Python Lab Manual

# **1. Exploring basics of python like data types (strings, list, array, dictionaries, set, tuples) and control statements.**

Problem Statement:

Write a python program to create variables of data types strings, list, array, dictionaries, set,

tuples. Use for and while loops wherever applicable to print their values.

**Program**

import array as arr

myStr = 'Sukritee Singh' # String

myList = ['abc','xyz','uvw'] # list

myArr = arr.array('i',[1,2,3,4,5]) # Array

myDict = {'Ten': 10, 'Twenty': 20, 'Thirty': 30} #Dictionary

mySet = {"apple", "banana", "cherry"} #Set

myTuple = (5,10,15,20,25) #Tuple

print("String :- ",end=" ")

print(myStr, "\n")

print("List :- ",end=" ")

for i in range(len(myList)):

print(myList[i], end=" ")

print("\n\nArray :- ",end=" ")

for x in myArr:

print(x,end=" ")

print("\n\nTuple :- ",end=" ")

for i in range(len(myTuple)):

print(myTuple[i], end=" ")

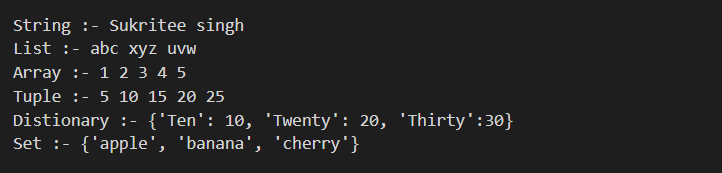
print("\n\nDictionary :- ",end=" ")

print(myDict, "\n")

print("Set :- ",end=" ")

print(mySet, "\n")

**Output**



# **2. Creating functions, classes and objects using python. Demonstrate exception handling and inheritance.**

**Problem Statement:**

Create a Python class named &#39;Member&#39; having the following members:

Data members

1 - Name

2 - Age

3 - Mobile number

4 - Address

5 - Salary

It also has a method named &#39;printSalary&#39; which prints the salary of the members.

Two classes Employee and Manager; inherits the Member class. The &#39;Employee and Manager classes have data members specialization and department

respectively. Now, assign name, age, mobile number, address and salary to an

employee and a manager by making an object of both of these classes and print the

same. Program should throw an exception if the phone number has digits less than 10.

**Program**

import re

class Member:

def \_\_init\_\_(self, name, age, number, address, salary):

self.name = name

self.age = age

self.number = number

self.address = address

self.salary = salary

class Employee(Member):

def \_\_init\_\_(self, specialization, name, age, number, address, salary):

super().\_\_init\_\_(name, age, number, address, salary)

self.specialization = specialization

def printValues(self):

print(self.name, "\n", self.age, "\n", self.number, "\n", self.address, "\n", self.specialization, "\n",

self.salary)

class Manager(Member):

def \_\_init\_\_(self, department, name, age, number, address, salary):

super().\_\_init\_\_(name, age, number, address, salary)

self.department = department

def printValues(self):

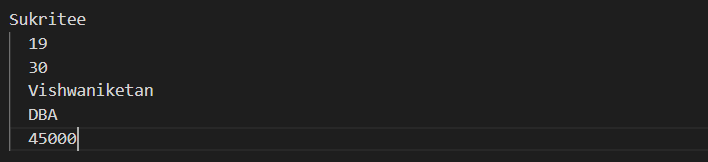
print(self.name, "\n", self.age, "\n", self.number, "\n", self.address, "\n", self.department, "\n",

self.salary)

object1 = Employee("DBA", "Sukritee", 19, 30, "Vishwaniketan", 45000)

object1.printValues()

**Output**



# **3. Exploring Files and directories**

**Problem Statement:**

a. Python program to append data to existing file and then display the entire file

b. Python program to count number of lines, words and characters in a file.

c. Python program to display file available in current directory

**Program**

import os

import sys

lines = 0

words = 0

charecter = 0

file1 = open("myfile.txt", "w")

L = ["This is Sukritee Singh \n", "I'm in Second Year \n"]

file1.writelines(L)

file1.close()

file2 = open("myfile.txt", "r")

for x in file2:

x = x.strip("\n")

wordsS = x.split()

lines += 1

words = len(wordsS)

charecter = len(x)

file2.close()

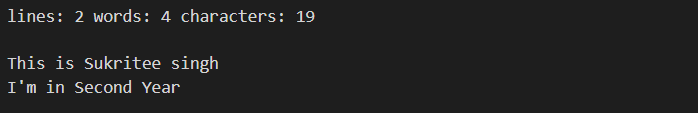
print("lines:", lines, "words:", words, "characters:", charecter)

print("\n")

with open(os.path.join(sys.path[0], "myfile.txt"), "r") as f:

print(f.read())

**Output**



# **4. Creating GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes.**

**Problem Statement:**

Create GUI in python which will allow user to enter name, email address, mobile number,

gender, computer languages known and one button with ‘Submit’ as caption. After clicking on submit button all values entered by user should get displayed in label.

**Program**

from tkinter import \*

class MyWindow:

def \_\_init\_\_(self, win):

self.lbl1 = Label(win, text='Name')

self.lbl2 = Label(win, text='Email Address')

self.lbl3 = Label(win, text='Number')

self.lbl4 = Label(win, text='Gender')

self.lbl5 = Label(win, text='Computer Languages known :')

self.lbl6 = Label(win, text='Result')

self.t1 = Entry(bd=3)

self.t2 = Entry()

self.t3 = Entry()

self.t4 = Entry()

self.t5 = Entry()

self.t6 = Entry()

self.btn1 = Button(win, text='Submit')

self.lbl1.place(x=100, y=50)

self.t1.place(x=300, y=50)

self.lbl2.place(x=100, y=100)

self.t2.place(x=300, y=100)

self.lbl3.place(x=100, y=150)

self.t3.place(x=300, y=150)

self.lbl4.place(x=100, y=200)

self.t4.place(x=300, y=200)

self.lbl5.place(x=100, y=250)

self.t5.place(x=300, y=250)

self.b1 = Button(win, text='Submit', command=self.display)

self.b1.place(x=100, y=400)

self.lbl6.place(x=100, y=300)

self.t6.place(x=300, y=300)

def display(self):

self.t6.delete(0, 'end')

name = str(self.t1.get())

email = str(self.t2.get())

number = int(self.t3.get())

gender = str(self.t4.get())

languages\_known = str(self.t5.get())

res = "name:",name,"\nemail:",email,"\nnumber:",number,"\ngender:",gender,"\nlanguages:",languages\_known

self.t6.insert(END, res)

window = Tk()

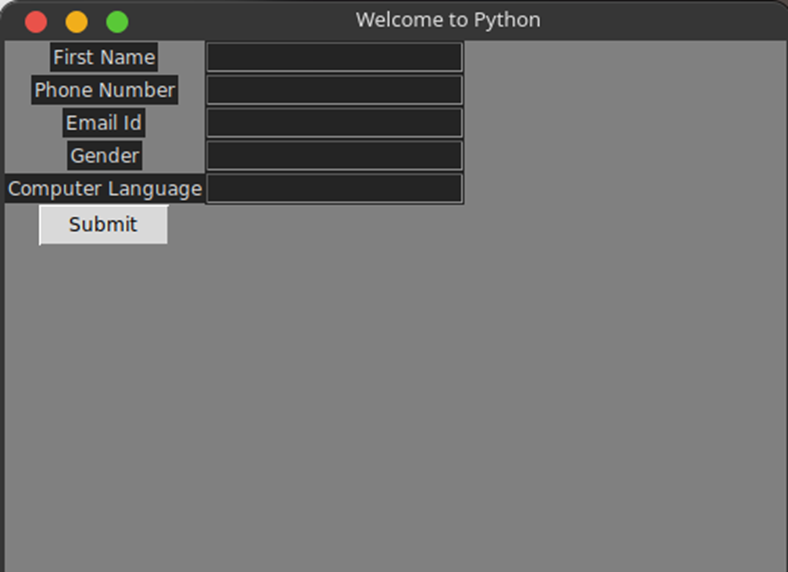
mywin = MyWindow(window)

window.title('Hello Python')

window.geometry("600x500")

window.mainloop()

**Output**



# **5. Data Structure**

**Problem Statement:**

Menu driven program for data structure using built in function for link list, stack and queue.

**Program**

def string():

print ('Assigning string to a variable')

a = "This is a string"

print (a)

def lists():

print ('Declaring a list')

L = [1, "a" , "string" , 1+2]

print (L)

L.append(6)

print (L)

L.pop()

print (L)

def Tupl():

print('Tuples')

tup = (1, "a", "string", 1+2)

print(tup)

print(tup[1])

def Itera():

print ('Iterations in python')

i = 1

while (i < 10):

print(i)

i += 1

def switch\_demo(argument):

if argument==1:

string()

elif argument==2:

lists()

elif argument==3:

Tupl()

elif argument==4:

Itera()

else :

print("Wrong Option")

print("Enter a digit from the following:\n")

