CBSE 12 Computer Science Question Paper with Solutions

Time Allowed :3 hours	**Maximum Marks :**100	**Total questions :**65

General Instructions

Read the following instructions very carefully and strictly follow them:

1. Please check this question paper contains 35 questions.
2. The paper is divided into 5 Sections - A, B, C, D and E.
3. Section A, consists of 18 questions (1 to 18). Each question carries 1 mark.
4. Section B, consists of 7 questions (19 to 25). Each question carries 2 marks.
5. Section C, consists of 5 questions (26 to 30). Each question carries 3 marks.
6. Section D, consists of 2 questions (31 to 32). Each question carries 4 marks.
7. Section E, consists of 3 questions (33 to 35). Each question carries 5 marks.
8. All programming questions are to be answered using Python Language only.

1. State True or False:

While defining a function in Python, the positional parameters in the function header must always be written after the default parameters.

Correct Answer: False.

Solution: In Python, the positional parameters must always appear before the default parameters in the function definition. This is because Python relies on the position of arguments for mapping to function parameters. If default parameters were placed before positional ones, it would create ambiguity in argument mapping.

Quick Tip

Always remember that positional arguments must come before default arguments in a function definition in Python.

---

2. The `SELECT` statement when combined with `_____` clause, returns records without repetition.

(a) `DISTINCT`

(b) `DESCRIBE`

(c) `UNIQUE`

(d) `NULL`

Correct Answer: (a) `DISTINCT`.

Solution: The `DISTINCT` clause in SQL is used to ensure that the results of a query do not contain duplicate rows. It filters out duplicate records from the output and ensures that each row in the result set is unique.

Example:

```
SELECT DISTINCT column_name FROM table_name;
```

This will return only unique values from the specified column. None of the other options (`DESCRIBE`, `UNIQUE`, or `NULL`) perform this functionality.

Quick Tip

Use `DISTINCT` in SQL queries to remove duplicate rows from the result set.

---

3. What will be the output of the following statement: `print(16*5/4*2/5-8)`

(a) -3.33

(b) 6.0

(c) 0.0

(d) -13.33

Correct Answer: (c) 0.0

Solution: The given expression evaluates step by step according to the precedence of arithmetic operators (division and multiplication from left to right, followed by subtraction):

$$16 \times 5 = 80, \quad 80 \div 4 = 20, \quad 20 \times 2 = 40, \quad 40 \div 5 = 8, \quad 8 - 8 = 0.0.$$

Therefore, the output of the statement is 0.0.

Quick Tip

Always follow operator precedence: parentheses first, then exponentiation, followed by multiplication/division, and finally addition/subtraction (PEMDAS).

---

4. What possible output from the given options is expected to be displayed when the following Python code is executed?

```
import random
Signal = ['RED', 'YELLOW', 'GREEN']
for K in range(2, 0, -1):
    R = random.randrange(K)
    print(Signal[R], end='#')
```

(a) `YELLOW # RED #`

(b) `RED # GREEN #`

(c) `GREEN # RED #`

(d) `YELLOW # GREEN #`

Correct Answer: (a) `YELLOW # RED #`.

Solution: The code iterates over the range $(2, 0, -1)$, which means $K = 2$ and $K = 1$. For each iteration: - When $K = 2$: `random.randrange(2)` generates either 0 or 1. The output could be `Signal[0]` (`'RED'`) or `Signal[1]` (`'YELLOW'`). - When $K = 1$: `random.randrange(1)` always generates 0. The output is `Signal[0]` (`'RED'`).

One possible output is `YELLOW # RED #`, which matches option (a).

Quick Tip

The `random.randrange(K)` function generates a random integer between 0 and $K-1$. Use loops carefully to trace all potential outputs.

---

5. In SQL, the aggregate function which will display the cardinality of the table is _____.

(a) `sum()`

(b) `count(*)`

(c) `avg()`

(d) `sum(*)`

Correct Answer: (b) `count(*)`.

Solution: In SQL, the `count(*)` function is used to count the number of rows in a table, which effectively gives the cardinality of the table. Other functions like `sum()` and `avg()` are used for numerical aggregations, not for counting rows.

Quick Tip

Use `count(*)` when you need to determine the total number of rows in a table, regardless of whether they contain NULL values or not.

---

6. Which protocol out of the following is used to send and receive emails over a computer network?

(a) `PPP`

(b) `HTTP`

(c) `FTP`

(d) `SMTP`

Correct Answer: (d) `SMTP`.

Solution: The `SMTP` (Simple Mail Transfer Protocol) is used to send emails, while receiving emails typically involves protocols like `IMAP` or `POP3`. The other options, such as `FTP` (File Transfer Protocol) and `HTTP` (Hypertext Transfer Protocol), are not related to email communication.

> **Quick Tip**
>
> Remember, `SMTP` is for sending emails, and `IMAP`/`POP3` is for retrieving emails from a mail server.

---

7. Identify the invalid Python statement from the following:

(a) `d = dict()`

(b) `e = {}`

(c) `f = []`

(d) `g = dict{}`

Correct Answer: (d) `g = dict{}`.

Solution: The statement `g = dict{}` is invalid because the `dict()` function should not include curly braces {} as an argument. The correct way to initialize an empty dictionary is `g = dict()` or `g = {}`. All other options correctly initialize dictionaries or lists.

> **Quick Tip**
>
> To initialize an empty dictionary, use `dict()` or {}. Avoid mixing the two syntax forms.

---

8. Consider the statements given below and then choose the correct output from the given options:

```
myStr = "MISSISSIPPI"
print(myStr[:4] + "#" + myStr[-5:])
```

(a) `MISSI#SIPPI`

(b) `MISS#SIPPI`

(c) `MISS#IPPIS`

(d) `MISSI#IPPIS`

Correct Answer: (b) `MISS#SIPPI`.

Solution: The slicing operations in the given code work as follows: - `myStr[:4]` extracts the first 4 characters, which are `"MISS"`. - `myStr[-5:]` extracts the last 5 characters, which are `"SIPPI"`. - These two substrings are concatenated with a `"#"` in between, resulting in `"MISS#SIPPI"`.

