

# **RAMAKRISHNA MISSION (C.B.S.E.), GWALIOR**



## **ACADEMIC SESSION :2022-23 C.S. ASSIGNMENT FILE**

**SUBMITTED TO:  
C.S. DEPARTMENT  
MR. HEMANT DIXIT SIR**

**SUBMITTED BY:  
SANSKAR SHRIVASTAVA  
CLASS : XII 'C'**

# **CERTIFICATE**

**This is to certify that, Sanskar Shrivastava, of Class 12th C has successfully completed the project file of C.S. Practicals under the Guidance of Hemant Dixit Sir during the Year 2022-23.**

Signature of Principal  
Swami Supradiptananda Maharaj

Signature of Teacher  
Hemant Dixit Sir

# ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who have provided me with the opportunity to perform this project. I take this opportunity to thank our head of the school, **Swami Supradiptananda Maharaj**, who was always supportive and helpful in fulfilling all our academic requirements. I would like to thank my **C.S. Teacher Mr. Hemant Dixit Sir**, whose valuable guidance has helped me to complete the project. His suggestions and instructions have served as a major contributor towards the successful outcome of this project. I would like to thank my parents for giving encouragement, enthusiasm and invaluable assistance to me. Last but not the least; I would like to thank all my classmates who have supported me in various aspects.

**SANSKAR SHRIVASTAVA**

**XII 'C'**

# INDEX

- 1.Extending List
- 2.Poping Item
- 3.Slicing List
- 4.Even Squarer
- 5.Remove Method
- 6.Mean Calculator of List
- 7.Sort
- 8.Duplicate Element Checker
- 9.Duplicate Remover
- 10.Tuple Creation using Existing Sequence
- 11.Traversing a Tuple
- 12.JoiningTuple
- 13.Slicing Tuple
- 14.Tuple Differentiator
- 15.Index of the minimum element
- 16.Minimum and Maximum marks Teller
- 17.Mixed Keys
- 18.Creating a Nested Dictionary
- 19.get() method
- 20.Accessing element using key
- 21.pop() method
- 22.Looping Dictionary
- 23.Copying Dictionary
- 24.Sort
- 25.CreatingTuplefrom InputSequence

# 1.EXTENDING LIST

```
In [3]: # Python program to demonstrate
# Addition of elements in a List

# Creating a List
List = [1, 2, 'three', 4]
print("Initial List: ")
print(List)
#extending List
List.extend([8, 'Python', 'Always'])
print("\nList after performing Extend Operation: ")
print(List)
```

OUTPUT: Initial List:  
[1, 2, 'three', 4]

List after performing Extend Operation:  
[1, 2, 'three', 4, 8, 'Python', 'Always']

# 2. POPING ITEM

```
List = [1,2,3,4,5]

# Removing element from the
# Set using the pop() method at specific location
List.pop(2)
print("\nList after popping an element: ")
print(List)
```

OUTPUT: List after popping an element:  
[1, 2, 4, 5]

### 3.SLICING LIST

```
# Creating a List
List = ['C', 'O', 'M', 'P', 'U', 'T',
        'E', 'R', 'W', 'O', 'R', 'K', 'S']
print("Initial List: ")
print(List)

# Print elements from beginning to a specific point using Slice
Sliced_List = List[:-5]
print("\nElements sliced till 6th element from last: ")
print(Sliced_List)
```

OUTPUT :

```
Initial List:
['C', 'O', 'M', 'P', 'U', 'T', 'E', 'R', 'W', 'O', 'R', 'K', 'S']

Elements sliced till 6th element from last:
['C', 'O', 'M', 'P', 'U', 'T', 'E', 'R']
```

### 4.EVEN SQUARER

```
#printing even square list
Even_square = []

for x in range(1, 11):
    if x % 2 == 0:
        Even_square.append(x**2)

print(Even_square)
```

OUTPUT : [4, 16, 36, 64, 100]

## 5.REMOVE METHOD

```
# Creating a List
List = [1, 2, 3, 4, 5, 6,
        7, 8, 9, 10, 11, 12]
print("Initial List: ")
print(List)

# Removing elements from List
# using Remove() method
List.remove(5)
List.remove(6)
print("\nList after Removal of two elements: ")
print(List)
```

OUTPUT : Initial List:  
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

List after Removal of two elements:  
[1, 2, 3, 4, 7, 8, 9, 10, 11, 12]

## 6.MEAN CALCULATOR OF LIST

```
# calculating mean of a list
list1 = eval(input("Enter list : "))
length = len(list1)
mean = sum = 0
for i in range(0, length):
    sum +=list1[i]
mean = sum/length
print("Given list is :", list1)
print("The mean of the given list is :", mean)
```

OUTPUT : Enter list : [1, 2, 3, 4]  
Given list is : [1, 2, 3, 4]  
The mean of the given list is : 2.5

## 7.SORT

```
#Sort
val = eval(input("Enter a list :") )
print("Original list :", val)
val = val * 2
print("Replicated list :", val)
val.sort( )
print("Sorted in ascending order :", val)
val.sort(reverse = True)
print("Sorted in descending order :", val)
```

OUTPUT : Enter a list :[1, 2, 4, 8, 16, 32]  
Original list : [1, 2, 4, 8, 16, 32]  
Replicated list : [1, 2, 4, 8, 16, 32, 1, 2, 4, 8, 16, 32]  
Sorted in ascending order : [1, 1, 2, 2, 4, 4, 8, 8, 16, 16, 32, 32]  
Sorted in descending order : [32, 32, 16, 16, 8, 8, 4, 4, 2, 2, 1, 1]

## 8. DUPLICATE ELEMENT CHECKER

```
# Duplicate Element Checker
listA = eval(input("Enter list1 :"))
listB = eval(input("Enter list2 :"))
len1 = len(listA)
len2 = len(listB)
for a in range(len1) :
    ele = listA[a]
    if ele in listB :
        print("Overlapped")
        break
else :
    print("Separated" )
```

OUTPUT: Enter list1 :[1, 3, 4, 5, 7, 9]  
Enter list2 :[2, 4, 6, 9, 11]  
Overlapped

## 9. DUPLICATE REMOVER

```
# Duplicate Remover
val = eval(input("Enter a list :"))
tmp = []
print("original list is :", val)
for a in val :
    if a not in tmp :
        tmp . append (a)
val = list(tmp)
print("List after removing duplicates :", val)
```

OUTPUT: Enter a list :[1, 2, 4, 8, 16, 32, 64, 128]  
original list is : [1, 2, 4, 8, 16, 32, 64, 128]  
List after removing duplicates : [1, 2, 4, 8, 16, 32, 64, 128]

## 10. TUPLE CREATION USING EXISTING SEQUENCE

```
# Creating tuple from Existing Sequences
T = tuple('HELLO')
print(T)
```

OUTPUT: ('H', 'E', 'L', 'L', 'O')



## 11. TRAVERSING A TUPLE

```
# Traversing a Tuple
T = (2,3,4,5,6)
for a in T:
    print(T[a])
```

OUTPUT:

```
2
3
4
5
6
```

## 12. JOINING TUPLE

```
# Joing Tuple
tpl1 = (1, 3, 4)
tpl2 = (6, 7, 8)
Megatuple = tpl1+tpl2
print(Megatuple)
```

OUTPUT: (1, 3, 4, 6, 7, 8)

## 13. SLICING TUPLE

```
# Slicing Tuple
tpl = (10, 12, 14, 16, 20, 22, 24, 30)
seq = tpl[3:-3]
print(seq)
```

OUTPUT: (16, 20)

## 14. TUPLE DIFFERENTIATOR

```
tup = eval(input ("Enter input for tuple : "))
t1 = tup[ 2:8:2]
t2 = tup[ -3:-9:-2]
t3 = t2[ : : -1]
if t3 == t1 :
    print("The two tuples contain the same elements in reversed order.")
else:
    print("The two tuples contain different elements.")
```

OUTPUT : Enter input for tuple : 1, 2, 3, 4, 5, 6, 7, 8  
The two tuples contain different elements.

