 questions.
2. All questions are compulsory. However, internal choices have been provided in some questions. Attempt only one of the choices in such questions
3. The paper is divided into 5 Sections- A, B, C, D and E.
4. Section A consists of 21 questions (1 to 21). Each question carries 1 Mark.
5. Section B consists of 7 questions (22 to 28). Each question carries 2 Marks.
6. Section C consists of 3 questions (29 to 31). Each question carries 3 Marks.
7. Section D consists of 4 questions (32 to 35). Each question carries 4 Marks.
8. Section E consists of 2 questions (36 to 37). Each question carries 5 Marks.
9. All programming questions are to be answered using Python Language only.
10. In case of MCQ, text of the correct answer should also be written.

	SECTION A [21x1= 21]	
1	TRUE	**1**
2	(a) ['Delhi', 'Mumbai', 'Chennai', 'Kolkata']#	**1**
3	(b) print (not False and True or False and True)	**1**
4	(d) ['Co', '', 'it', 'ent']	**1**
5	puC dlroW	**1**
6	(d) S1[4]="Y"	**1**
7	(d) Dict_student.update(Dict_marks)	**1**
8	(b) Replaces placeholders in the string with specified values	**1**
9	(c) DISTINCT	**1**
10	F=open("example.dat","rb+")	**1**
11	False	**1**

12	(b) 4	**1**
13	Alter Table TableName Modify ColName (NewDataType);	**1**
14	a) Only those columns whose emp_id is NULL	**1**
15	(b) BOOLEAN	**1**
16	(c) INSERT	**1**
17	(b) IP	**1**
18	(d) Both (a) and (b)	**1**
19	WiFi (IEEE 802.11 b/g/n)	**1**
20	(a) Both A and R are true and R is a correct explanation of A.	**1**
21	(d) A is false and R is true.	**1**

SECTION B **[7x2= 14 Marks]**

22

```
def fun(mylist):
    for value in mylist:
        if value.isdigit():
            print(value*3)
        else:
            print(value +'#')
mylist = ['41', 'DROND','GIRIRAJ','13','ZARA']
fun(mylist)
```

2

23	(i) Myfriends	(ii) MYFRIENDS	**2**
24	(I) A) L2.remove(20) OR B) L1.sort(reverse=True) (II) A) L2.index(50) OR L2.find(50) OR B) sum(L1)		**2**

25	Possible Outputs: (A) P* (B) P*g* (D) P*g*m* Minimum Possible Value of b: 3 Maximum Possible Value of b: 8	**2**

26	def change():
	String=**"HELLO"**
	for I in range(0,len(String)−1)**:**
	if String[I]>**="**M":
	print (String[I],end="*")
	else:
	print **(String[I-1])**
		2

27	(I) (A) PRIMARY KEY constraint should be applied to ensure that every value in the column is unique and not NULL. OR (B) UNIQUE constraint can be applied to ensure that every value in the column is unique, but NULL values are allowed. In SQL, NULL values are treated as distinct, so the UNIQUE constraint will allow multiple NULLs in the column. (II) (A) ALTER TABLE ORDERS DROP CONSTRAINT FK_CUSTOMER_ID; OR (B) ALTER TABLE ORDERS ADD CONSTRAINT FK_CUSTOMER_ID FOREIGN KEY (CUSTOMERSD) REFERENCES CUSTOMERS(ID);	**2**
28	**4 pairs – 8 wires** **The port name is Ethernet Port**	**2**
	SECTION C **[3x3= 9 Marks]**	

29 **3**

```
def CountLine():
        f=open("Help.txt",'r')
        c=0
        while True:
                line=f.readline()
                if len(line)==0:
                        break
                if line[0].isupper():
                        c=c+1

        return c
print("The No. of Lines:",CountLine())
```

30 **3**

```
R={"OM":76, "JAT":45, "BOB":89, "ALI":65, "ANU" :90, "TOM" :82}
def PUSH(S, N):
     S.append (N)
def POP(S):
     if S!=[]:
          return S.pop()
     else:
          return None

ST=[]
for k in R:
        if R[k]>=75:
                PUSH(ST, k)

while True:
     if ST!=[]:
          print( POP(ST), end=" ")
     else:
          break
```

OR

```
N=[12,13,34,56,21,79,98,22,35,38]
def PUSH(S,N):
        S.append (N)
def POP(S):
        if S!=[]:
                return S.pop()

        else:
                return None
ST=[ ]
for k in N:
        if k%2==0:
                PUSH(ST,k)
while True:
        if ST!=[]:
                print(POP(ST),end=" ")

        else:
                break
```

31 5#8#5#4# **3**

OR

300 @ 200

300 @ 100

120 @ 100

300 @ 120

SECTION D **[4x4= 16 Marks]**

32 (A) **4**

(I) Select sum(DA) from Employee group by Deptid
having sum(DA)>1000;

(II) Select * from Employee order by Qualification;

(III) Select distinct C Qualification from Employee;

(IV) Select Name, Basic+DA+HRA+Bonus as 'Gross Amount' from Employee;

OR

(i) ERROR (No DOJ Column found in Employee Table)

(ii)

Eid	Name	Deptid	Qualification	Sex	Points	Basic	DA	HRA	Bonus
1	Deepali	101	MCA	F	20	6000	2000	2300	200
2	Rajat	101	BCA	M	15	2000	300	300	30
3	Hari	102	B.A	M	15	1000	300	300	40
5	Sumit	103	B.Tech	M	32	8000	900	900	80