Quick Tip

In Python, positive indices count from the start (0), while negative indices count from the end (-1). Use them effectively for slicing strings.

---

9. Identify the statement from the following which will raise an error:

(a) `print("A"*3)`

(b) `print(5*3)`

(c) `print("15" + 3)`

(d) `print("15" + "13")`

Correct Answer: (c) `print("15" + 3)`.

Solution: In Python: - Option (a): `"A"*3` repeats the string `"A"` three times, resulting in `"AAA"`. - Option (b): `5*3` performs integer multiplication, resulting in `15`. - Option (d): `"15" + "13"` concatenates two strings, resulting in `"1513"`. - Option (c): `"15" + 3` raises a `TypeError` because Python does not allow concatenation of a string (`"15"`) and an integer (`3`).

Quick Tip

Always ensure that the operands in concatenation operations are of the same data type. Use `str()` to convert integers to strings if needed.

---

10. Select the correct output of the following code:

```
event = "G20 Presidency@2023"
L = event.split(' ')
print(L[::-2])
```

(a) `'G20'`

(b) `['Presidency@2023']`

(c) `['G20']`

(d) `'Presidency@2023'`

Correct Answer: (c) `['G20']`.

Solution: The `split()` method divides the string `"G20 Presidency@2023"` into a list of substrings based on spaces, resulting in:

$$L = \texttt{["G20", "Presidency@2023"]}.$$

The slicing operation `L[::-2]` works as follows: - `[::-2]` reverses the list and selects every second element. - Starting from the end, `L[::-2]` picks `"G20"`.

Thus, the output is `['G20']`.

Quick Tip

Use `list[::-step]` to reverse or skip elements in a list. The step size determines how elements are selected.

---

11. Which of the following options is the correct unit of measurement for network bandwidth?

(a) `KB`

(b) `Bit`

(c) `Hz`

(d) `Km`

Correct Answer: (b) `Bit`.

Solution: Network bandwidth is typically measured in terms of bits per second (e.g., Mbps or Gbps), which indicate the amount of data that can be transmitted per second. The other options, such as `KB` (kilobytes), `Hz` (frequency), and `Km` (distance), are unrelated to network bandwidth measurement.

Quick Tip

Remember, bandwidth measures the data transfer rate, and its unit is typically `bps` (bits per second).

---

12. Observe the given Python code carefully:

```
a = 20
def convert(a):
    b = 20
    a = a + b

convert(10)
print(a)
```

Select the correct output from the given options:

(a) `10`

(b) `20`

(c) `30`

(d) `Error`

Correct Answer: (b) `20`.

Solution: In Python, variables defined inside a function are local to that function and do not affect variables outside the function. Here, the variable `a` inside the function `convert()` is a local variable, and its value does not modify the global variable `a`. Therefore, the value of `a` printed outside the function remains `20`.

> **Quick Tip**
>
> To modify a global variable inside a function, use the `global` keyword. Otherwise, changes to the variable inside the function are local and do not affect the global scope.

---

13. State whether the following statement is True or False:

While handling exceptions in Python, name of the exception has to be compulsorily added with `except` clause.

Correct Answer: False.

Solution: In Python, the `except` clause can be used without specifying a particular exception. For example:

```
try:
    # some code
except:
    # handle any exception
```

However, it is a good practice to specify the exception type to make error handling more explicit and readable.

> **Quick Tip**
>
> Avoid using a bare `except` clause, as it can catch unexpected exceptions and make debugging difficult.

---

14. Which of the following is not a DDL command in SQL?

(a) `DROP`

(b) `CREATE`

(c) `UPDATE`

(d) `ALTER`

Correct Answer: (c) `UPDATE`.

Solution: `UPDATE` is a Data Manipulation Language (DML) command used to modify data in existing rows of a table. On the other hand, `DROP`, `CREATE`, and `ALTER` are Data Definition Language (DDL) commands, which are used to define and modify the structure of a database.

Quick Tip

Remember, DDL commands deal with the structure of the database (e.g., `CREATE`, `DROP`), while DML commands (e.g., `UPDATE`, `INSERT`) handle the data within the tables.

---

15. Fill in the blank:

_____ is a set of rules that needs to be followed by the communicating parties in order to have a successful and reliable data communication over a network.

Correct Answer: Protocol.

Solution: A protocol is a set of rules that defines how data is transmitted and received over a network. It ensures that communication between devices is reliable and successful. Examples of protocols include `HTTP`, `FTP`, and `SMTP`.

Quick Tip

Familiarize yourself with common protocols like `HTTP` (web communication), `FTP` (file transfer), and `SMTP` (email).

16. Consider the following Python statement:

```
F = open('CONTENT.TXT')
```

Which of the following is an invalid statement in Python?

(a) `F.seek(1, 0)`

(b) `F.seek(0, 1)`

(c) `F.seek(0, -1)`

(d) `F.seek(0, 2)`

Correct Answer: (c) `F.seek(0, -1)`.

Solution: The `seek()` method in Python is used to change the file's current position. It takes two arguments: 1. `offset`: The position (in bytes) to move the file pointer to. 2. `whence`: The reference point for `offset`. Valid values for `whence` are:

- `0`: Beginning of the file.
- `1`: Current file position.
- `2`: End of the file.

A value of `-1` for `whence` is invalid, making `F.seek(0, -1)` incorrect.

Quick Tip

Always use valid `whence` values (0, 1, or 2) with `F.seek()` to avoid runtime errors.

---

17. Assertion (A): CSV file is a human-readable text file where each line has a number of fields, separated by a comma or some other delimiter.

Reason (R): `writerow()` method is used to write a single row in a CSV file.

Mark the correct choice:

(a) Both (A) and (R) are true and (R) is the correct explanation for (A).

(b) Both (A) and (R) are true and (R) is not the correct explanation for (A).

(c) (A) is true but (R) is false.

(d) (A) is false but (R) is true.

Correct Answer: (b) Both (A) and (R) are true and (R) is not the correct explanation for (A).

