Python Scenarios

Variables

1. Create 2 variables with **x** as 100 & **y** as 10 respectively and find the Multiplication and division of both and store in some val as **z** and **z1**.
SOL:
>>> x=100
>>> y=10
>>> z=x*y
>>> z1=x/y
>>> z
1000
>>> z1
10.0

2. Create **a** as 2000 and find the division of **a** by **y** (created in step 1) and reassign **a** with the divided result (200).
SOL:
.>>> a=2000
>>> y=10
>>> a=a/y
>>> a
200.0

3. Prove Python is Dynamically Typed Language: Create x:int=100, then assign the **x** to **y**, but the datatype of **y** has to be of type string. (think about using some function like str()). Print the type of **y** and **x**
SOL:
>>> x:int=100
>>> y=x
>>> type(y)
<class 'int'>
>>> y=str(y)
>>> type(y)
<class 'str'>

4. Prove Python has dynamic inference feature
SOL:
X=abc
X=1

5. Prove Python is Strongly Typed Language
SOL:
A=ten,B=20
A+B not possible
6. Create variables a,b,c,d assigned with 10,20,30,40 respectively
SOL:
>>> [a,b,c,d]=[10,20,30,40]
>>> a

10
>>> b
20
>>> c
30
>>> d
40

7. Prove Python variables are case sensitive
a=10 and A will not work
8. Prove variable name can't start with numbers or cannot contains special character other than _
SOL:
>>> 1=a
File "<stdin>", line 1
SyntaxError: cannot assign to literal
>>> @=naveen
File "<stdin>", line 1
@=naveen
^
SyntaxError: invalid syntax

9. Show some examples of when do we use single, double and triple (single/double) quotes
SOL:
A='naveen'
B=" naveen's prog"
C="""" Naveen is "naveen" """"

10. Show an examples to use arithmetic, comparison, relational and logical operators.
SOL:
Arthimetic x+y
Logical x>y or x<y
Comparison x>y

Conditional Structures

11. Write a program to find the greatest of 3 numbers
SOL:
```
a=input("enter the frst number")
b=input("enter the second number")
c=input("enter the third number")

if a>b and a>c:
   print(f"greatest no is {a}")
elif b>c and b>a:
   print(f"greatest no is {b}")
else:
   print(f"greatest no is {c}")
```

12. Write a single program to find the given number is even or whether it is negative and print the output as (the given number is even but not negative or the given number is not even but negative or

the given number is neither negative nor even)
SOL:

```
a=int(input("enter any no"))
if a%2==0:
    if a>0:
        print("the given number is even but not negative",a)
    elif a<0:
        print("a is even and negative", a)
    else:
        print("given no is zero")
else:
    if a>0:
        print("the given number is not even and not negative",a)
    elif a<0:
        print("the given number is not even and negative", a)
```

13. Write a nested if then else to print the course fees - check if student choosing bigdata, then fees is 25000, if student choosing spark then fees is 15000, if the student choosing datascience then check if machinelearning then 25000 or if deep learning then 45000 otherwise if both then 25000+25000.
SOL:

14. Check whether the given string is palindrome or not (try to use some function like reverse). For eg: x="madam" and y="madam", if x matches with y then print as "palindrome" else "not a palindrome".
SOL:

```
x=input("enter the word")
if x==x[::-1]:
    print(f"given word {x} is palindrome")
else:
    print(f"given word {x} is not palindrome")
```

15. Check whether the x=100 is an integer or string. (try to use some functions like str or upper function etc to execute this use case) or use isinstanceof(variablename,datatype) function.
SOL:

```
x=input("enter a word or no")

if isinstance(x,int):
    print(f"given value {x} is integer")
else:
    print(f"given value {x} is string")
```

Control Statements

16. Write a program using for loop to print even numbers and odd numbers in the below range of data (generate using range function) [5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20] output should be with even as 6,8,10,12,14,16,18,20 and odd as 5,7,9,11,13,15,17,19.
SOL:

```
no=list(range(5,21))
odd=[]
even=[]
```

```
print(no)
for i in no:
    if i%2==0:
        print(f"given no {i} is even")
        even.append(i)
    else:
        print(f"given no {i} is odd")
        odd.append(i)
print(f"list of even no are {even}")
print((f"list of odd no are {odd}"))
```

17. Write a while loop to loop from 0 till 21 with the increment of 3, the result should be exactly 3,6,9,12,15,18 and store this result in a list

SOL:

```
output=[]
i=0
while i<18:
    i += 3
    output.append(i)

print(output)
```

18. Write a for or while loop to print the cube of 4, result should be 4*4*4=64 (initiate some variable outside the loop with 4 and loop through 3 times to achieve the result)

SOL:

```
number = int(input("enter the number for cubic multiply"))
result = number

for _ in range(2):
    result *= number

print(f"the cubic multiply of {number} is {result}")
```

or

```
number = int(input("enter the number for cubic multiply"))
result=1
for i in range(3):
    result *= number

print(f"the cubic multiply of {number} is {result}")
```

or

```
number = int(input("enter the number for cubic multiply"))
result=1
a,b=1,3
while a<=b:
    result *= number
    a+=1
    print(f"the cubic multiply of {number} is {result}")
```

19. Create a list as sal_lst=[10000,20000,30000,10000,15000], loop through the list and add 1000 bonus to the salary and store in another list sal_bonus_lst=[11000,21000,31000,11000,16000] then store the bonus applied salary in another list where sal>11000

SOL:

```
sal=[10000,20000,30000,10000,15000]
bonus=[]
bonusfil=[]
for i in sal:
    bonus.append(i+1000)
print(f"Bonus list is {bonus}")
for i in bonus:
    if i>11000:
        bonusfil.append(i)
print(f"Filtered bonus is {bonusfil}")
```

20. Write a do while loop to print “Inceptez technologies” n number of times as per the input you get from the user. Minimum it has to be printed at least one time regardless of the user input.

SOL:

```
n=int(input("enter the number of times to print inceptez tech"))
control=0
while True:
    print("inceptez technologies")
    control +=1
    if control>=n:
        break
```

21. From the given list of list of elements produce the following output using nested for loop lst1=[[10,20],[30,40,50],[60,70,80]], calculate the sum of all number, calculate the min value and the max value of all the elements in the lst1.

SOL:

```
lst=[[10,20],[30,40,50],[60,70,80]]
minv=lst[0][0]
maxv=lst[0][0]
tot=0
for i in lst:
    for j in i:
        tot=tot+j
        if j < minv:
            minv=j
        elif j > maxv:
            maxv=j
print(tot)
print(minv)
print(maxv)
```

22. Create a looping construct to create 3 tables upto 10 values. Output should be like this…

```
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
.
.
10 x 3 = 30
```

SOL:

```
x=int(input("enter number for table"))
for i in range(1,11):
    print(f"{i}x{x}={i*x}")

###to give range also as a input

x=int(input("enter number for table"))
y=int(input("enter the range for table"))
for i in range(1,y+1):
    print(f"{i}x{x}={i*x}")
```

Collections: List, Dictionary, Tuple and Set

23. Create a list with a range of 10 values starting from 2 to 11 and prove mutability by updating the 3rd element with 100 and prove resizable properties by adding 100 in the 5th position.

SOL:

```
>>> lst=list(range(2,11))
>>> lst.insert(4,100)
>>> lst
[2, 3, 4, 5, 100, 6, 7, 8, 9, 10]
>>> lst[2]=100
>>> lst
[2, 3, 100, 5, 100, 6, 7, 8, 9, 10]
```

24. Create a tuple of 2 fields eg. ("Inceptez","Technologies","Pvt","Ltd"), prove immutability and non resizable nature, access the 2nd and 4th fields and store in another tuple.

SOL:

```
>>> tuple
('Inceptez', 'Technologies', 'Pvt', 'Ltd')
>>> tuple[1]
'Technologies'
>>> tuple[1]='tech'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> tuple.append("new field")
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> tuplenew=(tuple[1],tuple[3])
>>> tuplenew
('Technologies', 'Ltd')
```

25. Convert the list of tuples [("Inceptez","Technologies"),("Apple","Incorporation")] to list of dictionary type, using for loop as given below
[{"Inceptez":"Technologies"},{"Apple":"Incorporation"}] , once the list of dictionary is arrived print only "Incorporation" by passing "Apple" as a key using dict["Apple"] and dict.get("Apple") and try with dict["Apple1"] and dict.get("Apple1") then find the difference between get and using[] notation.

SOL:

```
lt=[("Inceptez","Technologies"),("Apple","Incorporation")]
```

```
ld=[]
cd={}
for i,j in lt:
    ld.append({i:j})
    print(ld)

for k in ld:
    cd.update(k)
    print(cd)
print('Using dict["Apple"]:', cd["Apple"])
print('Using dict.get("Apple"):', cd.get("Apple"))

try:
    print('Using dict["Apple1"]:', cd["Apple1"])
except KeyError as e:
    print('Error', e)

try:
    print("use get apple", cd.get[apple1])
except NameError as f:
    print("name errr",f)
```

26. Create a list of tuple as given below and delete all duplicate tuples of the list
lst=[("Inceptez","Technologies"),("Apple","Incorporation"),("Inceptez","Technologies"),("Inceptez","Technologies")]

SOL:

```
lst=[("Inceptez","Technologies"),("Apple","Incorporation"),("Inceptez","Technologies"),("Inceptez","Technologies")]
lst=list(set(lst))
print(lst)
```

27. Append ("Intel","Corp") in the above de duplicated list

SOL:

```
lst=[("Inceptez","Technologies"),("Apple","Incorporation"),("Inceptez","Technologies"),("Inceptez","Technologies")]
lst=list(set(lst))
print(lst)
lst.append(("Intel","Corp"))
print(lst)
```

28. Convert the lst_dict= [{"Inceptez":"Technologies"},{"Apple":"Incorporation"}] to lst1=["Inceptez","Apple"] , think about using for loop, list() function, keys function and list append functions to achieve this.

SOL:

```
ld= [{"Inceptez":"Technologies"},{"Apple":"Incorporation"}]
#lst1=["Inceptez","Apple"]
lst=[]
for i in ld:
    keys=list(i.keys())
    print(keys)
    lst.append(keys[0])
    print(lst)
```

#key of zero to access the element only otherwise it will come like [[inceptez],[apple]]

29. Create a list of values lst=[10,20,40,30,20], find the first, last values of the list, sort the list in ascending order, sort in descending order, print the minumum and maximum values of the descending sorted list, find the sum of all elements in the list, remove the number 30 and 20 from the list.

SOL:

```
lst=[10,20,40,30,20]
fv=lst[0]
lv=lst[-1]
print(f"first value in list is {fv}")
print(f"last value in list is {lv}")
asc=sorted(lst)
print(asc)
desc=sorted(lst,reverse=True)
print(desc)
mini=min(desc)
maxi=max(desc)
print(f"min and max value is {mini},{maxi}")
tot=sum(lst)
print(f"total list is {tot}")
lst.remove(30)
lst.remove(20)
print(f"after removing 20 and 30 is {lst}")
```

30. Do the above same (step 29) operation in the tuple of elements tup=(10,20,40,30,20)

SOL:

```
tup=(10,20,40,30,20)
fv=tup[0]
lv=tup[-1]
print(f"first value in tuple is {fv}")
print(f"last value in tuple is {lv}")
asc=sorted(tup)
print(asc)
desc=sorted(tup,reverse=True)
print(desc)
mini=min(desc)
maxi=max(desc)
print(f"min and max value is {mini},{maxi}")
tot=sum(tup)
print(f"total tuple is {tot}")
##cannot remove directly from tuple we can do some workaround
tup=list(tup)
print(tup)
tup.remove(30)
tup.remove(20)
tup=tuple(tup)
print(f"after removing 20 and 30 is {tup}")
print(type(tup))
```

