**VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES**

**VIVEKANANDA SCHOOL OF INFORMATION TECHNOLOGY**



**BACHELOR OF COMPUTER APPLICATION**

**Basics of Python Programming Lab File**

**BCAP 211**

**Guru Gobind Singh Indraprastha University   
Sector - 16C Dwarka, Delhi – 110078**



**SUBMITTED TO:                                               SUBMITTED BY:**

Ms. Priyanka Gupta

Assistant Professor

VSIT

1. Write a program to print the following pattern using a loop.

1

212

32123

4321234

543212345

def pattern(n):

for i in range(1,n+1):

print(" "\*(n-i), end="")

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

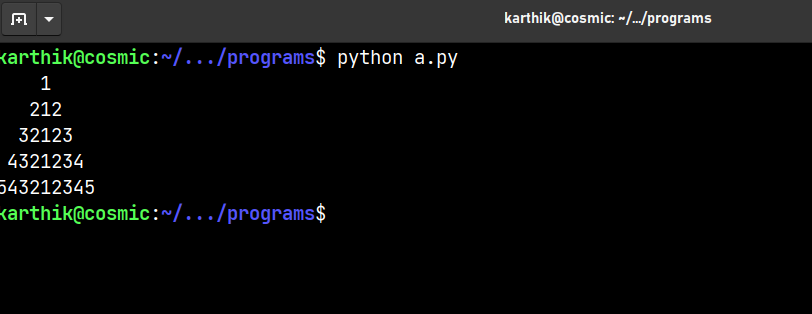
print(j, end="")

for k in range(2, i+1):

print(k, end="")

print()

pattern(5)



2. Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2. The character’s position doesn’t matter.

def balanced(s1, s2):

for i in s1:

if i not in s2:

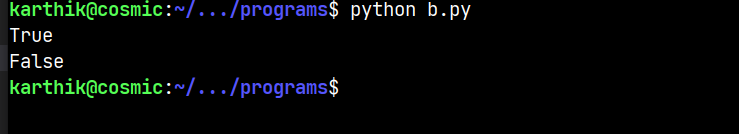
return False

return True

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

print(balanced("abc", "cba"))

print(balanced("abc","asb"))



3. Write a Python program to replace the last element in a list with another list. Sample data : [1, 3, 5, 7, 9, 10], [2, 4, 6, 8] Expected Output: [1, 3, 5, 7, 9, 2, 4, 6, 8]

def replace\_last\_element\_with\_another\_list(l1, l2):

l1[-1:]=l2

return l1

l1 = [1, 3, 5, 7, 9, 10]

l2 = [2, 4, 6, 8]

print(replace\_last\_element\_with\_another\_list(l1, l2))



4. Write a program to sort a tuple of tuples by 2nd item. Sample: ((‘r’,3),(‘t’,1),(‘e’,2),(‘y’,9)) Expected Output: ((‘t’,1),(‘e’,2),(‘r’,3),(‘y’,9))

def sort\_tuple\_of\_tuples\_by\_second\_item(t):

s=sorted([i[1] for i in t])

s\_t=()

for i in s:

for j in t:

if j[1]==i:

s\_t+=(j),

return s\_t

print(sort\_tuple\_of\_tuples\_by\_second\_item((('r',3),('t',1),('e',2),('y',9))))

# Different method

# def sort\_tuple\_of\_tuples\_by\_second\_item(t):

# t=sorted(t, key=lambda x:x[1])

# return t

# print(sort\_tuple\_of\_tuples\_by\_second\_item((('r',3),('t',1),('e',2),('y',9))))



5. Write a Python program to convert a given list of tuples to a list of lists. Original list of tuples: [(1, 2), (2, 3), (3, 4)] Convert the said list of tuples to a list of lists: [[1, 2], [2, 3], [3, 4]]

def list\_of\_tuples\_to\_list\_of\_lists(tuples):

return [list(t) for t in tuples]

print(list\_of\_tuples\_to\_list\_of\_lists([(1, 2), (2, 3), (3, 4)]))



6. Write a Python program to convert a binary number to decimal number using math module.

import math

def binary\_to\_decimal(binary):

n=len(binary)

dec=0

for i in binary:

dec+=int(i)\*math.pow(2,n-1)

n-=1

return dec

print(binary\_to\_decimal("1000101"))



7. Write a program to generate 100 unique random lottery tickets (each lottery number must be 10 digits long) and pick two lucky tickets from it as a winner. Use random module.

import random

def lottery\_generator(n):

tickets=[]

while len(tickets)!=n:

ticket=random.randint(1000000000,9999999999)

if ticket not in tickets:

tickets.append(ticket)

return tickets

def lucky\_tickets(tickets, n):

lucky\_tickets=[]

while len(lucky\_tickets)!=n:

lucky\_ticket=random.choice(tickets)

if lucky\_ticket not in lucky\_tickets:

lucky\_tickets.append(lucky\_ticket)

return lucky\_tickets

tickets=lottery\_generator(100)

print(f"Lucky tickets are {lucky\_tickets(tickets,2)}")



8. Write a function cubesum() that accepts an integer and returns the sum of the cubes of individual digits of that number. Use this function to make functions PrintArmstrong() and isArmstrong() to print Armstrong numbers and to find whether is an Armstrong number.

def cubesum(n):

sum = 0

while n>0:

sum+=(n%10)\*\*3

n=n//10

return sum

def isArmstrong(n):

return n if n==cubesum(n) else False

def PrintArmstrong(n):

if isArmstrong(n):

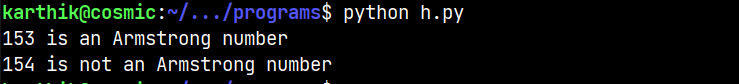
print(f"{n} is an Armstrong number")

else:

print(f"{n} is not an Armstrong number")

PrintArmstrong(153)

PrintArmstrong(154)



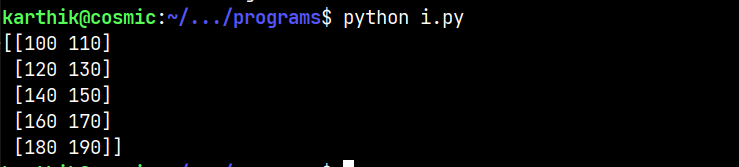
9. Create a 5X2 integer array from a range between 100 to 200 such that the difference between each element is 10. Use NumPy.

import numpy as np

def array\_generator(range, n, r, c):

return np.arange(range[0],range[1],n).reshape(r, c)

print(array\_generator([100,200],10,5,2))



10. Write a Python program to display a bar chart and

pie chart of the popularity of programming Languages using matplotlib.

import matplotlib.pyplot as plt

def bar\_chart(data):

plt.bar(data.keys(),data.values(),color="green")

plt.xlabel("Languages")

plt.ylabel("Popularity")

plt.title("Popularity of programming languages")

plt.show()

def pie\_chart(data):

plt.pie(data.values(),labels=data.keys(),autopct="%1.1f%%")

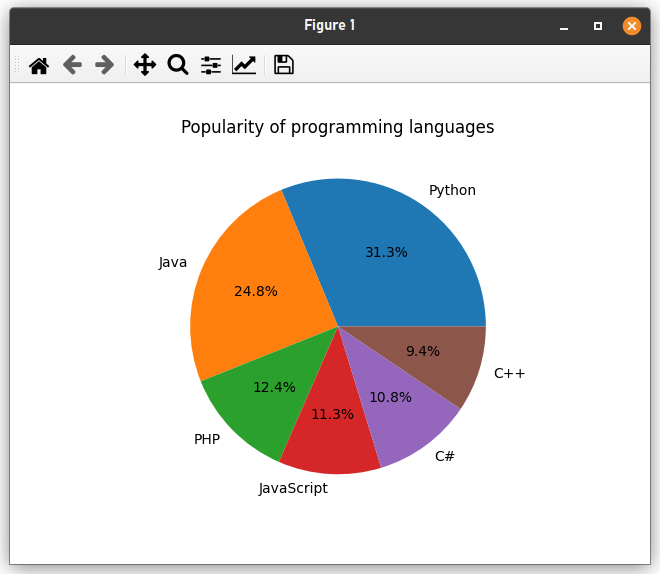
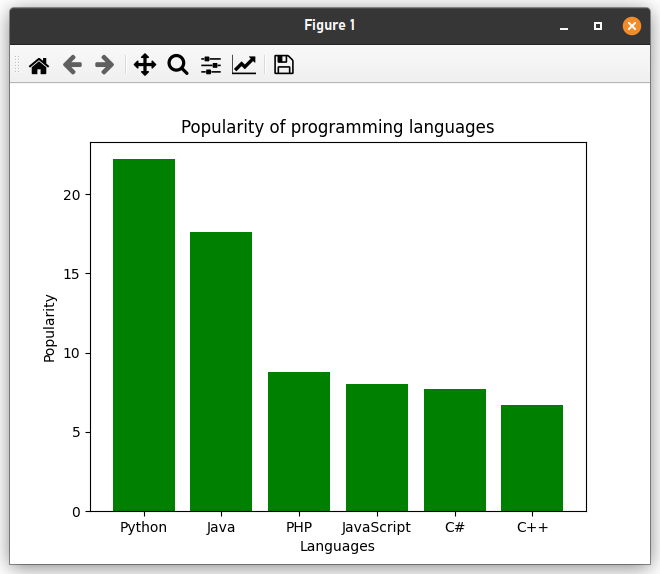
plt.title("Popularity of programming languages")

plt.show()

data={'Python': 22.2, 'Java': 17.6, 'PHP': 8.8, 'JavaScript': 8, 'C#': 7.7, 'C++': 6.7}

bar\_chart(data)

pie\_chart(data)



11. Write a program to count the number of upper-case alphabets present in a text file “PYTHON.TXT”

import os

def countUpperCase(file):

count = 0

for line in file:

for char in line:

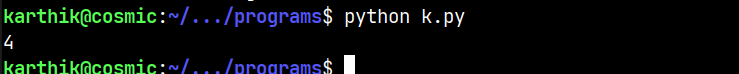
if char.isupper():

count += 1

return count

file = open(os.path.join("files", "PYTHON.TXT"), "r")

print(countUpperCase(file))



12. Create a dictionary whose keys are month names and whose values are the number of days in the corresponding months.

a. Ask the user to enter a month name and use the dictionary to tell them how many days are in the month.

b. Print out all keys in the alphabetical order

c. Print out all the months with 31 days

d. Print out the key value pairs sorted by number of days in each month

months={"January":31,"February":28,"March":31,"April":30,"May":31,"June":30,"July":31,"August":31,"September":30,"October":31,"November":30,"December":31}

# a

month=input("Enter a month name: ")

print(months[month])

# b

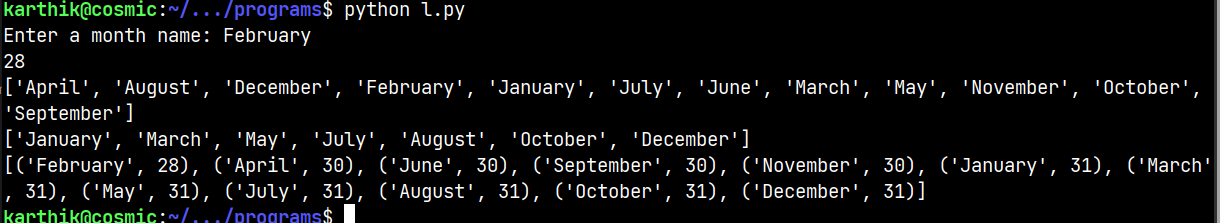
print(sorted(months.keys()))

# c

print([i for i in months if months[i]==31])

# d

print(sorted(months.items(), key=lambda x:x[1]))



13. Write a program to know the cursor position and print the text according to specifications given below:

● Print the initial position

● Move the cursor to 4th position

● Display next 5 characters

● Move the cursor to the next 10 characters

● Print the current cursor position

● Print next 10 characters from the current cursor position

import os

file=open(os.path.join("files","demo.txt"),"rb")

print(f"Initial position: {file.tell()}")

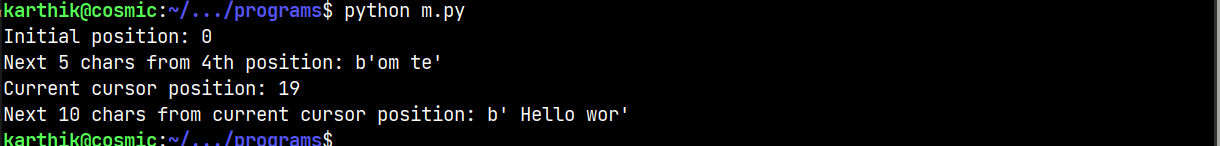
file.seek(4)

print(f"Next 5 chars from 4th position: {file.read(5)}")

file.seek(10,1)

print(f"Current cursor position: {file.tell()}")

print(f"Next 10 chars from current cursor position: {file.read(10)}")



14. Create a binary file with roll number, name and marks. Input a roll number and perform the following operations:

• update the marks.