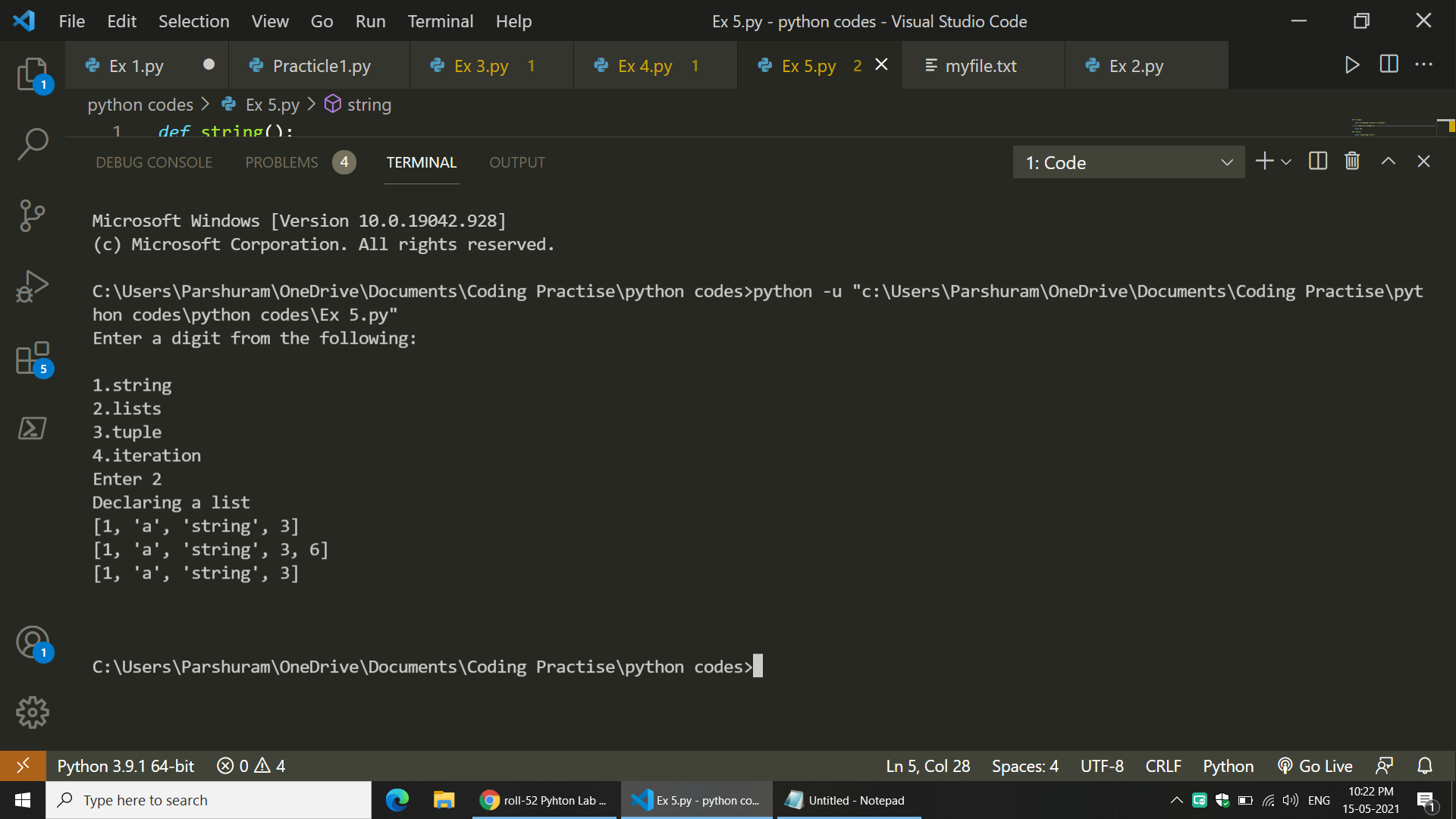
print("1.string\n2.lists\n3.tuple\n4.iteration")

argument=int(input("Enter"))

switch\_demo(argument)

print('\n')

**Output**



# **6. Program to demonstrate CRUD (create, read, update and delete) operations on databases (SQLite/ MySQL) using python.**

**Problem Statement:**

Create GUI in python which will allow Employee to enter his Employee ID, name, email address.

GUI should allow user to update, read, delete and insert new records.

**Program**

# SQL Server Database Connection Properties

DATABASE\_CONFIG ={

'Driver': 'SQL Server',

'Server': 'DESKTOP-QQ95JAD',

'Database': 'Test',

'UID': 'sa',

'Password': 'xxxxxxx'

}

import pypyodbc

import config

# Return the sql connection

def getConnection():

connection = pypyodbc.connect("Driver= {"+config.DATABASE\_CONFIG["Driver"]+"} ;

Server=" + config.DATABASE\_CONFIG["Server"] + ";

Database=" + config.DATABASE\_CONFIG["Database"] + ";

uid=" + config.DATABASE\_CONFIG["UID"] + ";pwd=" + config.DATABASE\_CONFIG["Password"])

return connection

import pypyodbc

import db\_connection as dbConn

from read import Read

from create import Create

from update import Update

from delete import Delete

def main():

print('Available Options: C=Create, R=Read, U=Update, D=Delete ')

choice = input('Choose your option = ')

if choice == 'C':

createObj=Create()

createObj.func\_CreateData()

elif choice == 'R':

readObj = Read()

readObj.func\_ReadData()

elif choice == 'U':

updateObj = Update()

updateObj.func\_UpdateData()

elif choice == 'D':

deleteObj = Delete()

deleteObj.func\_DeleteData()

else:

print('Wrong choice, You are going to exit.')

# Call the main function

main()

import db\_connection as dbConn

class Create:

def func\_CreateData(self):

# Get the sql connection

connection = dbConn.getConnection()

name = input('Enter Name = ')

email = input('Enter email = ')

empid = input(‘Enter EMP id=’)

try:

query = "Insert Into Employee(Name, email,empid) Values(?,?,?)"

cursor = connection.cursor()

# Execute the sql query

cursor.execute(query, [name, email,empid])

# Commit the data

connection.commit()

print('Data Saved Successfully')

except:

print('Something wrong, please check')

finally:

# Close the connection

connection.close()

import db\_connection as dbConn

class Read:

def func\_ReadData(self):

# Get the sql connection

connection = dbConn.getConnection()

cursor = connection.cursor()

# Execute the sql query

cursor.execute('Select \* from Employee')

# Print the data

for row in cursor:

print('row = %r' % (row,))

import db\_connection as dbConn;

class Update:

def func\_UpdateData(self):

# Get the SQL connection

connection = dbConn.getConnection()

id = input('Enter Employee Id = ')

try:

# Fetch the data which needs to be updated

sql = "Select \* From Employee Where Id = ?"

cursor = connection.cursor()

cursor.execute(sql, [id])

item = cursor.fetchone()

print('Data Fetched for Id = ', id)

print('ID\t\t Name\t\t\t Age')

print('-------------------------------------------')

print(' {}\t\t {} \t\t\t{} '.format(item[0], item[1], item[2]))

print('-------------------------------------------')

print('Enter New Data To Update Employee Record ')

name = input('Enter Name = ')

email = input('Enter email = ')

empid = input(‘Enter EMP id=’)

query = "Update Employee Set Name = ?, Age =? Where Id =?"

# Execute the update query

cursor.execute(query, [name, age, id])

connection.commit()

print('Data Updated Successfully')

except:

print('Something wrong, please check')

finally:

# Close the connection

connection.close()

import db\_connection as dbConn

class Delete:

def func\_DeleteData(self):

# Get the SQL connection

connection = dbConn.getConnection()

id = input('Enter Employee Id = ')

try:

# Get record which needs to be deleted

sql = "Select \* From Employee Where Id = ?"

cursor = connection.cursor()

cursor.execute(sql, [id])

item = cursor.fetchone()

print('Data Fetched for Id = ', id)

print('ID\t\t Name\t\t\t Age')

print('-------------------------------------------')

print(' {}\t\t {} \t\t\t{} '.format(item[0], item[1], item[2]))

print('-------------------------------------------')

confirm = input('Are you sure to delete this record (Y/N)?')

# Delete after confirmation

if confirm == 'Y':

deleteQuery = "Delete From Employee Where Id = ?"

cursor.execute(deleteQuery,[id])

connection.commit()

print('Data deleted successfully!')

else:

print('Wrong Entry')

except:

print('Something wrong, please check')

finally:

connection.close()

**Output :**

Avilable Options: C=Create, R=Read, U=Update, D=Delete

Choose your option = C

Enter Name = Sukritee Singh

Enter Email = [Sukritee@gmail.com](mailto:Sanjivani2002@gmail.com)

Enter Empid = 1

Data Saved Successfully

**7. Creation of simple socket for basic information exchange between server and client.**

**Problem Statement:**

Write a socket program in Python where server will receive a number from a client and server will return its factorial to the client.

**Program**

import socket

**Server program:-**

def server\_program():

host = socket.gethostname()

port = 5000

server\_socket = socket.socket()

server\_socket.bind((host, port))

server\_socket.listen(2)

conn, address = server\_socket.accept() # accept new connection

print("Connection from: " + str(address))

while True:

data = conn.recv(1024).decode() # receive data

if not data:

break

print("from connected user: " + str(data))

data = input(' -> ')

conn.send(data.encode())

conn.close()

if \_\_name\_\_ == '\_\_main\_\_':

server\_program()

**Client program:**

import socket

def client\_program():

host = socket.gethostname()

port = 5000

client\_socket = socket.socket()

client\_socket.connect((host, port))

message = input(" -> ")

while message.lower().strip() != 'bye':

client\_socket.send(message.encode()) # send message

data = client\_socket.recv(1024).decode() # receive data

print('Received from server: ' + data)

message = input(" -> ")

client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

client\_program()

**Output**



# **8. Django web framework**

**Problem Statement:**

Create web application using Django web framework to demonstrate functionality of user login

and registration (also validating user detail using regular expression).

**Program**

**Home.html:-**

<p>Welcome to Homepage.</p>

**My home/View.py:-**

from django.views.generic import TemplateView

class HomePageView(TemplateView):

template\_name = 'home.html'

**URLs.py:-**

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('myhome.urls')),

]

**Login.html:-**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>{% block title %}Coding Infinite: Tutorial for Web Login using Django{% endblock %}</title>

</head>

<body>

<main>

{% block content %}

{% endblock %}

</main>

</body>

</html>

{% extends 'base.html' %}

{% block title %}Login{% endblock %}

{% block content %}

<div style="text-align:center"><h2>Login</h2></div>

<div style="text-align:center">

<form method="post" id="form\_login">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Login</button>

</form></div>

{% endblock %}

{% extends 'base.html' %}

{% block title %}Home{% endblock %}

{% block content %}

{% if user.is\_authenticated %}

<div style="text-align:center">Hi {{ user.username }}! Welocme to Coding Infinite!!</div>

<p><a href="{% url 'logout' %}">logout</a></p>

{% else %}

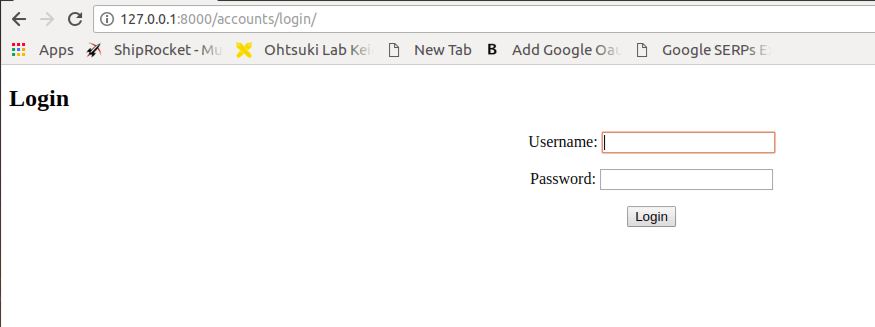
<p>You are not logged in</p>

<a href="{% url 'login' %}">login</a>

{% endif %}

{% endblock %}

**Output**



# **9. Programs on Threading using python.**

**Problem Statement:**

Write Python program to pass arguments to a function and execute it using thread.

**Program**

import threading

def print\_one():

for i in range(10):

print(1)

def print\_two():

for i in range(10):

print(2)

if \_name\_ == "\_main\_":

# create threads

t1 = threading.Thread(target=print\_one)

t2 = threading.Thread(target=print\_two)

# start thread 1

t1.start()

# start thread 2

t2.start()

# wait until thread 1 is completely executed

t1.join()

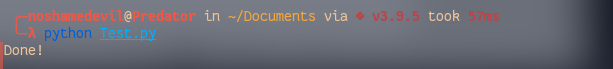
# wait until thread 2 is completely executed

t2.join()

# both threads completely executed

print("Done!")

**Output**



# **10. Exploring basics of NumPy Methods.**

**Problem Statement:**

Write Python program to implement sin, cos, tan, mean, max and min function on Numpy array.

