In [1]:
```
#Xor Operation
a=2
b=3
print(a^b)
print(a^a)
```

```
1
0
```

In [2]:
```
l=[1,2,3]
print('List repetition:',l*2)
s='abc'
print('String repetition:',s*2)
```

```
List repetition: [1, 2, 3, 1, 2, 3]
String repetition: abcabc
```

In [3]:
```
# Packages
def add(a,b):
  return a+b
a,b=map(int,input().split())
print(add(a,b))
```

```
1 4
5
```

In [4]:
```
#default constructor
class const:
  def __init__(self):
    pass
print('hi')
```

```
hi
```

In [5]:
```
class Stu:
  student_name='No name'
  def __init__(self,name):
    self.sname=name
    print('Class variable:',self.student_name)
a1=input()
s1=Stu(a1)
print('Passed variable:',s1.sname)
```

```
jack
Class variable: No name
Passed variable: jack
```

In [6]:
```
#student class
#variables - name,roll number,branch,marks,attendence
#methods - view_attendence(),view_marks(),view_name(),update_name()
class Student:
  #parameterized constructor
  def __init__(self,name,marks,attendence):
    self.name=name
    self.marks=marks
    self.attendence=attendence
  def view_name(self):
    print('Student name is:',name)
  def update_name(self,new_name):
    self.name=new_name
    print('Student new name:',self.name)
  def view_marks(self):
    print('Student marks:',self.marks)
  def attendance(self):
    print('Student attendance:',self.attendence)
name=input('Enter name:')
marks=int(input('Enter marks:'))
attendence=int(input('Enter attendence:'))
obj=Student(name,marks,attendence)
obj.view_name()
```

```
Enter name:jack
Enter marks:96
Enter attendence:90
Student name is: jack
```

In [7]:

```
#count no of pairs in the given list whose sum is equal to given sum
l=list(map(int,input().split()))
k=int(input())
d=[]
for i in range(len(l)):
  a=k-l[i]
  l1=l[i+1:]
  if a in l1 and [l[i],a] not in d:
    d.append([l[i],a])
print(d)
print(len(d))
```

```
1 2 3 4 2 2
4
[[1, 3], [2, 2]]
2
```

In [8]:

```
#bank details
class bank:
  def __init__(self,username,account_no,branch):
    self.username=username
    self.account_no=account_no
    self.branch=branch
  def print_details(self):
    print('Username:',self.username)
    print('Account_no:',self.account_no)
    print('Branch:',self.branch)
name=input()
account_no=int(input())
branch=input()
obj=bank(name,account_no,branch)
obj.print_details()
```

```
john
123
john
Username: john
Account_no: 123
Branch: john
```

In [9]:

```
# Encapsulation
#printing bank details
class Bank:
  #private object
  __account_IFSC=''
  __password=''
  def __init__(self,account_no,user_name,branch):
    self.account_no=account_no
    self.user_name=user_name
    self.branch=branch
    self.__password='random password'
    self.generate_IFSC()
  def generate_IFSC(self):
    self.account_IFSC=f"IFSC{self.branch}{self.account_no}"
    return self.account_IFSC
  def print_details_of_user(self):
    print('Account number:',self.account_no)
    print('User name:',self.user_name)
    print('Bank name:',self.branch)
username=input()
account_no=int(input())
branch=input()
a=Bank(account_no,username,branch)
print(a.generate_IFSC())
a.print_details_of_user()
```

```
user
1223
pdp
IFSCpdp1223
Account number: 1223
User name: user
Bank name: pdp
```

In [10]:

```
class User:
  full_name=''
  email=''
  __password=''
  mobile_number=''
  def __init__(self,full_name,email,password,mobile_number):
    self.full_name=full_name
    self.email=email
    self.password=password
    self.mobile_number=mobile_number
  def update_name(self,new_name):
    self.full_name=new_name
  def get_name(self):
    return self.full_name
  #setter method
  def update_password(self,new_password):
    self.__password=new_password
  def update_mobile_number(self,new_mobile_number):
    self.mobile_number=new_mobile_number
  #getter method
  def get_user_password(self):
    return self.__password
```

In [11]:

```
#login
class Login:
  __db=[]
  def __init__(self):
    self.print_menu()
  def print_menu(self):
    print('Welcome User')
    print('1.Register')
    print('2.Login')
    print('3.Exit')
  def create_user(self,name,email,password,mobile_number):
    new_user=User(name,email,password,mobile_number)
    self.__db.append(new_user)
    return True
  def validate_user(self,email,password):
    temp=self.__db.copy()
    for user in temp:
      if user.email==email:
        if user.get_user_password()==password:
          return 'Successfully logged in'
        else:
          return 'Password is incorrect'
    return 'User not found'
obj=Login()
while True:
  option=input('Enter your choice:')
  if option=='1':
    name=input('Enter your full name:')
    email=input('Enter your email:')
    password=input('Enter your password:')
    mobile_number=int(input('Enter mobile number:'))
    res=obj.create_user(name,email,password,mobile_number)
    if res==True:
      print('User successfully registered!!')
  elif option=='2':
    email=input('Enter your email:')
    password=input('Enter your password:')
    validation=obj.validate_user(email,password)
    print(validation)
  elif option=='3':
    break
  else:
    print('Invalid input')
```

```
Welcome User
1.Register
2.Login
3.Exit
Enter your choice:1
Enter your full name:abc
Enter your email:abc@gmail.com
Enter your password:12345
Enter mobile number:123456789
User successfully registered!!
Enter your choice:3
```

In [12]:

```
#validating password
#starting with alphabet
```

```
#end with @gmail.com
#abcdef@gmail.com
password=input()
s='abcdefghijklmnopqrstuvwxyz'
d=password[len(password)-10:]
if (password[0] in s or password[0].lower()) and d=='@gmail.com':
  print('Your email format is correct!!')
else:
  print('**OOPS you entered wrong format!!**')
```

```
abc@gmail.com
Your email format is correct!!
```

In [ ]: