**Practical No.: 01**

**Assignment Title: Develop programs to understand the control structures of python**

Code:

**1.1 Continue statement:**

# Program to find out even and odd number in between given range using for loop:

for num in range(10): if num % 2 == 0:

print(num, "is even number")

continue

print(num, "is odd number")

**Output:**

0 is even number

2 is even number

4 is even number

6 is even number 8 is even number

# program to print odd numbers from 1 to 10 using while loop:

num = 0

n = int(input("Enter a number in between 1 to 10: ")) if n > 10: print("please enter a number in between 1 to 10") else: while num < n: num += 1 if (num % 2) == 0:

continue print(num) **Output:**

Enter a number in between 1 to 10: 5

1

3

5

**1.2 Break Statement:**

# program to find first 5 multiples of 6

i = 1

n = int(input("Enter a number in between 1 to 10: ")) if n > 10: print("please enter a number in between 1 to 10") else: while i <= 10: print('6 \* ', (i), '=',6 \* i) if i >= n: break i = i + 1

**Output:**

Enter a number in between 1 to 10: 5

6 \* 1 = 6

6 \* 2 = 12

6 \* 3 = 18

6 \* 4 = 24

6 \* 5 = 30

**1.3 Pass Statement:**

#Program to find out odd number in given list

num = [1, 3, 6, 33, 76, 29, 17, 60, 47, 53, 88, 10, 2, 3, 100]

print('Odd numbers are: ') for i in num:

# check if the number is even if i % 2 == 0:

# if even, then pass pass

# print the odd numbers else: print (i)

**Output:**

1

3

33

29

17

47

53

3

**1.4 Conditional Statement (Chained if):**

#program to find out Grade of student:

marks = int(input("Enter the marks: ")) if marks>100:

print("Please enter proper marks!") elif marks > 85 and marks <= 100:

print("Congrats ! you scored grade A ...") elif marks > 60 and marks <= 85: print("You scored grade B + ...") elif marks > 40 and marks <= 60: print("You scored grade B ...") elif (marks > 30 and marks <= 40): print("You scored grade C ...") else: print("Sorry you are fail")

**Output:**

Enter the marks: 70

You scored grade B + ...

**1.5 Nested Loop:**

#program to print Multiplication table up to given number:

n = int(input("Enter any number up to 100:"))

# Iterating over numbers in the range 1 to n for row in range(1,n+1):

# Iterating over numbers in the range 1 to n for col in range(1,n+1):

# Printing the product of row and col print(row\*col, end="\t") print()

**Output:**

Enter any number up to 100: 10

1. 2 3 4 5 6 7 8 9 10
2. 4 6 8 10 12 14 16 18 20
3. 6 9 12 15 18 21 24 27 30
4. 8 12 16 20 24 28 32 36 40
5. 10 15 20 25 30 35 40 45 50
6. 12 18 24 30 36 42 48 54 60
7. 14 21 28 35 42 49 56 63 70
8. 16 24 32 40 48 56 64 72 80
9. 18 27 36 45 54 63 72 81 90
10. 20 30 40 50 60 70 80 90 100

**1.6 Nested Condition:**

a = int(input("Enter 1st number: ")) b = int(input("Enter 2nd number: ")) c = int(input("Enter 3rd number: ")) if(a>b): if(a>c):

print("a is greater")

if(b>a): if(b>c): print("b is greatest")

if(c>a): if(c>b): print("c is greatest")

if(a == b and b == c): print("all are equal")

**Output:**

Enter 1st number: 10

Enter 2nd number: 20

Enter 3rd number: 30 c is greatest

**Practical No :- 02(2.1)**

**Practical Title :- Develop program to learn different types of structures**

**(list, dictionary, tuples)in python**

**Code:-** **2.1 List:**

**2.1.1 Create and display list in python**

Student\_Name=["Nilesh","Prajwal","Dhiraj","Vishal","Nere","Vivek","Ketan"] print(Student\_Name) for i in range(len(Student\_Name)):

print(Student\_Name[i])

**OUTPUT:-**

['Nilesh', 'Prajwal', 'Dhiraj', 'Vishal', 'Nere', 'Vivek', 'Ketan']

Nilesh

Prajwal

Dhiraj

Vishal

Nere

Vivek

Ketan

**2.1.2 List Slicing in python**

*#Print all items*

print(Student\_Name[:])

**OUTPUT**:-

['Nilesh', 'Prajwal', 'Dhiraj', 'Vishal', 'Nere', 'Vivek', 'Ketan']

*#print certain range*

print(Student\_Name[3:5])

**OUTPUT**:-

['Vishal', 'Nere']

*#Print from starting range*

print(Student\_Name[3:])

**OUTPUT**:-

['Vishal', 'Nere', 'Vivek', 'Ketan'] *#print upto given range*

print(Student\_Name[:6])

**OUTPUT**:-

['Nilesh', 'Prajwal', 'Dhiraj', 'Vishal', 'Nere', 'Vivek']

**2.1.3 List Slicing in python**

**1.copy:-**

Copy\_Student\_Name=copy.copy(Student\_Name) for i in range(len(Copy\_Student\_Name)):

print(Copy\_Student\_Name[i])

**OUTPUT**:-

Nilesh

Prajwal

Dhiraj

Vishal

Nere

Vivek

Ketan

**2. deepcopy:-**

Deep\_Copy\_Student\_Name=copy.deepcopy(Student\_Name) for i in range(len(Deep\_Copy\_Student\_Name)):

print(Deep\_Copy\_Student\_Name[i])

**OUTPUT:-**

Nilesh

Prajwal

Dhiraj

Vishal

Nere

Vivek

Ketan

1. **clear:-**

Student\_Name.clear() print(Student\_Name) **OUTPUT:-**

[]

1. **extend:-**

Student\_Name=["Nilesh","Prajwal","Dhiraj","Vishal","Nere","Vivek","Ketan"] Student\_Name.extend(["Nilesh","Kiran","Kunal"]) for i in range(len(Student\_Name)):

print(Student\_Name[i])

**OUTPUT**:-

Nilesh

Prajwal

Dhiraj

Vishal

Nere

Vivek

Ketan

Nilesh

Kiran

Kunal

**5.index:-**

print(Student\_Name.index("Kiran")) **OUTPUT:-**

8

**2.1.4 List Membership in python**

list1=[1,2,3,4,5] list2=[6,7,8,9] for item in list1: if item in list2:

print("Overlapping") else: print("Not Overlapping")

**OUTPUT:-**

Not Overlapping

Not Overlapping

Not Overlapping

Not Overlapping

Not Overlapping

**OR**

x=int(input("Enter a number:")) list=[10,20,30,40,50]

if(x not in list):

print(x,"is NOT present in given list") else: print(x,"is present in given list")

**OUTPUT**:-

Enter a number:30

30 is present in given list **2.1.5 List Deletion in python**

del Student\_Name

print(Student\_Name)

**OUTPUT**:-

Traceback (most recent call last):

File "C:\Users\tanuj\PycharmProjects\secondpract\list.py", line 54, in <module> print(Student\_Name)

^^^^^^^^^^^^

NameError: name 'Student\_Name' is not defined. Did you mean: 'Copy\_Student\_Name'?