31. Convert the string to list from str1="Inceptez Technologies Pvt Ltd" to lst_str1=['Inceptez',

'Technologies', 'Pvt', 'Ltd']
SOL:
```
str1="Inceptez Technologies Pvt Ltd"
ls= str1.split()
print(ls)
```

32. With the below given data in the format of list(list(elements))
emplstlst= [["1", ("Arun","Kumar"), "10000"],["2", ("Bala","Mohan"), "12000"]]

Display the below output for all of the 5 given simple scenarios
a. Convert the first element of the above list into tuple
("1", ("Arun","Kumar"), "10000")
b. Print the second element's second element and reverse the first and last name as given below
("Mohan","Bala")
c. Convert the emplstlst into tuples(tuples)
emplstlst= (("1", ("Arun","Kumar"), "10000"),("2", ("Bala","Mohan"), "12000"))
d. Add all salary of the above list
22000
SOL:
```
lstlst= [["1", ("Arun","Kumar"), "10000"],["2", ("Bala","Mohan"), "12000"]]
a=tuple(lstlst[0])
print(a)
b=lstlst[1][1][1],lstlst[1][1][0]
print(b)
c=tuple(tuple(i) for i in lstlst)
print(c)
d=sum(int(i[2]) for i in lstlst)
print(d)
```

Functions
33. Write def functions to make the above usecases (conditional from 11 to 15 and control statements 26 to 32) and the upcoming usecases more generic.
SOL:
```
11, def greatest(a,b,c):
    if a>=b and a>=c:
        print(f"greatest no is {a}")
        return a
    elif b>=c and b>=a:
        print(f"greatest no is {b}")
        return b
    else:
        print(f"greatest no is {c}")
        return c

a=input("enter the frst number")
b=input("enter the second number")
c=input("enter the third number")
greatest(a,b,c)

12, a=int(input("enter any no"))
def oddeven(a):
    if a%2==0:
```

```
        if a>=0:
            print("the given number is even but not negative",a)
        if a<=0:
            print("a is even and negative", a)
    else:
        if a>=0:
            print("the given number is not even but negative",a)
        if a<=0:
            print("the given number is neither negative nor even", a)
    return a

oddeven()

13, def course():
    a="machinelearning"
    b="deeplearning"
    c="both"
    print("""courses are 'bigdata','spark','datascience' """)
    course=input("enter the course")
    if course=="bigdata":
        print("fee for bigdata is 25k")
        return 25000
    elif course=="spark":
        print("fee for spark is 15k")
    elif course=="datascience":
        print(f"select one of these 1.{a},2.{b}or 3.{c}")
        x=int(input("enter the option"))
        if x==1:
            print("fee for machinelearning is 25k")
        elif x==2:
            print("fee for deeplearning is 45k")
        else:
            print("fee for both is 25k+25k that is 50k")
    else:
        print("course u r selected not available")

d=course()
d=d+5000
print(d)

14, def pali(x):
    if x==x[::-1]:
        print(f"given word {x} is palindrome")
        return "palindrome"
    else:
        print(f"given word {x} is not palindrome")
        return "not palindrome"

print(pali("madam"))

15, def intstr(x):
    if isinstance(x,int):
```

```
        print(f"given value {x} is integer")
        return "integer"
    else:
        print(f"given value {x} is string")
        return "string"

print(intstr("madam"))
```

26,

```
x=[("Inceptez","Technologies"),("Apple","Incorporation"),("Inceptez","Technologies"),("Inceptez",
"Technologies")]

def listup(x):
    x=list(set(x))
    print(x)
    return x

print(listup(x))
```

27,

```
x=[("Inceptez","Technologies"),("Apple","Incorporation"),("Inceptez","Technologies"),("Inceptez",
"Technologies")]

def listup(x):
    x=list(set(x))
    print(x)
    return x

z=listup(x)

y=("Intel","Corp")
def app(y):
    z.append(y)
    return z

print(app(y))
```