## 15. INDEX OF THE MINIMUM ELEMENT

```
#Index of minimum element
tup = eval(input("Enter a tuple : "))
mn = min(tup)
print("Minimum element", mn,"is at index", tup.index(mn) )
```

OUTPUT : Enter a tuple : 23, 24, 25, 9, 10  
Minimum element 9 is at index 3

## 16. MINIMUM AND MAXIMUM MARKS TELLER

```
# Minimum and Maximum marks Teller
Student = eval(input ("Enter input class then marks : "))
print("Mininum marks obtained : ", min(Student[1]))
print("Mininum marks obtained : ", max(Student[1]))
```

OUTPUT : Enter input class then marks : (11, (90, 90, 40,))  
Mininum marks obtained : 40  
Mininum marks obtained : 90

## 17. MIXED KEYS

```
Dict = {1: 'Python', 2: 'For', 3: 'Life'}  
print("\nDictionary with the use of Integer Keys: ")  
print(Dict)  
  
# Creating a Dictionary  
# with Mixed keys  
Dict = {'Name': 'Python', 1: [1, 2, 3, 4]}  
print("\nDictionary with the use of Mixed Keys: ")  
print(Dict)
```

OUTPUT :

```
Dictionary with the use of Integer Keys:  
{1: 'Python', 2: 'For', 3: 'Life'}  
  
Dictionary with the use of Mixed Keys:  
{'Name': 'Python', 1: [1, 2, 3, 4]}
```

## 18. CREATING A NESTED DICTIONARY

```
# Creating a Nested Dictionary  
Dict = {1: 'Python', 2: 'For',  
        3: {'A' : 'Welcome', 'B' : 'To', 'C' : 'Life'}}  
  
print(Dict)
```

OUTPUT :

```
{1: 'Python', 2: 'For', 3: {'A': 'Welcome', 'B': 'To', 'C': 'Life'}}
```

## 19. GET() METHOD

```
# Creating a Dictionary  
Dict = {1: 'Python', 'name': 'For', 3: 'Life'}  
  
# accessing a element using get() method  
print("Accessing a element using get:")  
print(Dict.get(3))
```

OUTPUT :

```
Accessing a element using get:  
Life
```

## 20. ACCESSING ELEMENT USING KEY

```
# Creating a Dictionary
Dict = {'Dict1': {1: 'Python'},
        'Dict2': {'Name': 'For'}}

# Accessing element using key
print(Dict['Dict1'])
print(Dict['Dict1'][1])
print(Dict['Dict2']['Name'])
```

OUTPUT: {1: 'Python'}  
Python  
For

## 21. POP() METHOD

```
# Creating a Dictionary
Dict = {1: 'Computer', 'name': 'For', 3: 'Life'}

# Deleting a key using pop() method
pop_ele = Dict.pop(1)
print('\nDictionary after deletion: ' + str(Dict))
print('Value associated to popped key is: ' + str(pop_ele))
```

OUTPUT: Dictionary after deletion: {'name': 'For', 3: 'Life'}  
Value associated to popped key is: Computer

## 22. LOOPING DICTIONARY

```
#Looping Dictionary
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}

for x, y in thisdict.items():
    print(x, y)
```

OUTPUT: brand Ford  
model Mustang  
year 1964

## 23. COPYING DICTIONARY

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
mydict = thisdict.copy()  
print(mydict)
```

OUTPUT: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

## 24. SORT