**OR**

Student\_Name=["Nilesh","Prajwal","Dhiraj","Vishal","Nere","Vivek","Ketan"] Student\_Name.remove("Vivek") for i in range(len(Student\_Name)):

print(Student\_Name[i])

**OUTPUT**:-

Nilesh

Prajwal

Dhiraj

Vishal

Nere

Ketan

**PRACTICAL NO: 02(2.2)**

**PRACTICAL Title: Develop programs to learn different types of structures (list, dictionary, tuples) in python**

Code:

**2.2 Tuples:**

**2.2.1 Create and display Tuples in python**

Student\_Name = ["Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish", "Ketan"]

print(Student\_Name)

for i in range(len(Student\_Name)): print(Student\_Name[i])

**Output:**

['Nilesh', 'Dhiraj', 'Pankaj', 'Sanket', 'Bhupendra', 'Munish', 'Ketan']

Nilesh

Dhiraj

Pankaj

Sanket

Bhupendra

Munish

Ketan

**2.2.2 Tuples Slicing in python**

Student\_Name = ["Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish", "Ketan"]

print(Student\_Name[3:6])

**Output:**

['Sanket', 'Bhupendra', 'Munish']

**2.2.3 Copy Tuples in python**

Student\_Name = ("Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish", "Ketan")

data = tuple(Student\_Name) print("Copy Student\_Name",data) **Output:**

('Nilesh', 'Dhiraj', 'Pankaj', 'Sanket', 'Bhupendra', 'Munish', 'Ketan')

**2.2.4 Concatenation of Python Tuples**

Student\_Name = ("Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish",

"Ketan") add=("Nilesh","prankaj","prajwal") data=Student\_Name+add;

print("Concatenation of Python Tuples",data) **Output:**

Concatenation of Python Tuples ('Nilesh', 'Dhiraj', 'Pankaj', 'Sanket', 'Bhupendra', 'Munish', 'Ketan', 'Nilesh', 'prankaj', 'prajwal')

**2.2.5 Nesting of Python Tuples**

Student\_Name = (("Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish",

"Ketan"))

print(Student\_Name)

Student\_Name = (("Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish",

"Ketan"),("Ajay MCA"),("pankaj BCA"),("yug LLB")) print(Student\_Name) **Output:**

('Nilesh', 'Dhiraj', 'Pankaj', 'Sanket', 'Bhupendra', 'Munish', 'Ketan')

(('Nilesh', 'Dhiraj', 'Pankaj', 'Sanket', 'Bhupendra', 'Munish', 'Ketan'), 'Ajay MCA', 'pankaj BCA', 'yug LLB')

**2.2.6 Immutable Python Tuples**

Student\_Name = (("Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish", "Ketan"))

Student\_Name[0]=999;

**Output:**

Student\_Name[0]=999;

~~~~~~~~~~~~^^^

TypeError: 'tuple' object does not support item assignment

**2.2.7 Deleting a Tuple**

Student\_Name = (("Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish",

"Ketan")) print(Student\_Name) del(Student\_Name) print("After Deletion") print(Student\_Name) **Output:**

File "F:\PRACTICAL2.2\PRACTICAL2.2.py", line 5, in <module> print(Student\_Name)

^^^^^^^^^^^^

NameError: name 'Student\_Name' is not defined

**2.2.8 Converting list to a Tuple**

def convert(list): return tuple(list)

*# Driver function* list = [1, 2, 3, 4] print(convert(list)) **Output:**

(1, 2, 3, 4)

**2.2.9 Built in Functions of Tuples:**

1. **The len( ) Function**

Student\_Name = (("Nilesh", "Dhiraj", "Pankaj", "Sanket", "Bhupendra", "Munish",

"Ketan"))

print(len(Student\_Name)) **Output:**

7

1. **The count( ) Function**

Student\_Name = ["Sanket","Bhupendra","Munish","Ketan"] print(Student\_Name.count("Sanket")) **Output:**

1

1. **The index( ) Function**

Student\_Name = ["Sanket","Bhupendra","Munish","Ketan"] print(Student\_Name.index("Sanket")) **Output:**

0

1. **The sorted() function**

std\_Roll=(156,222,58,22,56,999) print(sorted(std\_Roll)) **Output:**

[22, 56, 58, 156, 222, 999]

1. **The min(),max(),sum() function**

std\_Roll=(156,222,58,22,56,999) print(min(std\_Roll)) print((max(std\_Roll))) print((sum(std\_Roll)))

**Output:**

22

999

1513

**Assignment No:-02(2.3)**

**Assignment Title :-Develop program to learn different types of structures**

**(list, dictionary, tuples)in python**

**Code:**

**2.3 Dictionary:**

**2.3.1 Create and display Dictionary in python**

*# Creating an empty Dictionary* Dict = {}

print("Empty Dictionary: ") print(Dict)

*# Creating a Dictionary*

*# with dict() method*

Student = dict({1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan'}) print("\nDictionary with the use of dict(): ") print(Student) *#Creating Dictionary:*

Student\_List = {1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal'} print(Student\_List) **Output:**

Empty Dictionary:

{}

Dictionary with the use of dict():

{1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan'}

{1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal'}

**2.3.2 Adding dictionary values**

*# Adding new item in Dictionary*

Student\_List = {1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal'} print(Student\_List) Student\_List[8] = 'Hemangi' print(Student\_List)

**Output:**

{1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal'}

{1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal', 8: 'Hemangi'}

**2.3.3 Accessing Values in Dictionary**

*#Accessing value in dictionary*

Student = {'Name':'Dhiraj Patil','Age':21,'Roll\_No':129} print("Student['Name']:",Student['Name'])

print("Student['Roll\_No']:",Student['Roll\_No'])

**Output:**

Student['Name']: Dhiraj Patil

Student['Roll\_No']: 129

**2.3.4 Print Dictionary using Loop**

*#print dictionary using loop*

Student = {'Name':'Dhiraj Patil','Age':21,'Roll\_No':129} for i,j in Student.items():

print(i,":",j)

**Output:**

Name : Dhiraj Patil

Age : 21

Roll\_No : 129

**2.3.5 Nested Dictionary**

*#Nested Dictionary*

Courses = { "BCA":{

"Years":"Three years course",

"Subjects":"c c++ web-design java....etc"

},

"MCA":{

"Years":"Two years course",

"Subjects":"os web-programming AI python DS ML....etc"

}

}

print(Courses)

print(Courses["BCA"]["Years"])

**Output:**

{'BCA': {'Years': 'Three years course', 'Subjects': 'c c++ web-design java....etc'}, 'MCA': {'Years':

'Two years course', 'Subjects': 'os web-programming AI python DS ML....etc'}}

Three years course

**2.3.6 Updating Dictionary** *#updating dictionary*

Student = {'Name':'Dhiraj Patil','Age':21,'Roll\_No':129} print(Student) Student['Age']=22

print(Student)

**Output:**

{'Name': 'Dhiraj Patil', 'Age': 21, 'Roll\_No': 129}

{'Name': 'Dhiraj Patil', 'Age': 22, 'Roll\_No': 129}

**2.3.7 Delete Dictionary Elements**

*#delete dictionary*

Student = {'Name':'Dhiraj Patil','Age':21,'Roll\_No':129} del Student['Name'] *#remove entry with key 'Name'* print(Student)del Student print(Student) **Output:**

{'Age': 21, 'Roll\_No': 129}

Traceback (most recent call last):

File "C:\Users\tanuj\PycharmProjects\secondpract\dictionary.py", line 55, in <module> print(Student)

^^^^^^^

NameError: name 'Student' is not defined

**2.3.7 Built-in Dictionary methods / functions**

1. **clear( ):-** *#clear()*

Students = {1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7:

'Kunal'} Students.clear() print(Students)

**Output:**

{}

1. **len( ):-**

*#len()*

Student = {'Name':'Dhiraj Patil','Age':21,'Roll\_No':129} print(len(Student))

**Output:**

3

1. **pop( ):-** *#pop()*

Students = {1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7:

'Kunal'} item=Students.pop(1) print(item) print(Students)

**Output:-**

Dhiraj

{2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal'}

1. **popitem( ):-** *#popitem()*

Students = {1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7:

'Kunal'} print(Students) new\_list=Students.popitem() print(Students)

**Output:**

{1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal'}

{1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran'}

1. **keys():-**

*#keys()*

Students = {1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7:

'Kunal'} print(Students) print(Students.keys())

**Output:**

dict\_keys([1, 2, 3, 4, 5, 6, 7])

1. **values():-** *#values()*

Students = {1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7:

'Kunal'} print(Students) new\_list = Students.values() print(new\_list)

**Output:**

{1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal'} dict\_values(['Dhiraj', 'Nilesh', 'Vishal', 'Ketan', 'Wani', 'Kiran', 'Kunal'])

1. **items():-** *#items()*

Students = {1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7:

'Kunal'} print(Students)

print(Students.items())

**Output:**

{1: 'Dhiraj', 2: 'Nilesh', 3: 'Vishal', 4: 'Ketan', 5: 'Wani', 6: 'Kiran', 7: 'Kunal'}

dict\_items([(1, 'Dhiraj'), (2, 'Nilesh'), (3, 'Vishal'), (4, 'Ketan'), (5, 'Wani'), (6, 'Kiran'), (7, 'Kunal')])

**Assignment No :- 03(3.1 Concept of functions in python)**

**Assignment Title :- Develop programs to learn concept of functions scoping, recursion and list mutability.**

**Code:-**

**3.1 Functions in Python:**

**3.1.1 Define a function in python:**

*#define a function* def my\_function(): print("Hello from a function")

*#calling a function*

my\_function()

**Output:-**

Hello from a function **or**

def squeare\_function(num): print(num\*num) n=int(input("Enter a number:")) squeare\_function(n)

**Output:**

Enter a number:10

100

**3.1.2 Calling a function:**

def a\_function( string ):

*"This prints the value of length of string"* return len(string) str = input("Enter a string: ") result = a\_function( str ) *# Calling the function we defined* print( "Length of the string Functions is: ", result)

**Output:-**

Enter a string: tanuja

Length of the string Functions is: 6

**3.1.3 return Statement:**

def square(num): return num \*\* 2

*# Calling function and passing arguments.* print("With return statement") print(square(52))

**Output:-**

With return statement

2704

**3.1.4 The Anonymous Functions**

Addition\_fun = lambda argument1, argument2: argument1 + argument2;

*# Calling the function and passing values*

print( "Value of the function is : ", Addition\_fun( 20, 30 ) ) print( "Value of the function is : ", Addition\_fun( 40, 50 ) )

**Output:-**

Value of the function is : 50

Value of the function is : 90

**3.1.5 Passing a List as an Argument**

def my\_function(fruits): for x in fruits:

print(x) fruits\_List = ["apple", "banana", "cherry"] my\_function(fruits\_List)

**Output:-**

apple

banana cherry

**PRACTICAL Title: Develop program to learn concept of function scoping ,recursion and list mutability.**

**Code:-**

**3.2 Function Scoping in python**

**3.2.1 Local Scope:**

def cube(item): result=item\*\*3 def display():

print("the cube is",result) display()

element = int(input("Enter thr numner")) cube(element)

**Output:**

Enter thr numner5 the cube is 125

**Now try to access the result outside the function**

def cube(item): result=item\*\*3 def display():

print("the cube is",result) display()

element = int(input("Enter thr numner")) cube(element) print(result) **output:**

Enter thr numner5 the cube is 125

Traceback (most recent call last):

File "F:\PRACTICAL2.2\PRACTICAL2.2.py", line 9, in <module>

print(result)

^^^^^^

NameError: name 'result' is not defined

**3.2.2 Global Scope:**

result = 0 def cube(item):

print("the test result" , result ) return item\*\*3

def display\_result():

element = int(input("Enter thr number")) result = cube(element) print("the cube of given number is", result) display\_result()

**Output:**

Enter thr number5

the test result 0 the cube of given number is 125

**Practical No.:- 03 (3.3 Concept of Mutability and Immutability in Python)**

**Assignment Title: Develop programs to learn concept of functions scoping, recursion and list mutability.**

Code:

**3.3 Mutability and Immutability in Python:**

**3.3.1 Mutability of List:**

my\_lsit = ["Nilesh","ajay","pankaj"] print(my\_lsit) my\_lsit[0]="Darshan" print(my\_lsit) **Output:**

['Nilesh', 'ajay', 'pankaj'] ['Darshan', 'ajay', 'pankaj']

**3.3.2 Mutability of Dictionary:**

my\_dect ={1:"Nilesh",

2:"Ajay",

3:"Bharat",

4:"Vaibhav",

5:"krunal"

}

print("dictory before updateing",my\_dect) my\_dect[1]="Ashavin"

print("dictory after updateing",my\_dect)  **Output:**

dictory before updateing {1: 'Nilesh', 2: 'Ajay', 3: 'Bharat', 4: 'Vaibhav', 5: 'krunal'} dictory after updateing {1: 'Ashavin', 2: 'Ajay', 3: 'Bharat', 4: 'Vaibhav', 5: 'krunal'}

**3.3.3 Immutability of Tuples:**

my\_tuple=(1,2,3) my\_tuple[1]="Nilesh" **Output:**

my\_tuple[1]="Nilesh"