28,

```
ld= [{"Inceptez":"Technologies"},{"Apple":"Incorporation"}]
#lst1=["Inceptez","Apple"]
lst=[]

def dic(ld):
    for i in ld:
        keys=list(i.keys())
        print(keys)
        lst.append(keys[0])
        print(lst)
    return lst


print(dic(ld))
```

29,

```
lst=[10,20,40,30,20]
```

```
def fv():
    x=lst[0]
    return x
# to find last value
def lv():
    x=lst[-1]
    return x
#ascending
def asc():
    x=sorted(lst)
    return x
#descending
def desc():
    x=sorted(lst,reverse=True)
    return x
#min value
def mini():
    x=sorted(lst,reverse=True)
    y=min(x)
    return y
#max value
def maxi():
    x=sorted(lst,reverse=True)
    y=max(x)
    return y
#total
def tot():
    x=sum(lst)
    return x
#remove
def rem():
    y=int(input("enter no to remove"))
    lst.remove(y)
    return lst


print(rem())
```

30, tup=(10,20,40,30,20)

```
def fv():
    x=tup[0]
    return x
# to find last value
def lv():
    x=tup[-1]
    return x
#ascending
def asc():
    x=sorted(tup)
    return x
```

```
#descending
def desc():
    x=sorted(tup,reverse=True)
    return x
#min value
def mini():
    x=sorted(tup,reverse=True)
    y=min(x)
    return y
#max value
def maxi():
    x=sorted(tup,reverse=True)
    y=max(x)
    return y
#total
def tot():
    x=sum(tup)
    return x
#remove
def rem():
    global tup
    lst=list(tup)
    y=int(input("enter no to remove"))
    lst.remove(y)
    tup=tuple(lst)
    return tup


print(maxi())

31, str1=input("enter to split")
def spli(str1):
    ls=str1.split()
    return ls
print(spli(str1))

32,       #to convert tuple
lstlst= [["1", ("Arun","Kumar"), "10000"],["2", ("Bala","Mohan"), "12000"]]
def tup(i):
    idx=lstlst[i]
    a=tuple(idx)
    return a

print(tup(0))
#to find list list of values with indrex
lstlst= [["1", ("Arun","Kumar"), "10000"],["2", ("Bala","Mohan"), "12000"]]
def idd(lst,*args):
    value=lst
    for i in args:
        value=value[i]
    return value
```

```
print(idd(lstlst,0,1,1))

#tuple of tuple
lstlst= [["1", ("Arun","Kumar"), "10000"],["2", ("Bala","Mohan"), "12000"]]

def tuptup(lst):
    for i in lst:
        x=tuple(tuple(i))
        return x

print(tuptup(lstlst))
print(idd(lstlst,0,1,1))

#sum
def som(lst):
    d=sum(int(i[2]) for i in lstlst)
    return d
print(som(lstlst))
```

34. Call the above function created for usecase 11 using positional and keyword arguments methodologies

SOL:

```
Print(greatest(2,3,4))
```

35. Write a function to create a calculator that accepts 3 arguments with the datatype of first 2 as int and 3rd one is str, based on the 3rd argument value passed as add/sub/div/mul perform either addition or subraction or multiplication or division respectively of argument 1 and 2 then return the result to the calling environment. Create a default argument function to handle "if the 3rd argument is not passed then default it to add".

SOL:

```
def calc(a1:int,a2:int,a3='add'):
    if a3=="add":
        return a1+a2
    elif a3=="sub":
        return a1-a2
    elif a3=="div":
        return a1/a2
    elif a3=="mul":
        return a1*a2

print(calc(2,3,"mul"))
```

36. Convert the above calculator function to return complex type as tuple by calculating addition, subtraction, multiplication and division in one shot and return result with multiple types for eg. calc(10,3,all) should return (13,7,30,.33)

