#Matrix Programs

```
a=[1,2,3]
b=[4,5,6,7]
ncol=2
nrow=4
lst1=[[0]*ncol]*nrow
print(lst1)
```

```
import numpy as np
lst1=[1,2,3,4]
arr1=np.array(lst1)
print(arr1)
```

#Relations Programs

```
a=[1,2,3]
b=[4,5,6,7]
x={(2, 7),(3, 134)}
ab=set()
for i in a:
	for j in b:
		print("(",i,",",j,")",end="")
		ab.add((i,j))
print()
print("SET PRODECT:-",ab)
if x<=ab:
	print("Same")
else:
	print("not same")
```

```
d=domain={"a","b","c","d"}
c=codomain={1,2,3,4,5,6,7,8}
p=pairs={("a",3),("c",8),("d",1),("a",6),("a",5),("a",2)}
for ti in p:
	if not(ti[0] in d):
		print("p is not relation")
	break
```

```
s1=[]
s2=[]
d={"a","b","c","d"}
c={1,2,3,4,5,6,7,8}
p={("a",3),("b",7),("c",1),("d",6)}
for ti in p:
	s1.append(ti[0])
	s2.append(ti[1])
print(s1)
print(s2)
for x in d:
	cal=s1.count(x)
	if cal==0:
		print("That is Function")
	else:
		 print("That is Not a Function")
```

```
d={"a","b","c","d"}
c={1,2,3,4,5,6,7,8}
p={("a",3),("a",7),("a",1),("a",6),("a",5),("a",2)}
ifRel=True
for ti in p:
	if not(ti[0] in d) or not(ti[1] in c):
		ifRel=False
	break
if ifRel:
	print("p in relation")
else:
	print("p in NOT relation")
```

```
#RECURSION
n=int(input("Enter Number:-"))
def rfun(n):
	print("Entering:-",n)
	if n==1:
		print("Exit:-",n)
		fact=1
		return fact

	fact=n*rfun(n-1)
	print("Entering:-",n)
	return fact
print(rfun(n))
```

```
#FIBOONA
n=int(input("Enter Number:-"))
def rfibo(n):
        if n==1:
                return 1
        if n==0:
                return 0
        fibo=rfibo(n-1)+rfibo(n-2)
        return fibo
print(rfibo(n))
```

```
#Function
l1=[]
l2=[]
r1={(1,3),(2,5),(3,7),(1,8)}
for t1 in r1:
        l1.append(t1[0])
        l2.append(t1[1])
print(l1)
print(l2)
isfn=0
for i in l1:
        cnt=l1.count(i)
        if cnt>1:
                isfn=1
        break
if isfn==0:
        print("This is a Function")
else:
        print("This is not a function")
```