~~~~~~~~^^^

TypeError: 'tuple' object does not support item assignment

**3.3.4 IMMUTABILITY OF NUMBER:**

a=96

print(id(a)) a=96

print(id(a)) **Output:**

140722871467784

140722871467784

3.3.5 **IMMUTABILITY OF STRING:**

a="NILESH"

print(id(a)) a="PATIL"

print(id(a))

Output:

1982962398320

1982960913072

**PRACTICAL NO . : - 04(4.1)**

**PRACTICAL TITLE : DEVELOP PROGRAMS TO UNDERSTAND OBJECT ORIENTED PROGRAMMING USING PYTHON. (CLASS AND OBJECT).**

**Code:-**

**4.1 Class and object in Python:**

**4.1.1 Creating class:**

class Employee:

def \_\_init\_\_(self,name,id):

self.id=id self.name=name def display(self): print("ID:",self.id,"Name:",self.name)

**4.1.2 Creating Object(Instance):**

class Employee:

def \_\_init\_\_(self,name,id):

self.id=id self.name=name def display(self):

print("ID:",self.id,"Name:",self.name)

emp1=Employee("Nilesh",45) emp2=Employee("Ajay",95)

emp1.display() emp2.display()

**Output:**

ID: 45 Name: Nilesh

ID: 95 Name: Ajay

**PRACTICAL NO.: 04(4.2)**

**PRACTICAL TITLE: DEVELOP PROGRAMS TO UNDERSTAND OBJECT ORIENTED PROGRAMMING USING PYTHON (INHERITANCE).**

**Code:-**

**4.2 Inheritance in Python:**

**4.2.1 Single Inheritance:**

class parent: def fun1(self):

print("Hello parent") class child(parent): def fun2(self):

print("Hello child") test = child() test.fun1() test.fun2() **Output:**

Hello parent

Hello child

**4.2.2: Multiple Inheritance:**

class parent1: def fun1(self):

print("Hello parent 1") class parent2: def fun2(self):

print("Hello parent 2") class parent3: def fun3(self):

print("Hello parent 3") class child(parent1,parent2,parent3):

def fun4(self):

print("Hello child") test = child() test.fun1() test.fun2() test.fun3() test.fun4() print(child.\_\_mro\_\_) **Output:**

Hello parent 1

Hello parent 2

Hello parent 3

Hello child

(<class '\_\_main\_\_.child'>, <class '\_\_main\_\_.parent1'>, <class '\_\_main\_\_.parent2'>, <class '\_\_main\_\_.parent3'>, <class 'object'>)

**4.2.3: Multilevel Inheritance:**

class grandparent: def func1(self):

print("Hello Grandparent") class parent(grandparent): def func2(self):

print("Hello parent") class child(parent): def func3(self): child().func1() child().func2() print("Hello child")

test=child() test.func3() **Output:**

Hello Grandparent

Hello parent

Hello child

**4.2.4: Hierarchical Inheritance:**

class parent1: def func1(self):

print("Hello Parents") class parent2: def fun2(self):

print("Hello parents") class child1(parent1): def func3(self):

print("Hello Child 1") class child2(child1,parent2): def func4(self):

print("Hello Child2")

test1 = child1() test2 = child2() test1.func1() test1.func3()

test2.func1() test2.fun2() test2.func3() test2.func4()

**Output:**

Hello Parents

Hello Child 1

Hello Parents

Hello parents

Hello Child 1

Hello Child2

**4.2.5: Hybrid Inheritance:**

class parents: def func1(self): print("Hello parents") class child1(parents): def func2(self): print("Hello Child 1") class child2(parents): def func3(self): print("Hello Child 2")

test1 = child1() test2 = child2()

test1.func1() test1.func2()

test2.func1() test2.func3() **Output:**

**Hello parents**

**Hello Child 1**

**Hello parents**

**Hello Child 2**

**Assignment No.: 04(4.3)**

**Assignment Title: Develop programs to understand object oriented programming using python (Overloading).**

Code:

**4.3 Overloading in Python:**

class areaClass:

def area(self,a,b=None,c=None,d=None): #when a and c are passed as arguments if a!=None and b!=None and a!=b and a!=c:

print("Area of the triangle",(0.5\*a\*b)) #when a,b,c and d are passed as arguments elif(b!=None and c!=None and d!=None and a==b and a==c):

print("Area of the square",(a\*c))

elif(b==None and c==None and d==None):

print("Arear of Circle: ", (3.14\*(a\*a)))

elif(a==None and b==None and c==None and d==None):

print("Enter more numbers")

else: if(a==c):

print("Area of the rectangle",(a\*b))

else:

print("Area of the rectangle",(a\*c))

obj=areaClass() obj.area(19,5,19) #Triangle obj.area(20,20,20,20) #Square obj.area(20,40,20,40) #Rectangle obj.area(6) #Circle **Output:**

Area of the rectangle 95

Area of the square 400

Area of the rectangle 800

Arear of Circle: 113.04

**PRACTICAL TITLE : DEVELOP PROGRAMS TO UNDERSTAND OBJECT ORIENTED PROGRAMMING USING PYTHON (OVERRIDING).**

**Code:-**

**4.2 Overriding in Python:**

*# Parent class* class Shape: *# properties* data1 = "abc" *# function no\_of\_sides* def no\_of\_sides(self):

print("My sides need to be defined. I am from shape class.")

*# function two\_dimensional* def two\_dimensional(self):

print("I am a 2D object. I am from shape class")

class Square (Shape):

data2 = "XYZ"

def no\_of\_sides (self):

print("I have 4 sides. I am from Square class")

def color(self):

print("I have teal color. I am from Square class.")

*# Create an object of Square class* sq = Square()

*# Override the no\_of\_sides of parent class* sq.no\_of\_sides()

*# Will inherit this method from the parent class* sq.two\_dimensional() *# It's own method color* sq.color() **Output:**

I have 4 sides. I am from Square class I am a 2D object. I am from shape class

I have teal color. I am from Square class.