(iii)

Eid	MIN(HRA)
101	300
102	300
103	900

(iv)

Name	Qualification
Sumit	B.Tech
Jyoti	M.Tech

33 **4**

```
# Function to create and append data to BOOKS.CSV
def CSVOpen():
    with open("BOOKS.CSV", mode="a", newline="") as file:
        writer = csv.writer(file)
        # Adding sample records
        writer.writerow(['Python Programming', 'Deitel', 500, 20])
        writer.writerow(['Java Programming', 'John', 550, 10])
        writer.writerow(['React Guide', 'Radha', 600, 15])
```

```
        writer.writerow(['Ruby on Rails', 'Thomas', 450, 25])
        print("Records added to BOOKS.CSV.")

    # Function to read and display records where the title starts with 'R'
    def CSVRead():
        try:
            with open("BOOKS.CSV", mode="r") as file:
                reader = csv.reader(file)
                print("Books with titles starting with 'R':")
                for row in reader:
                    if row[0].startswith('R'):
                        print(row)
        except FileNotFoundError:
            print("The file BOOKS.CSV does not exist. Please create it first.")

    # Main program
        CSVOpen()
        # Display records where title starts with 'R'
        CSVRead()
```

34 (4)

(i) SELECT NAME FROM COMPANY
WHERE COMPANY.CID=CUSTOMER. CID AND PRICE < 30000;

(ii) DELETE FROM CUSTOMER WHERE NAME = 'DEEPAK KUMAR';

(iii) UPDATE CUSTOMER
SET PRICE = PRICE + 1000
WHERE NAME LIKE 'S%' ;

(iv) (A) ALTER TABLE CUSTOMER ADD TOTALPRICE DECIMALC10,2);
OR
(B) SELECT MIN(PRICE), MAX(PRICE) FROM CUSTOMER WHERE QTY>10;

35 (4)

```
import mysql.connector
cnx = mysql.connector.connect(host="localhost", user="root", password="Arm@2021",
database="DefDB")
cursor = cnx.cursor()
query = "SELECT Wing, Grade FROM Defence WHERE Rank = 'Major' OR Rank = 'Brigadier'"
cursor.execute(query)
```

```
        results = cursor.fetchall()
        for row in results:
                wing, grade = row
                print(f"Wing: {wing}\tGrade: {grade}")
        cursor. closeO
        cnx.close()
```

SECTION E **[2x5= 10 Marks]**

36 (I) **5**

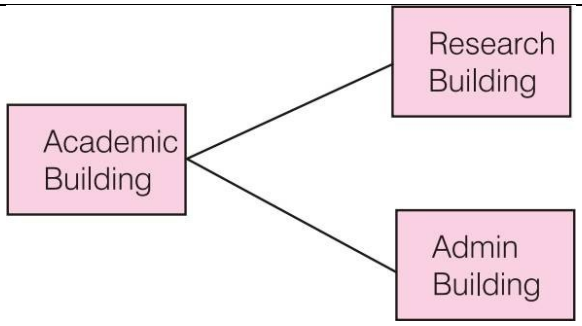
```
import pickle
def input_books():
        books = []
        n = int(input("Enter the number of books you want to add: "))
        for i in range(n):
                book_id = int(input("Enter BookID: "))
                book_title = input("Enter Book Title: ")
                author_name = input("Enter Author Name: ")
                price = float(input("Enter Price (in rupees): "))
                books.append([book_id, book_title, author_name, price])
        return books
books_list = input_books()
def append_book_data(books):
with open('BOOK.DAT', 'ab') as file:
        for book in books:

        pickle.dump(book, file)
        print("Book data appended successfully.")
append_book_data(books_list)
```

II)

```
import pickle
def update_title_ai():
        updated_books = []
        try:
                with open('BOOK.DAT', 'rb') as file:
                while True:
```

```
                        try:
                                book = pickle.load(file)
                                if book[3] > 550: # If price > 550 rupees
                                        book[1] = ' Python programming for AI '
                                updated_books.append(book)
                        except EOFError:
                                break # End of file reached
        except FileNotFoundError:
                print("No book data found. Please add books first.")
        return
        with open('BOOK.DAT', 'wb') as file:
                for book in updated_books:
                pickle.dump(book, file)
        print("Books Title updated to Python programming for AI.")
update_title_ai()
III)
import pickle
def display_book():
        try:
                with open(‘BOOK.DAT’, 'rb') as file:
                while True:
                try:
                        book = pickle.load(file)
                        if book[3] >550:
                                print(f"Book ID: {book[0]}")
                                print(f"Book Title: {book[1]}")
                                print(f"Author Name: {book[2]}")
                                print(f"Price: {book[3]}")
                                print("--------------------")
                except EOFError:
                        break # End of file reached
        except FileNotFoundError:
                print("No book data found. Please add books first.")
display_book():
```

37	 (i) (ii) The most suitable place to house the server is Academic Building as it has maximum number of computers. Thus, it decreases the cabling cost and increase efficiency of network. (iii) (c) Switch is to be installed in each of building to connect all the computers. (iv) (f) Satellite connection (v) (A) MAN (Metropolitan Area Network) Or (B) Local Area Network	**5**