```
# Dictionary Methods  
marks = {}.fromkeys(['Math', 'English', 'Science'], 0)  
  
# Output: {'English': 0, 'Math': 0, 'Science': 0}  
print(marks)  
  
for item in marks.items():  
    print(item)  
  
# Output: ['English', 'Math', 'Science']  
print(list(sorted(marks.keys())))
```

OUTPUT: {'Math': 0, 'English': 0, 'Science': 0}  
( 'Math', 0)  
( 'English', 0)  
( 'Science', 0)  
['English', 'Math', 'Science']

## 25. CREATING TUPLE FROM INPUT SEQUENCE

```
# Creating Tuple from Input Sequence  
t1 = tuple(eval(input("Enter input for tuple1: ")))  
t2 = tuple(eval(input("Enter input for tuple2: ")))  
t3 = tuple(eval(input("Enter input for tuple3: ")))  
print("Tuple1:", t1)  
print("Tuple2:", t2)  
print("Tuple3:", t3)
```

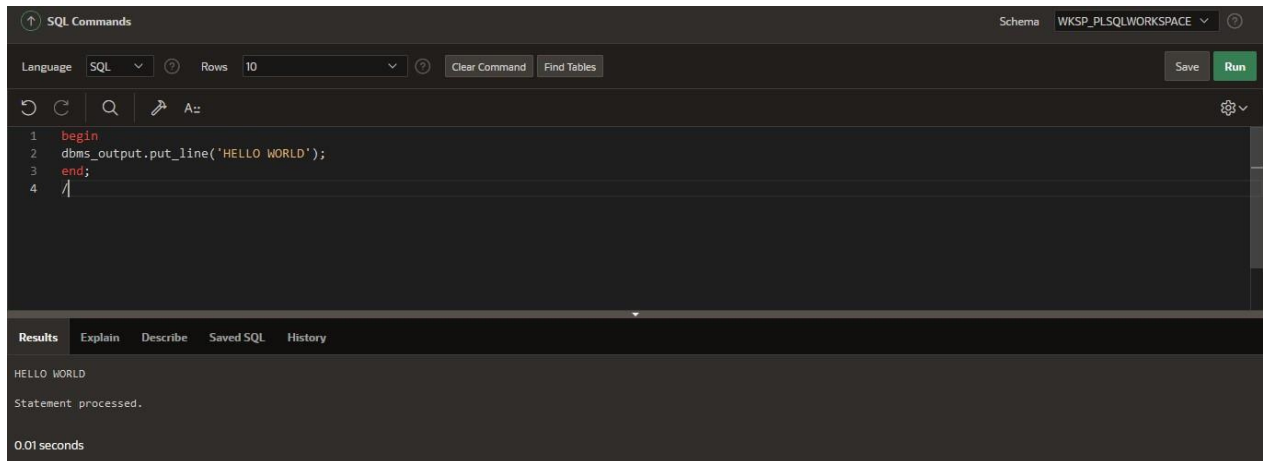
OUTPUT: Enter input for tuple1: "abcdef"  
Enter input for tuple2: 3, 5, 6  
Enter input for tuple3: [11, 12, 13]  
Tuple1: ('a', 'b', 'c', 'd', 'e', 'f')  
Tuple2: (3, 5, 6)  
Tuple3: (11, 12, 13)

# INDEX-2

1. Write a PL/SQL program to print "HELLO WORLD".
2. Write a PL/SQL code for inverting a number. (Example: 1234 – 4321)
3. Write a PL/SQL code to find the greatest number among three with Anonymous blocks.
4. Write a PL/SQL code to calculate the area of a circle where radius takes values from 3 to 7.  
Store the calculated area in Table AREA. The schema of table is given below: AREA (Radius, Area)
5. Write a PL/SQL program to accept a number and find the factorial of the number.
6. Write a PL/SQL program to display the months between two dates of a year

# SQL Programs

1. WRITE A PL/SQL PROGRAM TO PRINT “HELLO WORLD”.



The screenshot shows a SQL Commands window with the following code:

```
1 begin
2 dbms_output.put_line('HELLO WORLD');
3 end;
4 /
```

The Results tab shows the output:

```
HELLO WORLD
Statement processed.
0.01 seconds
```

2. WRITE A PL/SQL CODE FOR INVERTING A NUMBER. (EXAMPLE: 1234 – 4321)



The screenshot shows an SQL Worksheet with the following code:

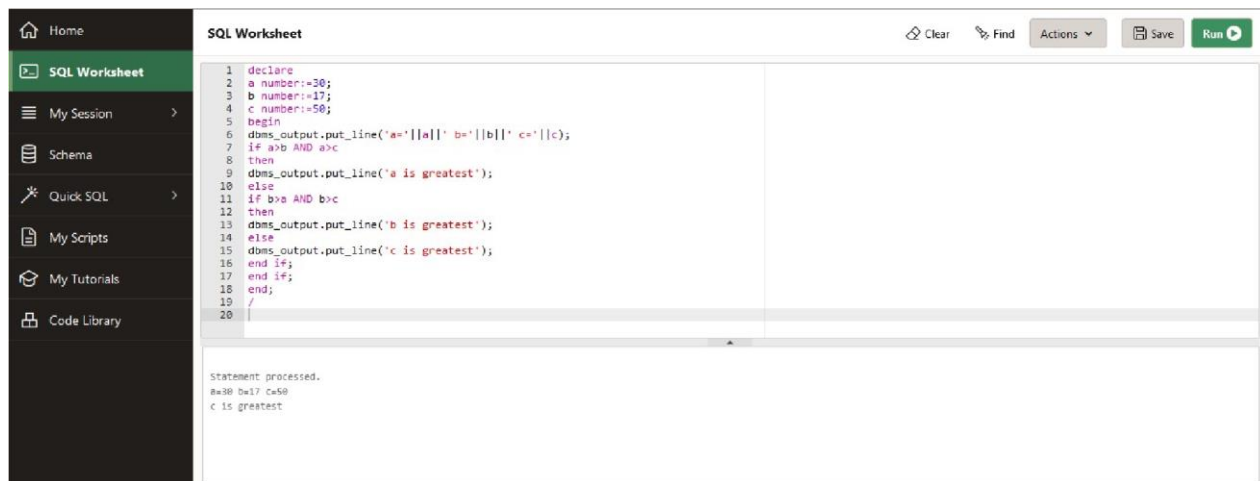
```
1 declare
2 num varchar2(5):='1234';
3 len number(2);
4 revnum varchar2(5);
5 begin
6 len := length(num);
7 for i in reverse 1.. len
8 loop
9 revnum := revnum || substr(num,i,1);
10 end loop;
11 dbms_output.put_line('given number = ' || num);
12 dbms_output.put_line('reverse number = ' || revnum);
13 end;
14 /
```

The Results tab shows the output:

```
Statement processed.
given number =1234
reverse number =4321
```



### 3. WRITE A PL/SQL CODE TO FIND THE GREATEST NUMBER AMONG THREE WITH ANONYMOUS BLOCKS.

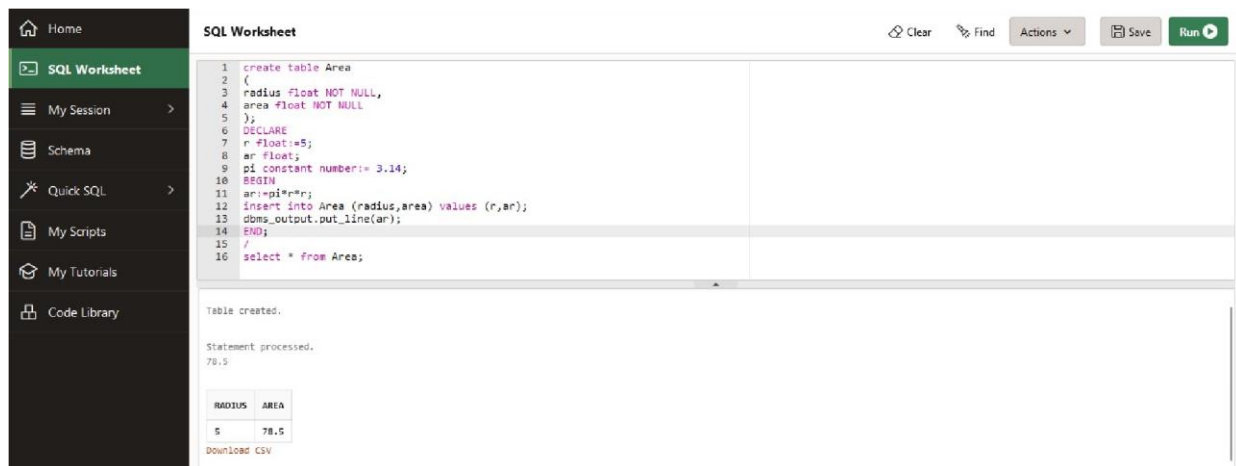


The screenshot shows an SQL Worksheet interface. On the left is a sidebar with navigation options: Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area displays a PL/SQL anonymous block. The code declares three variables: a, b, and c, all of type NUMBER and initialized to 50. It then uses a series of IF-THEN-ELSE statements to compare the values and print the greatest one. The output shows that 'c is greatest'.

```
1 declare
2   a number:=50;
3   b number:=17;
4   c number:=50;
5   begin
6     dbms_output.put_line('a='||a||' b='||b||' c='||c);
7     if a>b AND a>c
8     then
9       dbms_output.put_line('a is greatest');
10    else
11      if b>a AND b>c
12      then
13        dbms_output.put_line('b is greatest');
14      else
15        dbms_output.put_line('c is greatest');
16      end if;
17    end if;
18  end;
19 /
20
```

Statement processed.  
a=50 b=17 c=50  
c is greatest

### 4. WRITE A PL/SQL CODE TO CALCULATE THE AREA OF A CIRCLE WHERE RADIUS TAKES VALUES FROM 3 TO 7. STORE THE CALCULATED AREA IN TABLE AREA. THE SCHEMA OF TABLE IS GIVEN BELOW: AREA (RADIUS, AREA)



The screenshot shows an SQL Worksheet interface. The code creates a table named 'Area' with columns 'radius' and 'area', both of type FLOAT and NOT NULL. It then declares a variable 'r' of type FLOAT, a variable 'ar' of type FLOAT, and a constant 'pi' with a value of 3.14. The code uses a loop to calculate the area for radii from 3 to 7 and inserts the results into the 'Area' table. The output shows the table created and the statement processed, followed by a table with two columns: RADIUS and AREA. The data row shows a radius of 5 and an area of 78.5.

```
1 create table Area
2 (
3   radius float NOT NULL,
4   area float NOT NULL
5 );
6 DECLARE
7   r float:=5;
8   ar float;
9   pi constant number:= 3.14;
10 BEGIN
11   ar:=pi*r*r;
12   insert into Area (radius,area) values (r,ar);
13   dbms_output.put_line(ar);
14 END;
15 /
16 select * from Area;
```

Table created.  
Statement processed.  
78.5

RADIUS	AREA
5	78.5

Download CSV



## 5. WRITE A PL/SQL PROGRAM TO ACCEPT A NUMBER AND FIND THE FACTORIAL OF THE NUMBER.



The screenshot shows the SQL Worksheet interface. The left sidebar contains navigation links: Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main editor area displays a PL/SQL program to calculate the factorial of a number. The program declares two integer variables, 'a' and 'b', both initialized to 5. It then enters a loop that calculates the factorial by multiplying 'b' by 'a' and decrementing 'a' until 'a' reaches 1. The final result is stored in 'b' and printed using 'dbms\_output.put\_line'. The output area at the bottom shows the message 'Statement processed.' followed by the value '120'.