**Assignment No.: 05(5.1)**

**Assignment Title: Develop programs for data structure algorithms using python – searching, sorting and hash tables.(Sorting)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code:

**5.1 Sorting in Python:**

**5.1.1 Bubble Sort:**

# Python3 program for Bubble Sort Algorithm Implementation def bubbleSort(arr):

n = len(arr)

# For loop to traverse through all

# element in an array for i in range(n): for j in range(0, n - i - 1):

# Range of the array is from 0 to n-i-1

# Swap the elements if the element found # is greater than the adjacent element if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

# Driver code

# Example to test the above code

arr = [2, 1, 100, 23, 25, 50]

bubbleSort(arr)

print("Sorted array is:") for i in range(len(arr)): print("%d" % arr[i])

**Output:**

Sorted array is:

1

2

23

25

50

100

**5.1.2 Selection Sort:**

def selectionSort(array, size): for step in range(size): min\_idx = step

for i in range(step + 1, size): if array[i] < array[min\_idx]:

min\_idx = i

(array[step], array[min\_idx]) = (array[min\_idx], array[step])

# Initializing list1 list1 = [] n = int(input("Enter size: ")) for i in range(0, n): print("Enter Element: ") ele = int(input())

# adding the element

list1.append(ele)

# Function Call selectionSort(list1, n) print('Sorted Array in Ascending Order:') print(list1)

**Output:**

Enter size: 5

Enter Element:

10

Enter Element:

20

Enter Element: 30

Enter Element:

50

Enter Element:

40

Sorted Array in Ascending Order:

[10, 20, 30, 40, 50]

**5.1.2 Insertion Sort:**

def insertionSort(array): for step in range(1, len(array)):

key = array[step]

j = step - 1

while j >= 0 and key < array[j]:

array[j + 1] = array[j] j = j - 1

array[j + 1] = key

# Initializing list1 list1 = [] n = int(input("Enter size: ")) for i in range(0, n): print("Enter Element: ") ele = int(input())

# adding the element

list1.append(ele)

# Function call insertionSort(list1) print('Sorted Array in Ascending Order:') print(list1)

**Output:**

Enter size: 5

Enter Element:

10

Enter Element:

30

Enter Element:

20

Enter Element:

50

Enter Element:

40

Sorted Array in Ascending Order:

[10, 20, 30, 40, 50]

**Assignment No.: 05(5.2)**

**Assignment Title: Develop programs for data structure algorithms using python – searching, sorting and hash tables. (Searching)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Code:-**

**5.2 Searching in Python:**

**5.2.1 Linear Search:**

def linear\_Search(list1, n, key): # Searching list1 sequentially for i in range(0, n): if (list1[i] == key):

return i+1

return -1

list1 = []

n = int(input("Enter size: ")) for i in range(0, n): print("Enter Element: ") ele = int(input())

# adding the element

list1.append(ele)

Key = int(input("Enter Key: ")) item = linear\_Search(list1, n, Key) if(item != -1):

print("Item is at: ", item) else: print("Item is Not found")

**Output:**

Enter size: 3 Enter Element:

10

Enter Element:

50

Enter Element:

42

Enter Key: 50

Item is at: 2

**5.2.2 Binary Search:**

def binary\_search(list1, n): low = 0 high = len(list1) - 1

mid = 0

while low <= high: # for get integer result mid = (high + low) // 2

# Check if n is present at mid if list1[mid] < n: low = mid + 1

# If n is greater, compare to the right of mid elif list1[mid] > n: high = mid - 1

# If n is smaller, compared to the left of mid else:

return mid

# element was not present in the list, return -1 return -1

# Initializing list1 list1 = [] n = int(input("Enter size: ")) for i in range(0, n): print("Enter Element: ") ele = int(input()) # adding the element list1.append(ele)

n = int(input("Enter item: "))

# Sorting list for i in range(len(list1) - 1):

for j in range(0, len(list1) - i - 1):

if list1[j] > list1[j + 1]: temp = list1[j] list1[j] = list1[j + 1]

list1[j + 1] = temp

print("sorted list: ", list1)

# Function call

result = binary\_search(list1, n)

# Results if result != -1:

print("Element is present at index: ", str(result)) else:

print("Element is not present in list1") **Output:**

Enter size: 4 Enter Element:

10 Enter Element:

30 Enter Element:

50 Enter Element:

40

Enter item: 10

sorted list: [10, 30, 40, 50] Element is present at index: 0

**Practical No:- 06**

**Practical Name:- Develop programs to learn regular expressions using python.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code:-** import re s = 'GeeksforGeeks: A computer science portal for geeks' match = re.search(r'portal', s) print('Start Index:', match.start()) print('End Index:', match.end())

## OutPut:-

Start Index: 34

End Index: 40

-----------------------------------------------------------------------------------------------------

# \ – Backslash:-

import re

s = 'geeks.forgeeks'

# without using \ match = re.search(r'.', s) print(match)

# using \

match = re.search(r'\.', s)

print(match)

## OutPut:-

<re.Match object; span=(0, 1), match='g'>

<re.Match object; span=(5, 6), match='.'>

-----------------------------------------------------------------------------------------------------

# [] – Square Brackets:-

import re

string = "The quick brown fox jumps over the lazy dog" pattern = "[a-m]"

result = re.findall(pattern, string)

print(result)

## Output:-

['h', 'e', 'i', 'c', 'k', 'b', 'f', 'j', 'm', 'e', 'h', 'e', 'l', 'a', 'd', 'g'

--------------------------------------------------------------------------------------------------

## ^ – Caret:-

import re

# Match strings starting with "The"

regex = r'^The'

strings = ['The quick brown fox', 'The lazy dog', 'A quick brown fox'] for string in strings:

if re.match(regex, string):

print(f'Matched: {string}')

else:

print(f'Not matched: {string}')

### Output:-

Matched: The quick brown fox

Matched: The lazy dog

Not matched: A quick brown fox

--------------------------------------------------------------------------------------------------

## $ – Dollar:-

import re

string = "Hello World!" pattern = r"World!$"

match = re.search(pattern, string) if match:

print("Match found!")

else:

print("Match not found.")

### Output:-

Match found! --------------------------------------------------------------------------------------------------

## . – Dot:-

import re

string = "The quick brown fox jumps over the lazy dog." pattern = r"brown.fox"

match = re.search(pattern, string) if match:

print("Match found!")

else:

print("Match not found.")

### Output:-

Match found!

**PRACTICAL NO:-07**

**PRATICAL NAME:- Demonstrate implementation of the Anonymous Function Lambda.**

**Code:-**

**9.Anonymous Function Lambda in Python:**

def sum(num1, num2): return(num1+num2) sum\_lambda = lambda num1,num2:num1+num2

num1=int(input("enter 1st number for addition")) num2=int(input("enter 2nd number foraddition")) print(sum(num1,num2)) print(sum\_lambda(num1,num2)) **output:-** enter 1st number for addition10 enter 2nd number foraddition27

37

37

**PRACTICL NO : 08**

**PRACTICAL TITLE:-DEMONSTRATE IMPLEMENTATION FUNCTIONAL**

**PROGRAMMING TOOL SUCH AS FILTER AND REDUCE**

**Code:-**

**10.1.Filter()function in python:**

nums = (10,3,192,55,20,77,91)

#creating a function that return true if the number is Divisible by 5 #%here is the modules operator to check the reminder when divided by5

def divisible(i):

return True if i%5==0 else False #creating the filter function divisible\_by\_5= filter(divisible, nums) #to print the class of returned objejt print(type(divisible\_by\_5)) #print the list of filter numbers print(tuple(divisible\_by\_5)) **output:**

<class 'filter'>

(10, 55, 20)

**Simple for loop Vs. Filter Function**: #making an empty list to store valid ages valid\_ages=[] #gives list of ages ages=[12,21,18,23,9,55,82,69,14]

#defing function to test if enterd age is above 18 or not def eligible(i): for age in i: if age>= 18:

valid\_ages.append(age) #calling the function on ages eligible(ages) #print results print(valid\_ages) **output:**

[21, 18, 23, 55, 82, 69]

**10.2.Reduce()Function in python:** from functools import reduce nums =[1,2,3,4] ans= reduce(lambda x,y:x+y,nums) print(ans) **output:- 10**

**10.3.map()function in python:** import math #our transformation function def square\_root(n): return math.sqrt(n)

#we calc square root of all number using map() numbers =[16,36,100,4] result=map(square\_root,numbers)#get the mao object

#print()

#print(result)#we will get our maop object converted\_result=list(result) print(converted\_result) **output:**

[4.0, 6.0, 10.0, 2.0]

**Practical No:- 10**

**Practical Name: Develop programs to learn GUI programming using Tkinter.**

-----------------------------------------------------------------------------------------------------------

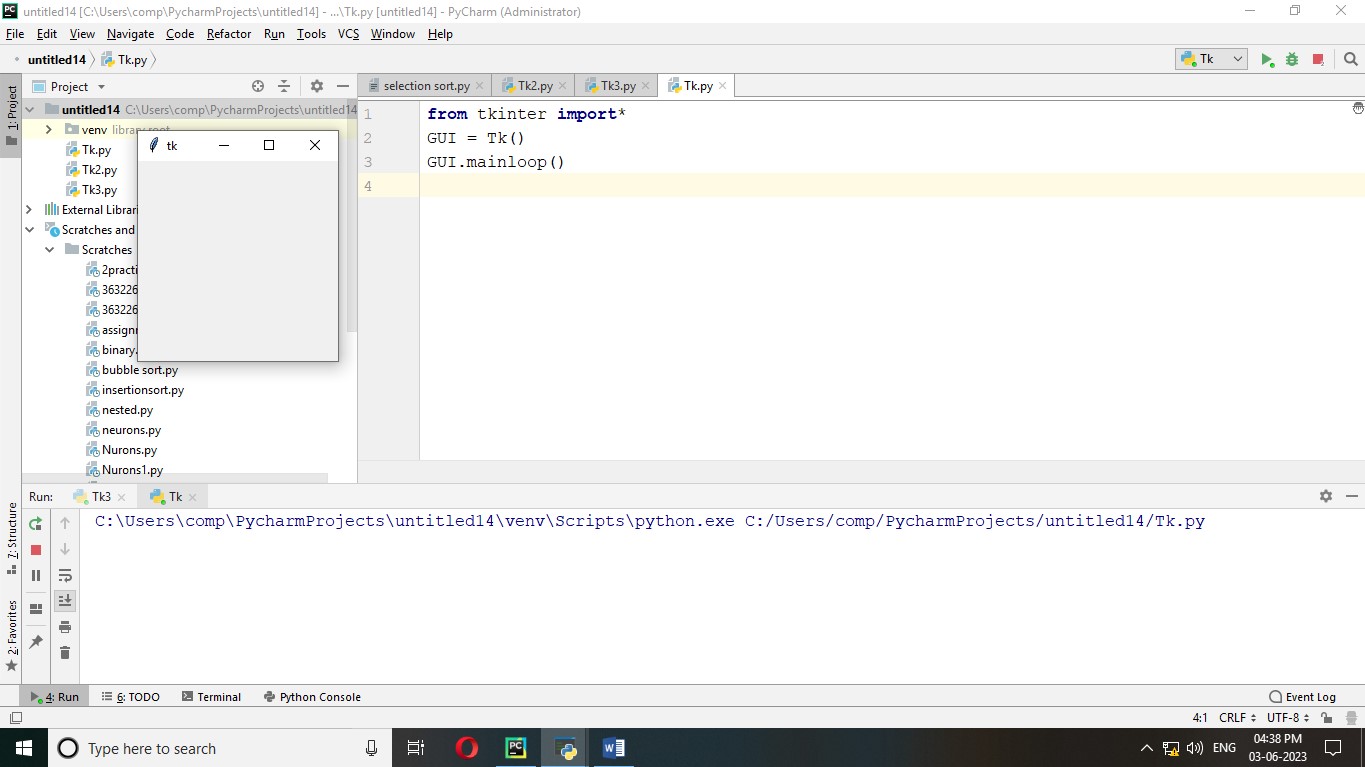
**Code:-**

**Create simple Application Window**

from tkinter import \* GUI = Tk()

GUI.mainloop()

**Output:**



**Application Window with size**

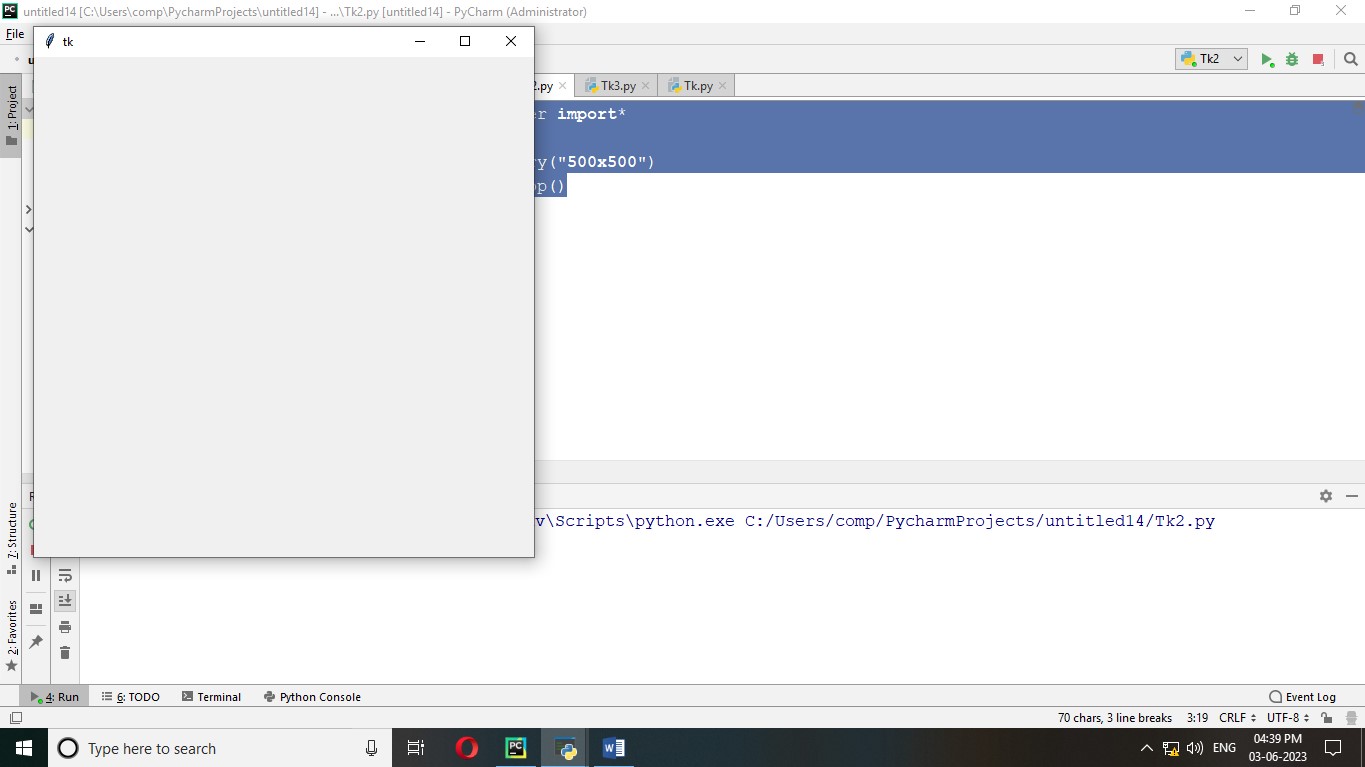
from tkinter import \* GUI = Tk()