• Delete the record

• Display the record

• Append the record

• Search the record

import os, pickle

def writerec(rno,name,marks):

f=open(os.path.join("files", "students.data"),'ab')

srec={"rno":rno,"name":name,"marks":marks}

pickle.dump(srec,f)

def readrec():

f=open(os.path.join("files", "students.data"),'rb')

print("Display Student Details")

r=int(input("Enter roll no whose record is to be displayed: "))

print("Roll No",'','Name','\t','Marks',end='')

print()

while True:

try:

rec=pickle.load(f)

if rec['rno']==r:

print('',rec['rno'],'\t',rec['name'],'\t',rec['marks'])

except:

break

def Input():

n=int(input("How many records do you want to enter?"))

for i in range(n):

rno=int(input("Enter roll no: "))

name=input("Enter name: ")

marks=int(input("Enter marks: "))

writerec(rno,name,marks)

def searchrec(rno):

f=open(os.path.join("files", "students.data"),'rb')

while True:

try:

rec=pickle.load(f)

if rec['rno']==rno:

print("Roll no: ",rec['rno'])

print("Name: ",rec['name'])

print("Marks: ",rec['marks'])

break

except EOFError:

print("Record not found \nTry again ")

break

def update():

f=open(os.path.join("files", "students.data"),'rb+')

rno=int(input("Enter roll no whose marks you want to update: "))

try:

while True:

pos=f.tell()

rec=pickle.load(f)

if rec['rno']==rno:

um=int(input("Enter updated marks: "))

rec['marks']=um

f.seek(pos)

pickle.dump(rec,f)

except EOFError:

f.close()

def delete():

f=open('Student.dat','rb')

l=[]

while True:

try:

rec=pickle.load(f)

l.append(rec)

except EOFError:

break

f.close()

rn=int(input("Enter roll no to delete record: "))

f=open('Student.dat','wb')

for i in l:

if i['rno']==rn:

continue

pickle.dump(x,f)

f.close()

while True:

print("1. Update marks \n2. Delete record\n3.Display record\n4.Append record\n5.Search record\n6.Exit")

ch=int(input("Enter choice: "))

if ch==1:

update()

elif ch==2:

delete()

elif ch==3:

readrec()

elif ch==4:

Input()

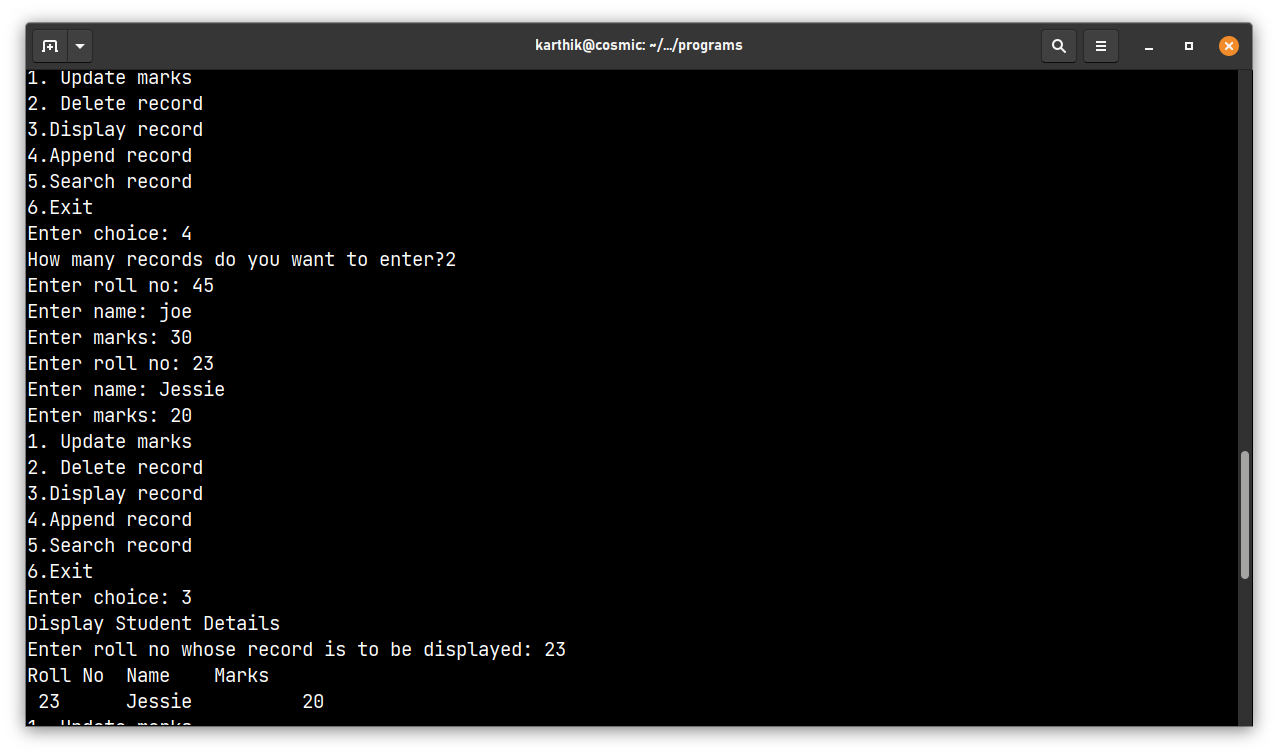
elif ch==5:

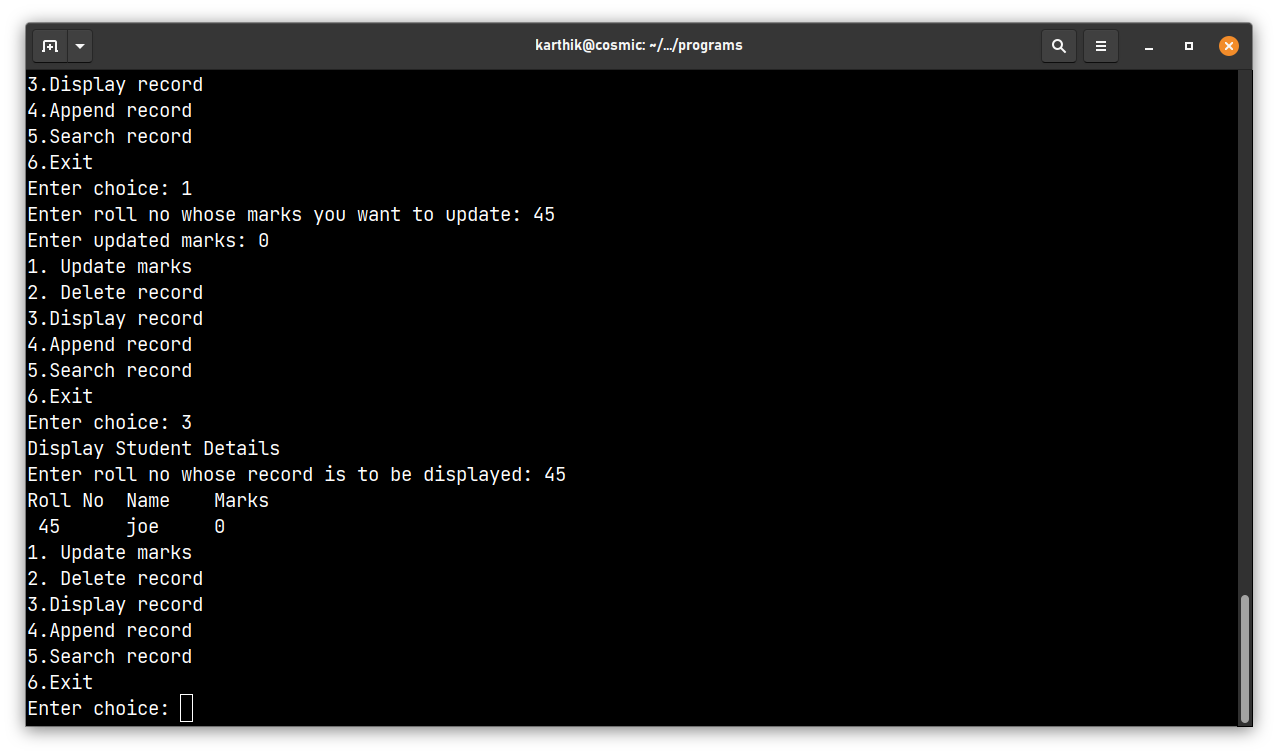
r=int(input("Enter roll no to be searched: "))

searchrec(r)

else:

exit





15. Write a program to Create a CSV file by entering user-id and password, read and search the password for given user id.

import csv

# create a new CSV file with a user-id and password

def create\_csv(user\_id, password):

with open('user\_data.csv', 'w', newline='') as csvfile:

csvwriter = csv.writer(csvfile)

csvwriter.writerow(['user\_id', 'password'])

csvwriter.writerow([user\_id, password])

# read the CSV file and search for a password for a given user-id

def search\_password(user\_id):

with open('user\_data.csv', 'r') as csvfile:

csvreader = csv.reader(csvfile)

# skip the header row

next(csvreader)

for row in csvreader:

if row[0] == user\_id:

return row[1]

return None

# example usage

user\_id = input("Enter user id: ")

password = input("Enter password: ")

create\_csv(user\_id, password)

user\_id = input("Enter user id to search ")

search\_result = search\_password(user\_id)

if search\_result:

print("Password for user id", user\_id, "is", search\_result)

else:

print("User id not found.")