Solution: the writerow() method is not the reasoning behind the assertion that CSV files are human-readable text files with fields separated by commas. The assertion describes the structure and purpose of CSV files, while the reason describes a function related to working with CSV files. Thus, while both statements are true, the reason does not explain the assertion.

Quick Tip

Always analyze the relationship between the assertion and reason carefully. Even if both statements are true, the reason must directly explain the assertion to qualify as a correct explanation.

---

18. Assertion (A): The expression `"HELLO".sort()` in Python will give an error.

Reason (R): `sort()` does not exist as a method/function for strings in Python.

Mark the correct choice:

(a) Both (A) and (R) are true and (R) is the correct explanation for (A).

(b) Both (A) and (R) are true and (R) is not the correct explanation for (A).

(c) (A) is true but (R) is false.

(d) (A) is false but (R) is true.

Correct Answer: (a) Both (A) and (R) are true and (R) is the correct explanation for (A).

Solution: The `sort()` method is not defined for string objects in Python. It is used for lists, not strings. Therefore, trying to use `sort()` on a string object like `"HELLO"` results

in an `AttributeError`, making both the assertion and reasoning true, with (R) correctly explaining (A).

Quick Tip

Use `sorted()` to sort characters in a string after converting it to a list using `list()`.

19. (A)

1. Expand the following terms:
   XML, **PPP**

2. Give one difference between circuit switching and packet switching.

OR

(B)

1. Define the term web hosting.

2. Name any two web browsers.

Solution:

(A)

1. **XML:** Extensible Markup Language
   PPP: Point-to-Point Protocol

2. **Difference:**

Circuit Switching	Packet Switching
A dedicated communication path is established between sender and receiver.	Data is divided into packets and transmitted individually over the network.
Suitable for real-time communication like voice calls.	Suitable for data communication like emails and file transfers.

Quick Tip

For technical terms like XML and PPP, remember their full forms and purposes, as they are often used in networking and data transmission. Similarly, understanding the fundamental differences between circuit switching and packet switching is crucial for real-time and data communication scenarios.

(B)

1. **Web Hosting:** Web hosting is essential for deploying websites online. It provides a platform where website files, such as HTML, CSS, and JavaScript, are stored and managed, ensuring they are accessible to users 24/7.
2. Examples of web browsers: Google Chrome, Mozilla Firefox.

Quick Tip

Circuit switching is ideal for uninterrupted communication, whereas packet switching is cost-effective and efficient for non-real-time data transmission.

---

20. The code given below accepts five numbers and displays whether they are even or odd:

Observe the following code carefully and rewrite it after removing all syntax and logical errors. Underline all the corrections made.

Given Code:

```
def EvenOdd()
    for i in range(5):
        num=int(input("Enter a number")
        if num/2==0:
            print("Even")
        else:
            print("Odd")
EvenOdd()
```

Corrected Code:

```
def EvenOdd():  # Added colon at the function definition
    for i in range(5):  # No change
        num = int(input("Enter a number: "))  # Added a colon and clos
        if num % 2 == 0:  # Changed division '/' to modulus '%' for ev
            print("Even")  # No change
        else:
            print("Odd")  # No change
EvenOdd()  # No change
```

Corrections Made:

- Added a colon ':' after the function definition 'def EvenOdd()'.
- Fixed the missing colon in the input statement: 'input("Enter a number: ")'.
- Replaced the division operator '/' with the modulus operator '

Quick Tip

Always check for syntax errors (e.g., missing colons or parentheses) and ensure the correct operators are used for logical conditions (e.g., '

---

21. (A) Write a user-defined function in Python named `showGrades(S)` which takes the dictionary `S` as an argument.

The dictionary `S` contains `Name: [Eng, Math, Science]` as `key:value` pairs. The function displays the corresponding grade obtained by the students according to the following grading rules:

Average of Eng, Math, Science	Grade
≥ 90	A
< 90 but ≥ 60	B
< 60	C

Example: Consider the following dictionary: `S={"AMIT": [92, 86, 64], "NAGMA": [65, 42, 43], "DAVID": [92, 90, 88]}`

The output should be: `AMIT - B NAGMA - C DAVID - A`

Solution:

```
def showGrades(S):  # Function definition
    for name, marks in S.items():  # Loop through each student and the
        avg = sum(marks) / len(marks)  # Calculate the average of mark
        if avg >= 90:  # Check for grade A
            grade = "A"
        elif avg >= 60:  # Check for grade B
            grade = "B"
        else:  # Check for grade C
            grade = "C"
        print(f"{name} - {grade}")  # Print the name and grade

# Example dictionary
S = {"AMIT": [92, 86, 64], "NAGMA": [65, 42, 43], "DAVID": [92, 90, 88
showGrades(S)  # Call the function
```

Explanation:

- The function `showGrades(S)` iterates through the dictionary `S`, where each key is a student's name, and the value is a list of marks in English, Math, and Science.
- The average is calculated using `sum(marks) / len(marks)`.
- Grades are assigned based on the average using the conditions:
  - If the average is greater than or equal to 90, the grade is `"A"`.
  - If the average is less than 90 but greater than or equal to 60, the grade is `"B"`.
  - If the average is less than 60, the grade is `"C"`.
- The result is printed in the format `Name - Grade`.

Quick Tip

When dealing with dictionaries containing lists as values, use `.items()` to loop through the key-value pairs and apply logic to the values directly.

(B) Write a user-defined function in Python named `Puzzle(W, N)` which takes the argument `W` as an English word and `N` as an integer and returns the string where every `N`th alphabet of the word `W` is replaced with an underscore (`"_"`)

Example: If `W` contains the word `"TELEVISION"` and `N` is 3, then the function should return the string `"TE_EV_SI_N"`. Likewise, for the word `"TELEVISION"` if `N` is 4, the function should return `"TEL_VIS_ON"`.