**Program**

import numpy

sin = numpy.sin(numpy.pi / 2)

cos = numpy.cos(0)

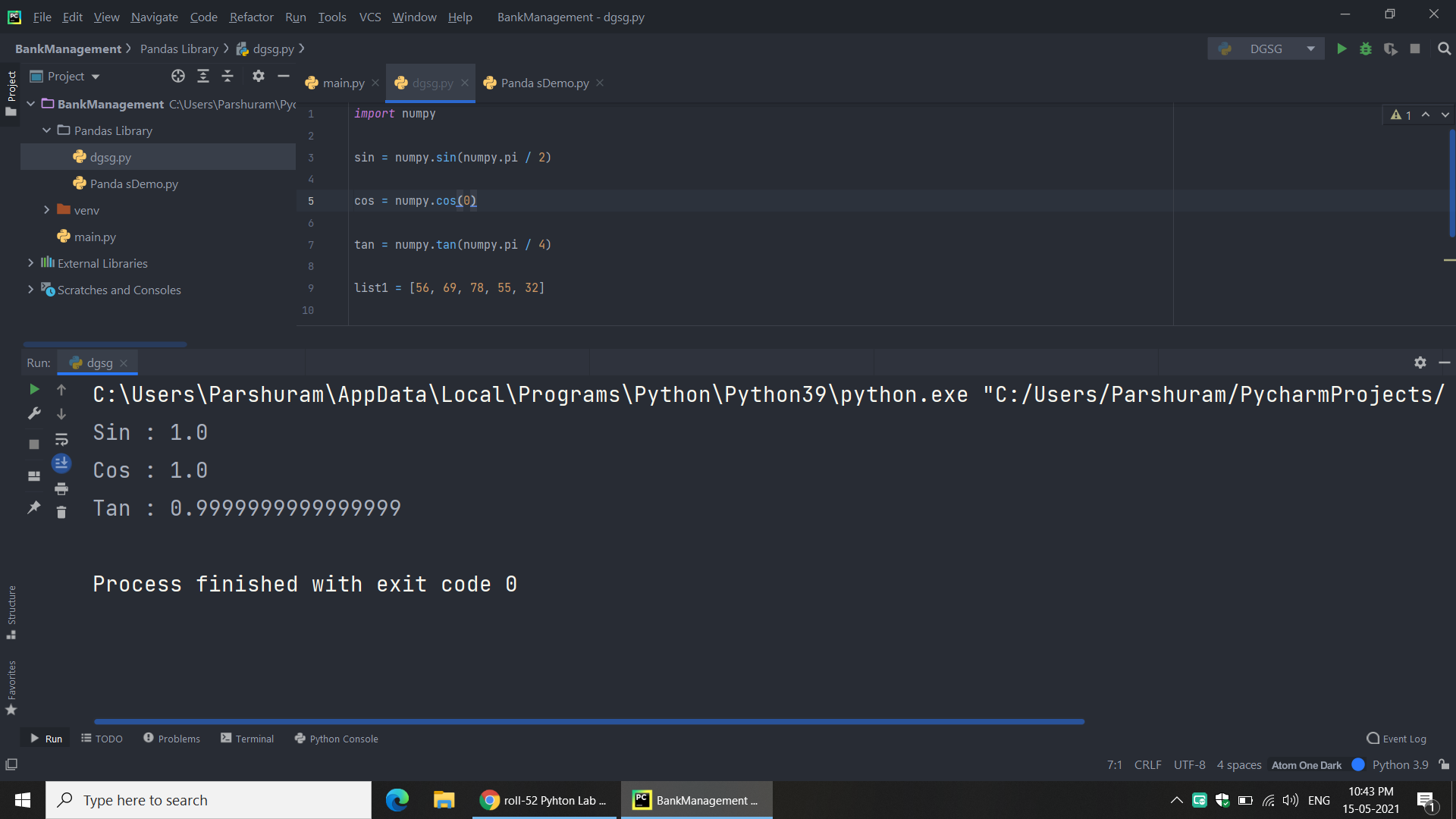
tan = numpy.tan(numpy.pi / 4)

list1 = [56, 69, 78, 55, 32]

list2 = [10, 77, 69, 55, 48]

print("Sin :", sin, "\nCos :", cos, "\nTan :", tan)

**Output**



# **11. Program to demonstrate use of NumPy: Array objects**.

**Problem Statement:**

Write Python program to accept a matrix from the keyboard and display its transpose matrix.

**Program**

import numpy

matrix = [[1, 2, 3],

[4, 5, 6],

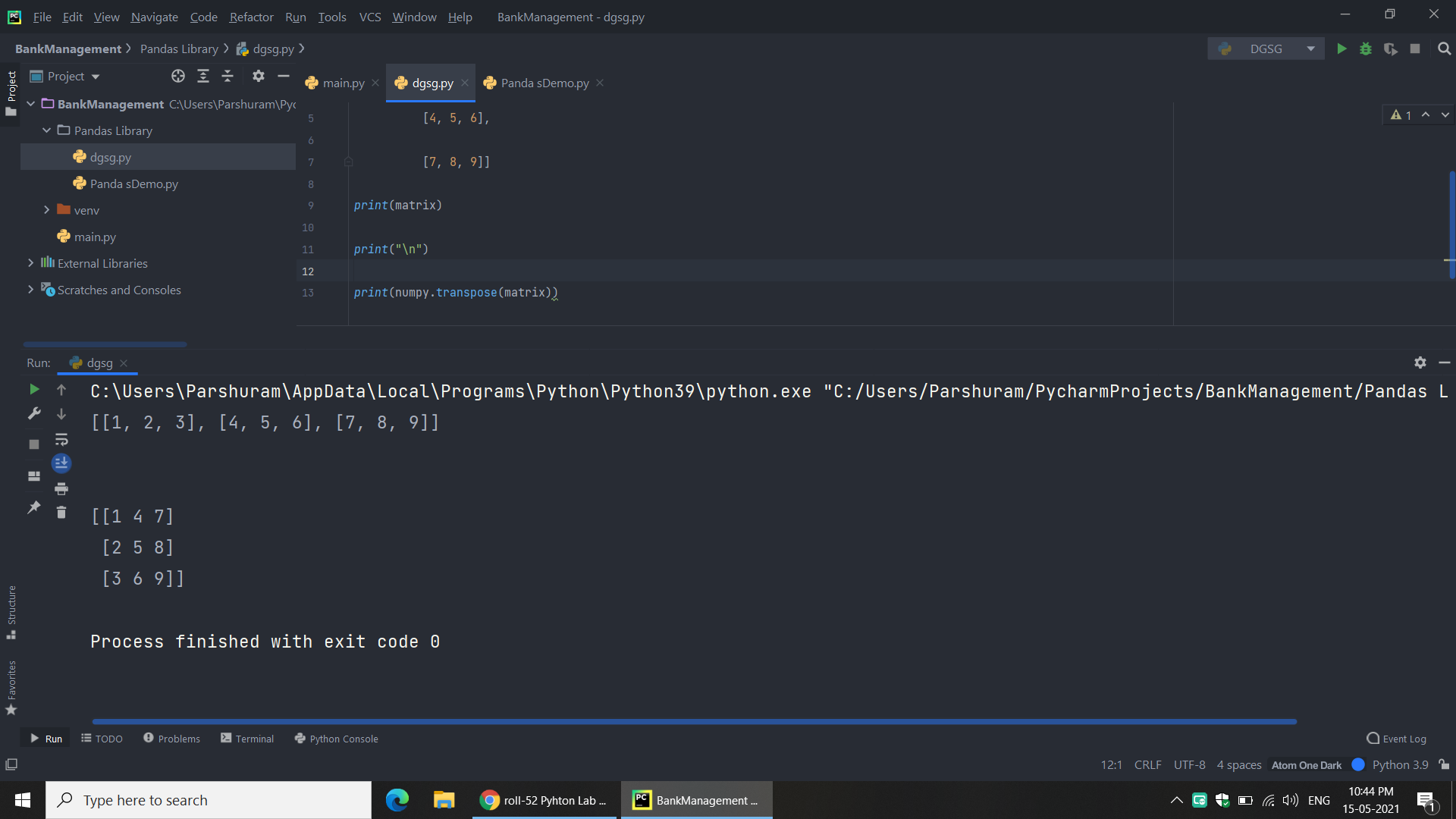
[7, 8, 9]]

print(matrix)

print("\n")

print(numpy.transpose(matrix))

**Output**



# **12. Program to demonstrate Data Series and Data Frames using Pandas.**

**Problem Statement:**

Write Python program to create series and data frame using pandas. Demonstrate conditional

selection, methods to deal with missing data, groupby(), merging and concatenation on data

Frames.

**Program**

print("Concatenation (Outer Join)")

import pandas as pd

import numpy as np

a = pd.DataFrame({'column1': ['A', 'C', 'D', 'E'],

'column2': ['F', 'G', 'H', 'I'],

'column3': ['J', 'K', 'L', 'M'],

'column4': ['N', 'O', 'P', 'Q']},

index=[1,2,3,4])

b = pd.DataFrame({'column3': ['R', 'S', 'T', 'U'],

'column5': ['V', 'W', 'X', 'Y'],

'column6': ['Z', 'a', 'b', 'c'],

'column7': ['n', 'u', 'l', 'k']},

index=[3,4,5,6])

result = pd.concat([a, b], axis=1)

print("Merge")