# If you want to provide size of the window

GUI.geometry("500x500")

GUI.mainloop()

**Output:**



**Application Window to get information from user**

from tkinter import \* GUI = Tk()

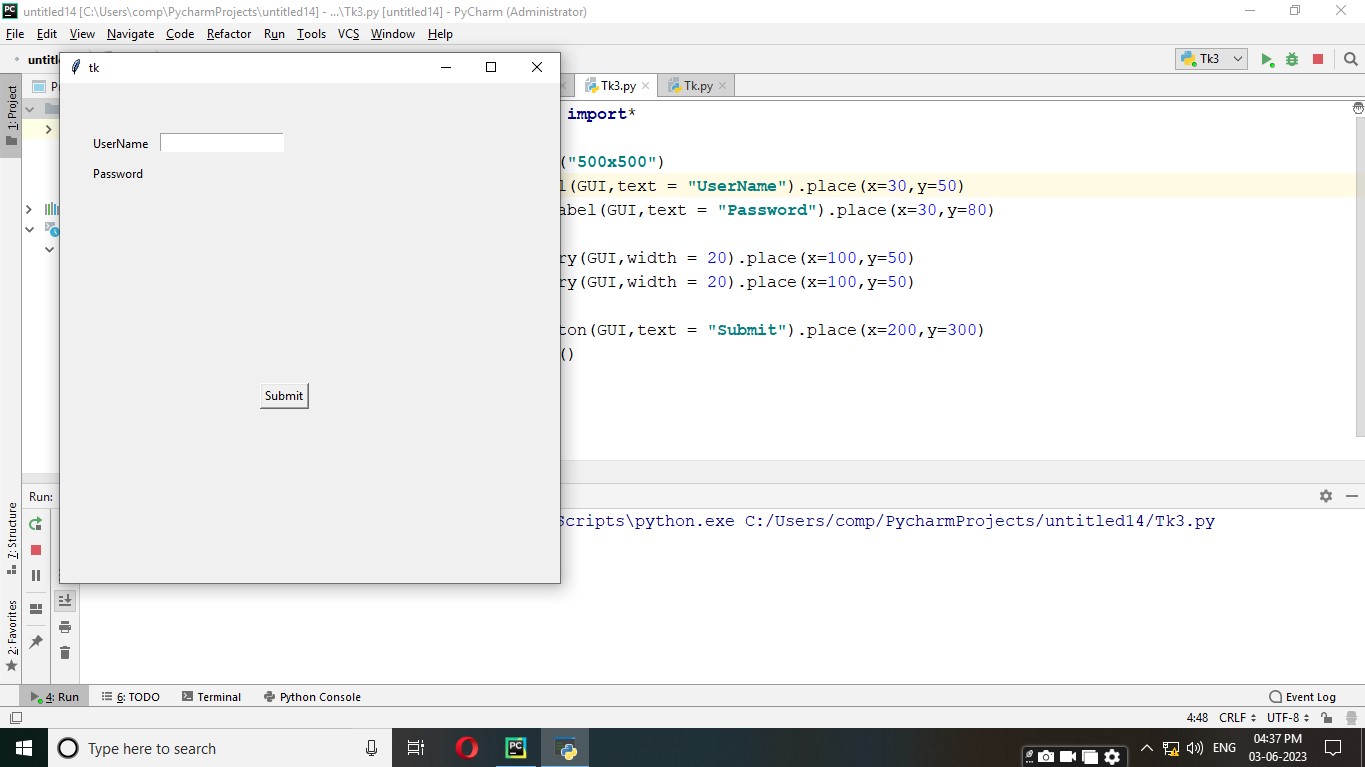
# If you want to provide size of the window GUI.geometry("500x500")

# If you want to add labels uname = Label(GUI, text = "Username").place(x = 30, y = 50) password = Label(GUI, text = "Password").place(x = 30, y = 80)

# Add Textbox txtbx1 = Entry(GUI, width = 20).place(x = 100, y = 50) txtbx2 = Entry(GUI, width = 20).place(x = 100, y = 80)

# Add Button on window sbmitbtn = Button(GUI, text = "Submit").place(x = 220, y = 300) GUI.mainloop()

**Output:**



**Practical N0:-11**

**Assignment Name:-Demonstrate database connectivity using MySql**

**Create connection with mysql Workbench** import mysql.connector conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29",database="test\_py charm")

my\_cur = conn.cursor() conn.commit() conn.close()

print("Connected")

**Output:-**

Connected

**Create table in pycharm with mysql:-** import mysql.connector

conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29", database="test\_pycharm")

my\_cur = conn.cursor()

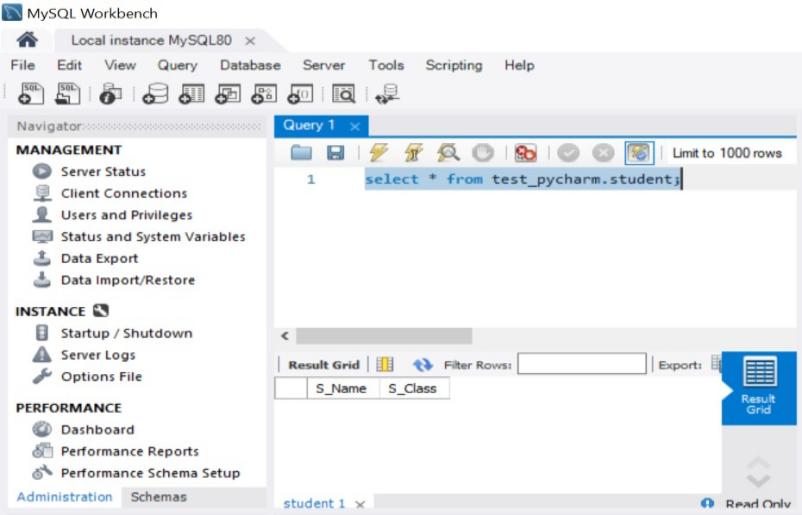
my\_cur.execute("CREATE TABLE Student (S\_Name VARCHAR(255), S\_Class VARCHAR(255))")

conn.commit() conn.close()

print("Connected")

**Output:-**

Connected



**Show tables in current database:-**

import mysql.connector

conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29", database="test\_pycharm")

my\_cur = conn.cursor()

my\_cur.execute("SHOW TABLES")

for x in my\_cur: print(x)

conn.close()

**Output:-**

('student',)

**Apply Primary Key:-** import mysql.connector

conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29", database="test\_pycharm")

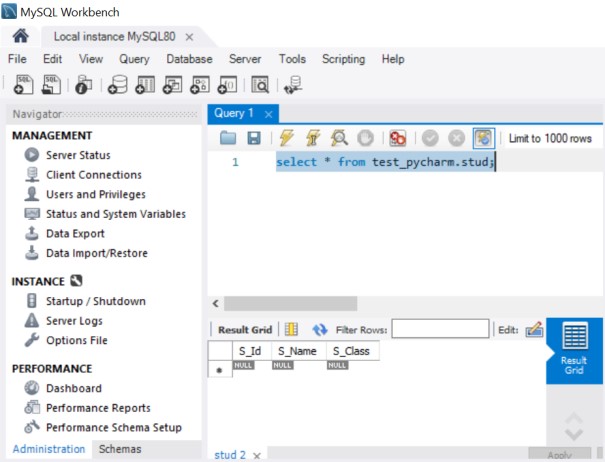
my\_cur = conn.cursor()

my\_cur.execute("CREATE TABLE Stud(S\_Id int AUTO\_INCREMENT primary key,S\_Name

VARCHAR(255),S\_Class VARCHAR(255))")

conn.close()

**Output:-**



**Alter table:-**

import mysql.connector conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29", database="test\_pycharm")

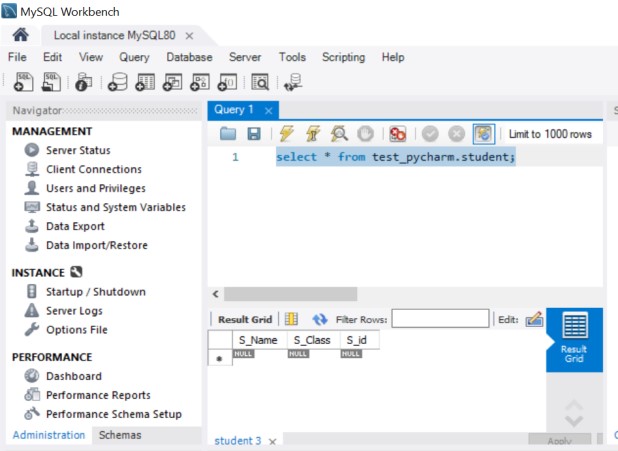
my\_cur = conn.cursor()

my\_cur.execute("ALTER TABLE student ADD COLUMN S\_id INT AUTO\_INCREMENT PRIMARY KEY")

print("Table Altered") conn.close()

**Output:-**

Table Altered



**Insert records:-** import mysql.connector

conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29", database="test\_pycharm")

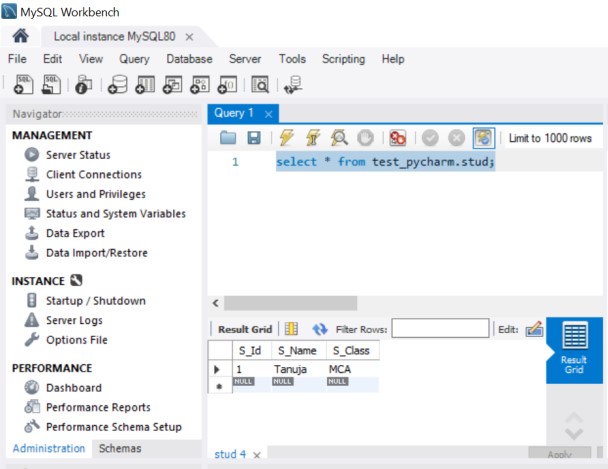
my\_cur = conn.cursor()

sql = "INSERT INTO stud(S\_id,S\_Name,S\_Class) VALUES (%s,%s, %s)" val =("1","Tanuja", "MCA") my\_cur.execute(sql, val) conn.commit() print("Done")

conn.close()

**Output:-**

Done



**Insert multiple records:-** import mysql.connector conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29", database="test\_pycharm")

my\_cur = conn.cursor()

sql = "INSERT INTO stud(S\_id,S\_Name,S\_Class) VALUES (%s,%s, %s)" val = [

("2","Yojana","MCA"),

("3","Hemangi", "MBA"),

("4","Neha", "BCA"),

("5","Koyal","M-COM")

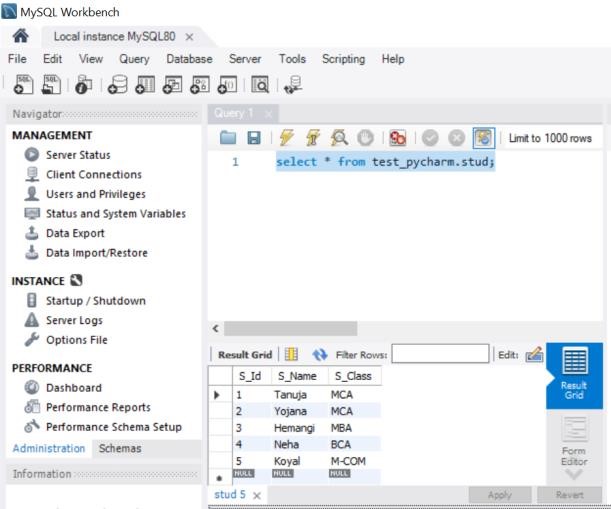
]

my\_cur.executemany(sql, val) conn.commit() print("Done")

conn.close()

**Output:-**

Done



**Select statement (show records)**

import mysql.connector

conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29", database="test\_pycharm")

my\_cur = conn.cursor()

my\_cur.execute("SELECT \* FROM test\_pycharm.stud;")

Records = my\_cur.fetchall() for x in Records:

print(x)

conn.close()

**Output:-**

**s**

(1, 'Tanuja', 'MCA')

(2, 'Yojana', 'MCA')

(3, 'Hemangi', 'MBA') (4, 'Neha', 'BCA') (5, 'Koyal', 'M-COM')

**Using where statement:-** import mysql.connector

conn = mysql.connector.Connect(host="localhost",username="root",password="Tanuja@29", database="test\_pycharm")

my\_cur = conn.cursor()

Query = "SELECT \* FROM stud WHERE S\_id =1"

my\_cur.execute(Query)

records = my\_cur.fetchall() for x in records:

print(x)

conn.close()

**Output:-**

(1, 'Tanuja', 'MCA')