Solution:

```
def Puzzle(W, N):  # Function definition
    result = ""  # Initialize an empty string to store the result
    for i in range(len(W)):  # Loop through each character in the word
        if (i + 1) % N == 0:  # Check if the (i+1)th character is the
            result += "_"  # Replace the Nth character with an underscore
        else:
            result += W[i]  # Otherwise, keep the character as is
    return result  # Return the resulting string

# Example usage
W = "TELEVISION"
N = 3
print(Puzzle(W, N))  # Output: "TE_EV_SI_N"

N = 4
print(Puzzle(W, N))  # Output: "TEL_VIS_ON"
```

Explanation:

- The function `Puzzle(W, N)` iterates through each character of the input string `W`.
- Using the condition `(i + 1) % N == 0`, it checks whether the position of the character (1-based index) is a multiple of `N`.

- If true, the character is replaced with an underscore (`"_"`), otherwise, the original character is retained.
- The resulting string is built incrementally and returned as output.

Quick Tip

Use `(i + 1) % N == 0` to easily identify every `N`th element in a list or string when iterating with a zero-based index.

---

22. Write the output displayed on execution of the following Python code:

```
LS = ["HIMALAYA", "NILGIRI", "ALASKA", "ALPS"]
D = {}
for S in LS:
    if len(S) % 4 == 0:
        D[S] = len(S)
for K in D:
    print(K, D[K], sep = "#")
```

Solution:

```
Output:
HIMALAYA#8
ALPS#4
```

Explanation:

- The list `LS` contains the strings: `["HIMALAYA", "NILGIRI", "ALASKA", "ALPS"]`.
- The code iterates through each string in `LS`. If the length of the string is divisible by 4, it adds the string as a key and its length as the value in dictionary `D`.
- The dictionary after execution becomes: `{"HIMALAYA": 8, "ALPS": 4}`.
- Finally, the dictionary keys and values are printed with a `#` separator.

Final Output:

```
HIMALAYA#8
ALPS#4
```

Quick Tip

Use the `len()` function to calculate the length of strings, and the `%` operator to check divisibility conditions in Python loops.

---

23. (A) Write the Python statement for each of the following tasks using built-in functions/methods only:

(i) To remove the item whose key is `"NISHA"` from a dictionary named `Students`.

Solution:

```
Students.pop("NISHA", None)
```

Explanation:

- The `pop()` method removes the item with the specified key `"NISHA"` from the dictionary.
- If the key does not exist, the second argument (`None`) ensures that no error is raised.

(ii) To display the number of occurrences of the substring `"is"` in a string named `message`. **Solution:**

```
message.count("is")
```

Explanation:

- The `count()` method counts the number of occurrences of the substring `"is"` in the string `message`.
- For example, if `message = "This is his book"`, the output will be `3`.

Quick Tip

Use `dict.pop()` to safely remove items from a dictionary and `str.count()` to count substrings in Python.

(B) A tuple named `subject` stores the names of different subjects. Write the Python commands to convert the given tuple to a list and thereafter delete the last element of the list.

Solution:

```
# Given tuple
subject = ("Mathematics", "Physics", "Chemistry", "Biology")

# Step 1: Convert the tuple to a list
subject_list = list(subject)

# Step 2: Delete the last element of the list
subject_list.pop()

# Resulting list
print(subject_list)  # Output: ['Mathematics', 'Physics', 'Chemistry']
```

Explanation:

- A tuple named `subject` is given, which stores the names of different subjects.
- To convert the tuple to a list, the `list()` function is used: `subject_list = list(subject)`
- The `pop()` method is then used to remove the last element of the list: `subject_list.pop()`.
- The resulting list contains all elements of the tuple except the last one.

Example:

```
Input tuple: ("Mathematics", "Physics", "Chemistry", "Biology")
Resulting list: ['Mathematics', 'Physics', 'Chemistry']
```

Quick Tip

Use `list()` to convert a tuple to a list, and `pop()` to remove the last element of a list.

---

24. (A) Ms. Veda created a table named `Sports` in a MySQL database, containing columns `Game_id`, `P_Age`, and `G_name`.

After creating the table, she realized that the attribute `Category` has to be added. Help her to write a command to add the `Category` column. Thereafter, write the command to insert the following record in the table:

- `Game_id`: G42
- `P_Age`: Above 18
- `G_name`: Chess
- `Category`: Senior

Solution:

```
-- Step 1: Add the Category column to the table
ALTER TABLE Sports ADD Category VARCHAR(20);

-- Step 2: Insert the given record into the table
INSERT INTO Sports (Game_id, P_Age, G_name, Category)
VALUES ("G42", "Above 18", "Chess", "Senior");
```

Explanation:

- The `ALTER TABLE` statement is used to modify the structure of an existing table. The command `ADD Category VARCHAR(20)` adds a new column named `Category` with a data type of `VARCHAR` and a maximum length of 20 characters.
- The `INSERT INTO` statement is used to add a new record to the table `Sports`. The column names and their corresponding values are specified.

- The resulting table `Sports` will now include the record:
-

Game_id	P_Age	G_name	Category
G42	Above 18	Chess	Senior

Quick Tip

Use the `ALTER TABLE` command to add new columns and the `INSERT INTO` command to add records to a table in MySQL.

(B) Write the SQL commands to perform the following tasks:

(i) View the list of tables in the database `Exam`.

(ii) View the structure of the table `Term1`.

Solution:

```
-- (i) View the list of tables in the database Exam
SHOW TABLES;

-- (ii) View the structure of the table Term1
DESCRIBE Term1;
```

Explanation:

- The `SHOW TABLES` command lists all the tables in the currently selected database. Ensure that the database `Exam` is selected before executing this command using `USE Exam`.
- The `DESCRIBE` command displays the structure of the specified table `Term1`, including column names, data types, and constraints.

Quick Tip

Use `SHOW TABLES` to list tables in a database and `DESCRIBE` to view the structure of a specific table.

---

25. Predict the output of the following code:

```
def callon(b=20, a=10):
    b = b + a
    a = b - a
    print(b, "#", a)
    return b

x = 100
y = 200
x = callon(x, y)
print(x, "@", y)
y = callon(y)
print(x, "@", y)
```

Solution:

```
# Step-by-step execution:
# 1. Initially, x = 100 and y = 200.
# 2. First function call: callon(100, 200)
#    b = 100 + 200 = 300
#    a = 300 - 200 = 100
#    Output: 300 # 100
#    Return value: 300
#    x is updated to 300.

# 3. print(x, "@", y)
#    Output: 300 @ 200

# 4. Second function call: callon(200)
#    b = 200 + 10 = 210
#    a = 210 - 10 = 200
```

```
#     Output: 210 # 200
#     Return value: 210
#     y is updated to 210.

# 5. print(x, "@", y)
#     Output: 300 @ 210

# Final Output:
300 # 100
300 @ 200
210 # 200
300 @ 210
```

Explanation:

- The function `callon` takes two arguments `b` and `a`, with default values `b=20` and `a=10`.
- Inside the function:
  - `b` is updated as `b = b + a`.
  - `a` is updated as `a = b - a`.
  - The function prints the values of b and a, separated.
  - Finally, the updated value of `b` is returned.
- The first function call updates `x`, while the second function call updates `y`.
- The final output is generated based on the updated values of `x` and `y`.

Quick Tip

Understand the sequence of operations and how variables are updated during function calls to accurately predict the output of a Python program.

---

26. Write the output on execution of the following Python code:

```
S = "Racecar Car Radar"
L = S.split()
for W in L:
    x = W.upper()
    if x == x[::-1]:
        for I in x:
            print(I, end="*")
    else:
        for I in W:
            print(I, end="#")
print()
```

Solution:

```
# Step-by-step execution:
# 1. S = "Racecar Car Radar"
# 2. L = S.split() => L = ["Racecar", "Car", "Radar"]

# Iteration 1: W = "Racecar"
#    x = W.upper() => x = "RACECAR"
#    Check if x == x[::-1]: True (Palindrome)
#    Output: R*A*C*E*C*A*R*

# Iteration 2: W = "Car"
#    x = W.upper() => x = "CAR"
#    Check if x == x[::-1]: False (Not a palindrome)
#    Output: C#a#r#

# Iteration 3: W = "Radar"
#    x = W.upper() => x = "RADAR"
#    Check if x == x[::-1]: True (Palindrome)
#    Output: R*A*D*A*R*
```

```
# Final Output:
R*A*C*E*C*A*R*
C#a#r#
R*A*D*A*R*
```

Explanation:

- The string `S` is split into words using the `split()` function, resulting in a list `L = ["Racecar", "Car", "Radar"]`.
- The program iterates over each word in `L`, converts it to uppercase (`x = W.upper()`), and checks if it is a palindrome (`x == x[::-1]`).
- If the word is a palindrome, each character of the word is printed in uppercase, separated by `"*"`.
- If the word is not a palindrome, each character of the word is printed as is, separated.
- The final output is printed as a single line with no additional spaces or newlines in between.

Quick Tip

Use `x[::-1]` to reverse a string and compare it with the original to check for palindromes. Also, use `end=""` in `print()` to control the separator between characters.

---

27. Consider the table `ORDERS` given below and write the output of the SQL queries that follow:

ORDNO	ITEM	QTY	RATE	ORDATE
1001	RICE	23	120	2023-09-10
1002	PULSES	13	120	2023-10-18
1003	RICE	25	110	2023-11-17
1004	WHEAT	28	65	2023-12-25
1005	PULSES	16	110	2024-01-15
1006	WHEAT	27	55	2024-04-15
1007	WHEAT	25	60	2024-04-30

(i) `SELECT ITEM, SUM(QTY) FROM ORDERS GROUP BY ITEM;`

(ii) `SELECT ITEM, QTY FROM ORDERS WHERE ORDATE BETWEEN '2023-11-01' AND '2023-12-31';`

(iii) `SELECT ORDNO, ORDATE FROM ORDERS WHERE ITEM = 'WHEAT' AND RATE>=60;`

Solution:

Query (i):

```
Output:
ITEM        SUM(QTY)
RICE        48
PULSES      29
WHEAT       80
```

Explanation:

- The `GROUP BY` clause groups the rows by the `ITEM` column.
- The `SUM(QTY)` function calculates the total quantity for each item.

Query (ii):

```
Output:
ITEM        QTY
RICE        25
WHEAT       28
```

Explanation:

- The `WHERE` clause filters rows with `ORDATE` between `'2023-11-01'` and `'2023-12-31'`.
- Only rows with `ORDNO = 1003 (RICE)` and `ORDNO = 1004 (WHEAT)` satisfy the condition.

Query (iii):

```
Output:
ORDNO       ORDATE
1004        2023-12-25
1007        2024-04-30
```

Explanation:

- The `WHERE` clause filters rows where `ITEM = 'WHEAT'` and `RATE >= 60`.
- Only rows with `ORDNO = 1004` and `ORDNO = 1007` meet the condition.

Quick Tip

Use the `GROUP BY` clause for aggregation and functions like `SUM()` or `COUNT()`, and the `WHERE` clause with conditions for filtering rows.

---

28. (A) Write a user-defined function in Python named `showInLines()` which reads contents of a text file named `STORY.TXT` and displays every sentence in a separate line.

Assume that a sentence ends with a full stop (`.`), a question mark (`?`), or an exclamation mark (`!`).

Example: If the content of the file `STORY.TXT` is: `Our parents told us that we must eat vegetables to be healthy. And it turns out, our parents were right! So, what else did our parents tell?`

Then the function should display: `Our parents told us that we must eat vegetables to be healthy.`

```
And it turns out, our parents were right!
So, what else did our parents tell?
```

Solution:

```
def showInLines():
    #Open the file STORY.TXT in read mode
    with open("STORY.TXT", "r") as file:
        content = file.read()  # Read the entire content of the file
        sentences = content.split('. ')  # Split by '. ' for sentences

        # Process further to handle '? ' and '! '
        final_sentences = []
        for sentence in sentences:
            sub_sentences = sentence.split('? ') if '?' in sentence el
            final_sentences.extend(sub_sentences)

        # Print each sentence in a separate line
        for sentence in final_sentences:
            print(sentence.strip())  # Strip leading/trailing whitespa
```

Explanation:

- The file `STORY.TXT` is opened in read mode using the `with open()` statement.
- The content of the file is read into a string and split into sentences using `split('. ')`.
- For sentences containing `?` or `!`, further splitting is done to ensure each sentence is correctly separated.
- Each sentence is then printed on a new line after removing extra whitespaces using `strip()`.

Quick Tip

Use `split()` to separate text into sentences based on delimiters like `.`, `?`, and `!`. For handling multiple delimiters, additional splitting or regex can be used.

(B) Write a function, `c_words()`, in Python that separately counts and displays the number of uppercase and lowercase alphabets in a text file, `Words.txt`.

Solution:

```
def c_words():
    # Open the file Words.txt in read mode
    with open("Words.txt", "r") as file:
        content = file.read()  # Read the entire content of the file
        upper_count = 0  # Initialize uppercase count
        lower_count = 0  # Initialize lowercase count

        # Iterate through each character in the content
        for char in content:
            if char.isupper():  # Check for uppercase letters
                upper_count += 1
            elif char.islower():  # Check for lowercase letters
                lower_count += 1

        # Display the counts
        print(f"Uppercase letters: {upper_count}")
        print(f"Lowercase letters: {lower_count}")
```

Explanation:

- The file `Words.txt` is opened in read mode, and its content is read into a string.
- The function iterates through each character in the file.
- The `isupper()` method is used to check for uppercase letters, and the `islower()` method is used to check for lowercase letters.
- Counts of uppercase and lowercase letters are maintained in separate variables and displayed after iteration.

Quick Tip

Use `isupper()` and `islower()` to efficiently count uppercase and lowercase letters in a string. Combine with `open()` for file handling.

---

29. Consider the table `Projects` given below:

P_id	Pname	Language	Startdate	Enddate
P001	School Management System	Python	2023-01-12	2023-04-03
P002	Hotel Management System	C++	2022-12-01	2023-02-02
P003	Blood Bank	Python	2023-02-11	2023-03-02
P004	Payroll Management System	Python	2023-03-12	2023-06-02

Based on the given table, write SQL queries for the following:

(i) Add the constraint, `primary key` to column `P_id` in the existing table `Projects`.

(ii) Change the language to `Python` of the project whose ID is `P002`.

(iii) Delete the table `Projects` from the MySQL database along with its data.

Solution:

Query (i): Add the primary key constraint to the column `P_id`.

```
ALTER TABLE Projects
ADD PRIMARY KEY (P_id);
```

Explanation:

- The `ALTER TABLE` statement modifies the structure of the table.
- The `ADD PRIMARY KEY` command assigns the primary key constraint to the `P_id` column, ensuring that it uniquely identifies each record in the table.

Query (ii): Change the language to `Python` for the project with ID `P002`.

```
UPDATE Projects
SET Language = "Python"
WHERE P_id = "P002";
```

Explanation:

- The UPDATE statement modifies existing records in the table.
- The `SET` clause updates the Language column to ”Python” for the row where the P-id is ”P002”.
- The WHERE clause ensures that only the specified row is updated.

Query (iii): Delete the table `Projects` along with its data.

```
DROP TABLE Projects;
```

Explanation:

- The `DROP TABLE` statement permanently deletes the table `Projects` from the database, including all its data and structure.
- This action cannot be undone, so use it with caution.

Quick Tip

Use `ALTER TABLE` to modify table structure, `UPDATE` to change data, and `DROP TABLE` to permanently remove a table from the database.

---

30. Consider a list named `Nums` which contains random integers.

Write the following user-defined functions in Python and perform the specified operations on a stack named `BigNums`:

(i) `PushBig()`: It checks every number from the list `Nums` and pushes all such numbers which have 5 or more digits into the stack `BigNums`.

(ii) `PopBig()`: It pops the numbers from the stack `BigNums` and displays them. The function should also display `"Stack Empty"` when there are no more numbers left in the stack.

Example: If the list `Nums` contains the following data: `Nums = [213, 10025, 167, 254923, 14, 1297653, 31498, 386, 92765]`

On execution of `PushBig()`, the stack `BigNums` should store: `[10025, 254923, 1297653, 31498, 92765]`

On execution of `PopBig()`, the following output should be displayed:

```
92765
31498
1297653
254923
10025
Stack Empty
```

Solution:

```
# Stack for BigNums
BigNums = []

def PushBig():
    global BigNums
    # List of random integers
    Nums = [213, 10025, 167, 254923, 14, 1297653, 31498, 386, 92765]

    for num in Nums:  # Iterate through each number in Nums
        if len(str(num)) >= 5:  # Check if the number has 5 or more di
            BigNums.append(num)  # Push to the stack BigNums

def PopBig():
    global BigNums
    while BigNums:  # While the stack is not empty
        print(BigNums.pop())  # Pop and display the top element
    print("Stack Empty")  # Display message when stack is empty

# Example execution
PushBig()
```

```
print("Stack after PushBig():", BigNums)  # Display the stack after pu
PopBig()  # Display the popped elements
```

Explanation:

- The `PushBig()` function iterates through the list `Nums`, checks the number of digits in each number using `len(str(num))`, and pushes numbers with 5 or more digits to the stack `BigNums`.
- The `PopBig()` function pops elements from the top of the stack `BigNums` using `pop()` and displays them. When the stack becomes empty, it displays `"Stack Empty"`.
- The example input `Nums` results in `BigNums = [10025, 254923, 1297653, 31498, 92765]` after executing `PushBig()`. The elements are popped in reverse order.

Quick Tip

Use `len(str(num))` to count the number of digits in a number and the `pop()` method to remove elements from a stack in LIFO order.

---

31. Consider the tables `Admin` and `Transport` given below:

Table: Admin

S_id	S_name	Address	S_type
S001	Sandhya	Rohini	Day Boarder
S002	Vedanshi	Rohtak	Day Scholar
S003	Vibhu	Raj Nagar	NULL
S004	Atharva	Rampur	Day Boarder

Table: Transport

S_id	Bus_no	Stop_name
S002	TSS10	Sarai Kale Khan
S004	TSS12	Sainik Vihar
S005	TSS10	Kamla Nagar

Write SQL queries for the following:

(i) Display the student name and their stop name from the tables `Admin` and `Transport`.

(ii) Display the number of students whose `S_type` is not known.

(iii) Display all details of the students whose name starts with `'V'`.

(iv) Display student ID and address in alphabetical order of student name, from the table `Admin`.

Solution:

Query (i): Display the student name and their stop name from the tables `Admin` and `Transport`.

```
Select A.S_name, T.Stop_name from Admin A ,Transport T
where A.S_id=T.S_id;
```

Explanation:

- The `INNER JOIN` combines the tables `Admin` and `Transport` based on the common column `S_id`.
- The query retrieves the `S_name` from `Admin` and `Stop_name` from `Transport`.

Quick Tip

Use table aliases to simplify SQL queries and ensure clarity when referencing multiple tables in joins or conditions.

Query (ii): Display the number of students whose `S_type` is not known.

```
SELECT COUNT(*) AS Unknown_S_type_Count
FROM Admin
WHERE S_type IS NULL;
```

Explanation:

- The `WHERE` clause filters rows where `S_type` is `NULL`.
- The `COUNT(*)` function calculates the number of such rows.

Query (iii): Display all details of the students whose name starts with `'V'`.

```
SELECT *
FROM Admin
WHERE S_name LIKE 'V%';
```

Explanation:

- The `LIKE` operator checks for names that start with the letter `'V'`.
- The `%` is a wildcard that matches any number of characters following `'V'`.

Query (iv): Display student ID and address in alphabetical order of student name, from the table `Admin`.

```
SELECT S_id, Address
FROM Admin
ORDER BY S_name ASC;
```

Explanation:

- The `ORDER BY` clause sorts the rows alphabetically based on the `S_name` column in ascending order (`ASC`).
- The query selects the `S_id` and `Address` columns for display.

Quick Tip

Use `INNER JOIN` to combine data from multiple tables, `LIKE` for pattern matching, and `ORDER BY` for sorting data in SQL queries.

---

32. Sangeeta is a Python programmer working in a computer hardware company. She has to maintain the records of the peripheral devices. She created a csv file named `Peripheral.csv` to store the details.

Structure of `Peripheral.csv`: `[P_id, P_name, Price]` where:

- `P_id` is the Peripheral device ID (integer).
- `P_name` is the Peripheral device name (string).
- `Price` is the Peripheral device price (integer).

Sangeeta wants to write the following user-defined functions:

1. `Add_Device()`: To accept a record from the user and add it to a CSV file, `Peripheral.csv`.
2. `Count_Device()`: To count and display the number of peripheral devices whose price is less than 1000.

Solution:

```
import csv

# Function to add a record to Peripheral.csv
def Add_Device():
    # Open the file in append mode
    with open("Peripheral.csv", "a", newline="") as file:
        writer = csv.writer(file)
        # Accept device details from the user
        P_id = int(input("Enter Peripheral ID: "))
        P_name = input("Enter Peripheral Name: ")
        Price = int(input("Enter Price: "))
        # Write the record to the CSV file
        writer.writerow([P_id, P_name, Price])
        print("Record added successfully.")

# Function to count devices with price < 1000
def Count_Device():
    count = 0  # Initialize count
    # Open the file in read mode
    with open("Peripheral.csv", "r") as file:
        reader = csv.reader(file)
        for row in reader:
            try:
                # Check if price is less than 1000
```

```
                if int(row[2]) < 1000:
                    count += 1
            except ValueError:
                pass  # Skip header or invalid rows
    print(f"Number of devices with price less than 1000: {count}")

# Example Usage
# Add_Device()
# Count_Device()
```

Explanation:

- The `Add_Device()` function:
  - Opens the file `Peripheral.csv` in append mode.
  - Accepts the details of the peripheral device (`P_id`, `P_name`, and `Price`) from the user.
  - Writes the new record as a row in the CSV file using the `csv.writer`.
- The `Count_Device()` function:
  - Opens the file `Peripheral.csv` in read mode.
  - Iterates through each row in the file and checks if the `Price` (third column) is less than 1000.
  - Increments the count for each such device and displays the total count at the end.

Quick Tip

Use `csv.writer` to write data to a CSV file and `csv.reader` to read data from a CSV file. Handle exceptions for invalid or header rows when processing numeric values.

---

33. Infotainment Ltd. is an event management company with its prime office located in Bengaluru. The company is planning to open its new division at three different locations in Chennai named Vajra, Trishula, and Sudershana.

Distances between various locations:

From - To	Distance
Vajra to Trishula	350 m
Trishula to Sudershana	415 m
Sudershana to Vajra	300 m
Bengaluru Office to Chennai	2000 km

Number of computers installed at various locations:

Location	Number of Computers
Vajra	120
Sudershana	75
Trishula	65
Bengaluru Office	250

Solution:

(i) **Suggest and draw the cable layout to efficiently connect various locations in Chennai division for connecting the digital devices.**

- The most efficient cable layout would form a triangular topology connecting Vajra, Trishula, and Sudershana.
- The connections will be as follows:
  - Vajra to Trishula (350 m)
  - Trishula to Sudershana (415 m)
  - Sudershana to Vajra (300 m)
- This topology minimizes the total cable length and provides redundancy.

(ii) **Which block in Chennai division should host the server? Justify your answer.**

- The server should be hosted at Vajra because it has the highest number of computers (120) among the locations, ensuring maximum direct access and reducing latency for the majority of users.

(iii) **Which fast and effective wired transmission medium should be used to connect the prime office at Bengaluru with the Chennai division?**

- Optical fiber should be used as it supports high-speed data transfer over long distances (2000 km) with minimal data loss.

(iv) **Which network device will be used to connect the digital devices within each location of Chennai division so that they may communicate with each other?**

- A switch should be used to connect the devices within each location as it allows efficient communication between multiple devices within a local network.