a = pd.DataFrame({'key': ['A', 'C', 'D', 'E'],

'column2': ['F', 'G', 'H', 'I'],

'column3': ['J', 'K', 'L', 'M'],

'column4': ['N', 'O', 'P', 'Q']},

index=[1,2,3,4])

b = pd.DataFrame({'key': ['C', 'D', 'T', 'U'],

'column5': ['V', 'W', 'X', 'Y'],

'column6': ['Z', 'a', 'b', 'c'],

'column7': ['n', 'u', 'l', 'k']},

index=[3,4,5,6])

result=pd.merge(a, b, on='key')

print("Group by")

a=pd.DataFrame({"key": ["a", "b", "c","a", "b", "c"] ,

"values": [1,2,3,1,2,3]})

b=a.groupby("key").sum()

b

print("Missing Operations")

# dictionary of lists

dict = {'First Score':[100, 90, np.nan, 95],

'Second Score': [30, 45, 56, np.nan],

'Third Score':[np.nan, 40, 80, 98]}

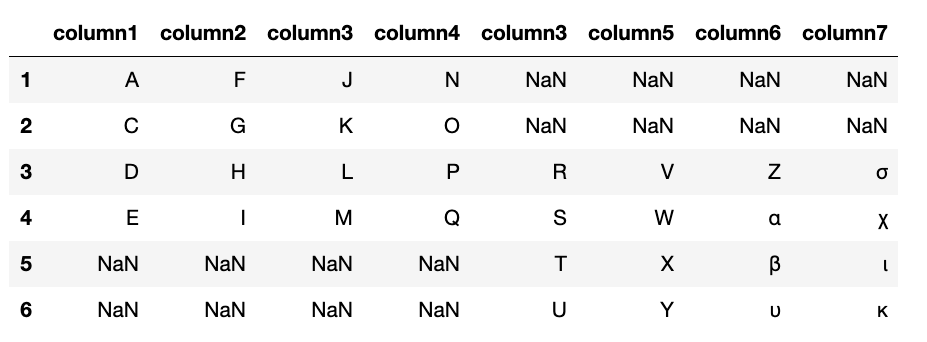
# creating a dataframe from list

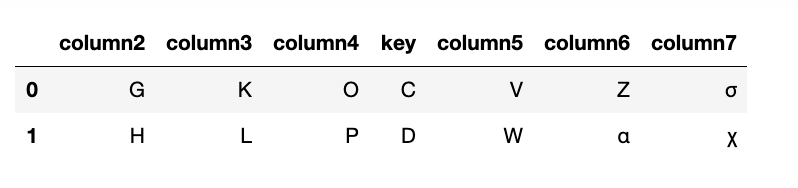
df = pd.DataFrame(dict)

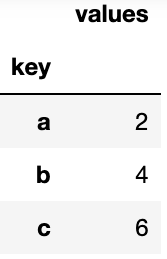
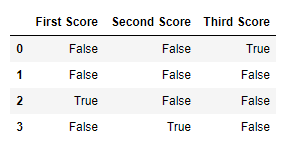
# using isnull() function

df.isnull()

**Output:**





# **13. Program to send email and read content of URL.**

**Problem Statement:**

a. Write Python program to send email to any mail address.

b. Write a Python program to read contents from given URL.

**Program**

1. from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

# For each contact, send the email:

for name, email in zip(names, emails):

msg = MIMEMultipart() # create a message

# add in the actual person name to the message template

message = message\_template.substitute(PERSON\_NAME=name.title())

# setup the parameters of the message

msg['From']=MY\_ADDRESS

msg['To']=email

msg['Subject']="This is TEST"

# add in the message body

msg.attach(MIMEText(message, 'plain'))

# send the message via the server set up earlier.

s.send\_message(msg)

print(“Done”)

del msg

1. from http.client import HTTPConnection

conn = HTTPConnection("example.com")

conn.request("GET", "/")

result = conn.getresponse()

# retrieves the entire contents.

contents = result.read()

print(contents)

**Output:**

**=> Email :** Done

**=> Text from URL**

b'<!doctype html>\n<html>\n<head>\n <title>Example Domain</title

>\n\n <meta charset="utf-8" />\n <meta http-equiv="Content-ty

pe" content="text/html; charset=utf-8" />\n <meta name="viewport

" content="width=device-width, initial-scale=1" />\n <style type

="text/css">\n body {\n background-color: #f0f0f2;\n

margin: 0;\n padding: 0;\n font-family: "Open Sans

", "Helvetica Neue", Helvetica, Arial, sans-serif;\n \n }

\n div {\n width: 600px;\n margin: 5em auto;\n