SOL:

```
def calc(a1:int,a2:int,a3='add'):
    if a3=="add":
        return a1+a2
    elif a3=="sub":
        return a1-a2
```

```
    elif a3=="div":
        return a1/a2
    elif a3=="mul":
        return a1*a2
    elif a3=="all":
        d=(a1+a2,a1-a2,a1/a2,a1*a2)
        return tuple(d)

print(calc(10,3,"all"))
```

37. Create a method to accept a string like “inceptez technologies” and return the following values in multiple result types (capitalize, upper case, length of the string, number of words, ends with “s” or not, replace ‘e’ with ‘a’) for eg. the result should be like this.. (Inceptez Technologies, INCEPTEZ TECHNOLOGIES, 21, 2, True,incaptaz tachnologias)

SOL:

```
def casu(a:str):
    c=a.capitalize()
    u=a.upper()
    l=len(a)
    n=len(a.split())
    e=a.endswith("s")
    r=a.replace("e","a")
    return c,u,l,n,e,r
print(casu("naveen vijays"))
```

38. Create a regular function called promo that accepts 3 arguments where arg1 is amount, arg2 is the offer_percent and arg3 is the offer_cap_limit. Calculate if amount*offer_percent < offer_cap_limit then return amount-(amount*offer_percent) else return the amount-offer_cap_limit

a. Create the function with the above logic and call with all 3 params passed for eg: promo(1000,.10,200)

b. Create the function with the above logic (with default parameter) and call with first 2 params passed for eg: promo(1000,.10)

c. Create the function with the above logic (with arbitrary arguments) and call with all 3 params passed for eg: promo(1000,.10,200)

d. Create the function with the above logic (with keyword arbitrary arguments) and call with all 3 params passed for eg: promo(offer_percent=.10,offer_cap_limit=200,amount=1000)

SOL:

```
def swiggy(a1,a2,a3=100):
    if a1*(a2/100)<a3:
        return round(a1-(a1*(a2/100)))
    else:
        return a1-a3
print(swiggy(2000,10,200)) #a
print(swiggy(2000,10)) #b

def swiggy(*args):
    a1=0
    a2=0
    a3=100
    if len(args)==3:
        a1=args[0]
        a2=args[1]
```

```
        a3=args[2]
        if a1*(a2/100)<a3:
            return round(a1-(a1*(a2/100)))
        else:
            return a1-a3
    elif len(args)==2:
        a1=args[0]
        a2=args[1]
        if a1*(a2/100)<a3:
            return round(a1-(a1*(a2/100)))
        else:
            return a1-a3
    elif len(args)==1:
        a1=args[0]
        if a1*(a2/100)<a3:
            return round(a1-(a1*(a2/100)))
        else:
            return a1-a3

print(swiggy(2000)) #c

print(swiggy(a1=2000,a3=200,a2=10)) #d
```

39. Create a lambda function with the logic of **lam=amount-(amount*offer_percent)** and create a regular function with the above same logic with 4 arguments to Calculate if amount*offer_percent < offer_cap_limit then return **lam(amount,offer_percent)** else return the amount-offer_cap_limit.
Eg. Call this function like promo(1000,.10,200,lam) to ensure this is a higher order function
SOL:

```
lam=lambda amt,off: amt-(amt*(off/100))
def promo(amt,off,lim,lam):
    if amt*(off/100) <= lim:
        return lam(amt,off)
    elif amt*(off/100)>lim:
        offf=200
        return amt-offf

print(promo(1000,10,200,lam))
```

40. Create a lambda function as like the regular function created in step 38.
SOL:

```
x=lambda amt,off,lim : amt-(amt*(off/100)) if amt*(off/100) <= lim else amt-lim
print(x(3000,10,200))
```

Exception Handling
41. Apply exception handler code in the above usecase number 35 to achieve the followings
    a. If the calculator function is called with either the first or second argument as non integer values then raise Exception and call the calculator function with the type casted value
for eg. calc("10",20, "add") in the except block of the exception handler we have to call the same function as calc(int("10"),20, "add") and return the result to the calling environment.
SOL:

```
def calc(a1,a2,a3):
    try:
```

```
        if isinstance(a1,int) and isinstance(a2,int):
            if a3=="add":
                return a1+a2
            elif a3=="sub":
                return a1-a2
            elif a3=="div":
                return a1/a2
            elif a3=="mul":
                return a1*a2
        else:
            raise TypeError
    except TypeError as err:
        a1=int(a1)
        a2=int(a2)
        return(calc(a1,a2,a3))

print(calc(20,'20','add'))
```

42. Write an exception handler code to raise exception if the 2nd argument passed is a negative value in the function created in step 38

SOL:

```
def promo(amt,off,lim):
    try:
        if off<0:
            raise TypeError
        else:
            if amt*(off/100) <= lim:
                return amt-(amt*off/100)
            elif amt*(off/100)>lim:
                offf=200
                return amt-offf
    except TypeError as err:
        return("value is negative")
print(promo(1000,-10,200))
```

OOPS

43. Create a package namely inceptez.usecases
44. Inside the above package, Create a module called oops.
45. Create 2 classes, one class called mask and one more class called endecode
46. Inside the class mask create a private value as addhash=100 and a method hashMask(str) should return the hash of str+addhash value. For eg. If I pass hashMask("abc"+str(100)) it should return 6867843172862474341
47. Inside the class endecode create a private value as prefixstr=”aix” and a method revEncode(str) ={return the prefixstr+reverse of str value}
48. Create another module inside the same package and create an object namely objmask to instantiate the class mask and objendecode to instantiate the class endecode
49. Create a list with 3 names like [“arun”,”ram kumar”,”yoga murthy”], loop the elements using map function and apply hashMask for all 3 elements and print the masked values.
50. Loop the list created in the above step and apply the revEncode function for all the 3 elements and print of the encoded values.

51. Create a decode function inside endecodeclass as revDecode and write a logic to decode the encoded string we got in step 50.

SOL:

MODULE OOPS

```
class Mask():
    def __init__(self):
        self.addhash=100

    def hashmask(self,arg):
        hashvalue=0
        prime=31
        comb=arg+str(self.addhash)
        for i in comb:
            hashvalue=hashvalue*prime+ord(i)
        return hashvalue



class Endecode():
    def __init__(self):
        self.prefix="aix"

    def revcode(self,args):
        rev=args[::-1]
        combi=self.prefix+rev
        return combi

    def revDecode(self, enc):
        if enc.startswith(self.prefix):
            reversed_s = enc[len(self.prefix):]
            return reversed_s[::-1]
        else:
            raise ValueError("Encoded string does not start with the expected prefix.")
```

MODULE TO CALL

```
from oops import *

objmask = Mask()
objrev = Endecode()
lst = ["arun", "naveen", "vijay"]

# Step 1: Calling mask method
hashed_values = list(map(objmask.hashmask, lst))
print(hashed_values)
print("Hashed values:")
for name, hash_value in zip(lst, hashed_values):
    print(f"{name}: {hash_value}")

# Step 2: Apply revEncode using map
```

```
encoded_values = list(map(objrev.revcode, lst))
print(encoded_values)
print("\nEncoded values:")
for name, encoded_value in zip(lst, encoded_values):
    print(f"{name}: {encoded_value}")
#zip will make it look like (arun,22394349)
# Step 3: Apply revDecode
print("\nDecoded values:")
for encoded_value in encoded_values:
    decoded_value = objrev.revDecode(encoded_value)
    print(f"{encoded_value}: {decoded_value}")

REWRITED:
from oops import *
obj=Mask()
obj1=Endecode()
#step 1:
lst=["naveen","hari","vijay"]

masked=list(map(obj.hashmask, lst))
print(masked)
print("\nMasked list:")
for i,j in zip(lst,masked):
    print(f"{i} : {j}")

encode=list(map(obj1.revcode,lst))
print(encode)
print("\nEncoded list:")
for i,j in zip(lst,encode):
    print(f"{i}:{j}")
print("\nDecodes values:")
for i in encode:
    decode=obj1.revDecode(i)
    print(f"{i}:{decode}")
```