(v) **Suggest a networking device that should be installed to refresh the data and reduce the data loss during transmission to and from different locations of Chennai division.**

- A repeater should be installed to amplify and regenerate the signal, reducing data loss during transmission between the locations in the Chennai division.

Quick Tip

Use optical fiber for long-distance connections, switches for internal networks, and repeaters to mitigate signal degradation over large distances.

---

34. (A) (i) Differentiate between `'w'` and `'a'` file modes in Python.

Solution:

File Mode	Description
`'w'`	Opens the file for writing. If the file exists, its content is erased.
`'a'`	Opens the file for appending. Data is added to the end of the file.

(ii) Consider a binary file, `items.dat`, containing records stored in the given format:

```
{item_id:  [item_name, amount]}
```

Write a function, `Copy_new()`, that copies all records whose `amount` is greater than 1000 from `items.dat` to `new_items.dat`.

Solution:

```
import pickle

def Copy_new():
    # Open source file for reading
    with open("items.dat", "rb") as source_file:
        # Open destination file for writing
        with open("new_items.dat", "wb") as dest_file:
            try:
                while True:
                    # Load record from source file
                    record = pickle.load(source_file)
                    # Check if amount > 1000
                    if record["amount"] > 1000:
                        # Write to destination file
                        pickle.dump(record, dest_file)
            except EOFError:
                pass  # End of file reached
```

Explanation:

- The source file `items.dat` is opened in binary read mode (`'rb'`), and the destination file `new_items.dat` is opened in binary write mode (`'wb'`).
- Each record is read using `pickle.load()` and checked for the condition `amount > 1000`.
- Records satisfying the condition are written to the destination file using `pickle.dump()`.

(B) (i) What is the advantage of using `with` clause while opening a data file in Python? Also give syntax of `with` clause.

Solution:

- **Advantage:** The `with` clause automatically manages resources. It ensures that the file is properly closed after its suite finishes execution, even if an exception occurs.

Syntax:

```
with open("filename", "mode") as file:
    # Perform file operations
```

(ii) A binary file, `EMP.DAT`, has the following structure:

`[Emp_Id, Name, Salary]`

where

`Emp_Id` : Employee ID

`Name` : Employee Name

`Salary` : Employee Salary

Write a user-defined function, `disp_Detail()`, that would read the contents of the file `EMP.DAT` and display the details of those employees whose salary is below `25000`.

Solution:

```
import pickle

def disp_Detail():
    # Open the binary file for reading
    with open("EMP.DAT", "rb") as file:
        try:
            while True:
                # Load a record from the file
                record = pickle.load(file)
                # Check if salary < 25000
                if record[2] < 25000:
                    print(f"Emp_Id: {record[0]}, Name: {record[1]}, Sa
        except EOFError:
            pass  # End of file reached
```

Explanation:

- The file `EMP.DAT` is opened in binary read mode (`'rb'`).

- Each record is read using `pickle.load()` and checked for the condition `Salary < 25000`.
- Details of matching records are printed in the specified format.

Quick Tip

Use `pickle.load()` for reading records from a binary file and handle the `EOFError` exception to terminate reading at the end of the file.

---

35. (A) (i) Define cartesian product with respect to RDBMS.

Solution: The Cartesian product in RDBMS is a result of the `CROSS JOIN` operation between two tables. It combines every row of the first table with every row of the second table, resulting in all possible combinations of rows.

(ii) Write a program in Python to update the quantity to 20 of the records whose item code is 111 in the table named `shop` in the MySQL database named `Keeper`.

Solution:

```
import mysql.connector

def update_quantity():
    try:
        # Establish connection to the database
        conn = mysql.connector.connect(
            host="localhost",
            user="admin",
            password="Shopping",
            database="Keeper"
        )
        cursor = conn.cursor()
```

```
        # SQL query to update quantity
        query = "UPDATE shop SET Qty = 20 WHERE Item_code = 111"
        cursor.execute(query)
        conn.commit()  # Save changes to the database
        print("Quantity updated successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
        conn.close()

# Call the function
update_quantity()
```

Explanation:

- The connection is established using `mysql.connector.connect()` with the given credentials.
- The `UPDATE` statement modifies the `Qty` field for rows where `Item_code = 111`.
- `conn.commit()` saves the changes to the database.
- The connection is closed using `cursor.close()` and `conn.close()`.

(B) (i) Give any two features of SQL.

Solution:

1. SQL is used to perform CRUD operations: Create, Read, Update, and Delete data in a relational database.
2. It supports functions like `JOIN`, `GROUP BY`, and `ORDER BY` to efficiently retrieve and manipulate data.

(ii) Sumit wants to write a code in Python to display all the details of the passengers from the table `flight` in MySQL database, `Travel`. The table contains the following attributes:

- `F_code`: Flight code (String)
- `F_name`: Name of flight (String)
- `Source`: Departure city of flight (String)
- `Destination`: Destination city of flight (String)

Consider the following to establish connectivity between Python and MySQL:

- Username : `root`
- Password : `airplane`
- Host : `localhost`

Solution:

```
import mysql.connector

def display_passengers():
    try:
        # Establish connection to the database
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="airplane",
            database="Travel"
        )
        cursor = conn.cursor()

        # SQL query to fetch all records from flight table
        query = "SELECT * FROM flight"
        cursor.execute(query)

        # Display records
        print("F_code | F_name | Source | Destination")
```

```
        for row in cursor.fetchall():
            print(f"{row[0]} | {row[1]} | {row[2]} | {row[3]}")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
        conn.close()

# Call the function
display_passengers()
```

Explanation:

- The connection is established using `mysql.connector.connect()` with the given credentials.
- The `SELECT * FROM flight` query fetches all rows and columns from the `flight` table.
- `cursor.fetchall()` retrieves all the results and prints them in a formatted manner.
- The connection is closed using `cursor.close()` and `conn.close()`.

Quick Tip

Always close the database connection after completing your operations to free up resources and maintain security.