```
1 DECLARE
2   a integer:=5;
3   b integer:=1;
4 BEGIN
5   IF a=0 THEN
6     dbms_output.put_line(b);
7   ELSIF a<0 THEN
8     dbms_output.put_line('Not Possible');
9   ELSE
10    while a>1
11    loop
12      b:=b*a;
13      a:=a-1;
14    end loop;
15    dbms_output.put_line(b);
16  END IF;
17 END;
```

Statement processed.  
120

## 6. WRITE A PL/SQL PROGRAM TO DISPLAY THE MONTHS BETWEEN TWO DATES OF A YEAR



The screenshot shows the SQL Worksheet interface. The left sidebar is the same as in the previous screenshot. The main editor area displays a SQL query that uses the 'MONTHS\_BETWEEN' function to calculate the number of months between two dates: '02-02-2015' and '02-06-2014'. The result is aliased as 'Months' and is selected from the 'DUAL' table. The output area at the bottom shows a table with one column, 'Months', and one row containing the value '11.87096774193548387096774293548387096774'. A 'Download CSV' link is also present.

```
1 SELECT MONTHS_BETWEEN (TO_DATE('02-02-2015','MM-DD-YYYY'), TO_DATE('02-06-2014','MM-DD-YYYY')) "Months" FROM DUAL;
```

Months
11.87096774193548387096774293548387096774

[Download CSV](#)



# INDEX-3

- 1. Use Python to connect to a MySQL database and execute a query.**
- 2. Use Python to execute an SQLite query.**
- 3. Write a program that uses Python and SQL together to create a simple database application.**
- 4. Write a program using Python and SQL to update a record in table.**

# PYTHON AND SQL CONNECTIVITY

## 1. USE PYTHON TO CONNECT TO A MYSQL DATABASE

```
main.py
1 import mysql.connector
2 db_connection = mysql.connector.connect(
3     host="localhost",
4     user="root",
5     passwordd="root"
6 )
7 print(db_connection)
```

Line 5 : Col 11

Debugger x Shell x Console x +

<mysql.connector.connection.MySQLConnection object at 0x000002338A4C6B00>

## 2. CREATING A TABLE

```
main.py
1 import mysql.connector
2 mydb = mysql.connector.connect(host="localhost",user = "root",passwd="",database = "School")
3
4 mycursor= mydb.cursor()
5 mycursor.execute("CREATE TABLE Student (Rollno int(3) Primary key)")
```

Line 5 : Col 69

Debugger x Shell x Console x +

### 3. WRITE A PROGRAM THAT USES PYTHON AND SQL TOGETHER TO CREATE A SIMPLE DATABASE APPLICATION.

```
main.py

1  import pymysql
2  db=pymysql.connect("localhost","root","")
3  c=db.cursor()
4  sql="create database bank;"
5  c.execute(sql)
6  db.close()
```

Line 6 : Col 11

Debugger x Shell x Console x +

Replit: Updating package configuration

--> python3 -m poetry add pymysql

Using version ^1.0.2 for PyMySQL

### 4.WRITE A PROGRAM USING PYTHON AND SQL TO UPDATE A RECORD IN TABLE.

```
import mysql.connector

# Connect to the database
conn = mysql.connector.connect(user='username',
                                password='password',
                                host='host',
                                database='database')

# Create a cursor object
cursor = conn.cursor()

def update_user(id, name, age):
    # Update a user in the table
    cursor.execute("UPDATE users SET name=?, age=? WHERE id=?", (name, age, id))
    conn.commit()

# Update a user
update_user(1, "Alice Smith", 26)

# Close the connection
conn.close()
```

THANK  
you