Python Question and Answers

Python Questions with Solutions

Question I

Create a function with the name check_palindrome which takes a list of strings as input.

The function checks each string of the list whether it is a palindrome or not and returns another list containing the palindrome from the input string.

Note:

i. A palindrome is a a word, phrase, or sequence that reads the same backwards as forwards, e.g. madam

ii. The check for the string to be palindrome should be case-insensitive , e.g. madam and Madam both should be considered as palindrome.

You can use the below sample input and output to verify your solution.

Sample Input:

```
3
Hello
Malayalam
Radar
```

Output:

```
Malayalam
Radar
```

The first line in the sample input is the count of strings to be included in the list as the 1st input. The strings are then provided one by one as the next lines of input.

For more details, please refer to the below main section of code

You can use the below sample main section to test your code.

Sample Main Method:

```
if __name__=='__main__':
   count=int(input())
```

```
inp_str=[]
for i in range(count):
    inp_str.append(input())
output=check_palindrome(inp_str)
 if len(output)!=0:
    for i in output:
        print(i)
else:
    print('No palindrome found')
```

Solution 1

```
def check_palindrome(str_list):
    res=[]
    for str in str_list:
      if str[::-1].lower()==str.lower():
         res.append(str)
    return res

if __name__=='__main__':
    count=int(input())
    inp_str=[]
    for i in range(count):
        inp_str.append(input())
    output=check_palindrome(inp_str)
    if len(output)!=0:
      for i in output:
        print(i)
    else:
        print('No palindrome found')
```

Question II

Create a class Employee with the below attributes:

emp_id of type Number

emp_name of type String

emp_role of type String

emp_salary of type Number

Create the __init__ method which takes all parameters in the above sequence. The method should set the value of attributes to parameter values inside the method.

Create a method inside the class with the name increase_salary . This method takes a Number value as arguments which is the percentage by which the salary should be incremented and increases the salary of the employee by the given percentage amount. If the employee salary is 10000 and the given percentage is 10, then the updated salary of the employee should be 11000.

Create another class Organisation with the below attributes:

Create another class Organisation with the below attributes:

org_name of type String

emp_list of type List having Employee objects

Create the __init__ method which takes all parameters in the above sequence. The method should set the value of attributes to parameter values inside the method.

Create another method inside the class with the name calculate_increment. This method takes a String value and a Number value as argument which is the employee role and the increment percentage respectively. The method checks all the employees in the employee list of the organisation whose role is same as the given role and increase their salary by the given increment percentage. The method returns a list of employees containing all the employees whose salary was incremented.

Note: use the method increase_salary defined in the Employee class to calculate the incremented salary of the employees whose role is same as the given role.

You can use the below given sample input and output to verify your solution. The 1st input taken in the main section is the number of Employee objects to be added to the list of Employees.

The next set of inputs are the employee id, employee name, employee role and employee salary for each employee taken one by one.

Sample Input:

```
4
100
Rajesh
Developer
40000
101
Ayesha
Developer
41000
102
Hari
Analyst
45000
103
Aman
Manager
60000
Developer
5
```

Output:

```
Rajesh 42000.0
Ayesha 43050.0
```

For more clarity, please refer to the below main section of code.

You can use this section to test your code

Sample main section:

```
if __name__=='__main__':
```

```
emp_list=[]
count=int(input())
for i in range(count):
    eid=int(input())
    name=input()
    role=input()
   salary=int(input())
   emp_list.append(Employee(eid,name,role,salary))
o=Organisation("XYZ Corp",emp_list)
inp_role=input()
inp_percent=int(input())
result=o.calculate_increment(inp_role,inp_percent)
if len(result)!=0:
  for emp in result:
     print(emp.emp_name,'\t',emp.emp_salary)
else:
  print('No employee found with the given role')
```

Solution II

```
class Employee:
    def __init__(self,eid,ename,role,salary):
        self.emp_id=eid
        self.emp_name=ename
        self.emp_role=role
        self.emp_salary=salary

    def increase_salary(self,percent):
        self.emp_salary+=self.emp_salary*percent*0.01

class Organisation:
      def __init__(self,name,elist):
        self.org_name=name
        self.emp_list=elist
    def calculate_increment(self,role,percent):
        emp_res=[]
        for emp in self.emp_list:
            if emp.emp_role==role:
                emp.increase_salary(percent)
                emp_res.append(emp)
        return emp_res

if __name__=='__main__':

    emp_list=[]

    count=int(input())

    for i in range(count):

        eid=int(input())

        name=input()

        role=input()

        salary=int(input())

        emp_list.append(Employee(eid,name,role,salary))
    o=Organisation("XYZ Corp",emp_list)

    inp_role=input()

    inp_percent=int(input())

    result=o.calculate_increment(inp_role,inp_percent)

    if len(result)!=0:

        for emp in result:

            print(emp.emp_name,'\t',emp.emp_salary)

    else:

        print('No employee found with the given role')
```

Question III

Create a class Account with the below attributes:

account_no of type Number
account_name of type String
account_balance of type Number

Create the **__init__** method which takes all parameters in the above sequence and sets the value of the attributes inside the method.
Create a method **depositAmnt** inside the Account class which an takes a number value as input parameter. The number value represents the amount to be deposited into the Account.The method updates the balance of the Account i.e. adds the given amount to the existing balance of the Account object.
Create another method withdrawAmnt inside the Account class which takes which an takes a number value as input parameter. The number value represents the amount to be withdrawn from the Account.The method deducts the amount from the existing balance of teh Account object. However, the minimum balance of the Account obejct is to be maintained as 1000. i.e withdrawal of the given amount will be possible only if the balance of the Account object is greter than or equal to 1000 after the amount is withdrawn. The method return 1 if the withdrawal is possible; else returns 0 if the withdrawal is not possible.

To test the code against your customized input through console, the input data needs to be entered in the below order(as shown below in the sample input).

The first three lines in the below sample input represents the input for three variables of account object i.e account no.(account_no),account name (account_name) and account balance (account_balance), with which the Account object will be created.
The fourth line in the sample input is the input for the amount to be deposited in the account object and fifth line is the input for the amount to be withdrawn from the account object
Sample input:

```
120
Rajesh
1500
1200
2000
```

Output:

```
2700

Insufficient balance for withdrawal
```

For more details, please refer the below main section of the code.

Note: The below main section is already added in your console, You are requested not to modify anything in that section or it might result in failure of the test cases configured.

Sample main section:

```
if __name__ == '__main__':
  acno=int(input())
  acname=raw_input()
  acntbal=int(input())
  depamnt=int(input())
  withamnt=int(input())
  acnt=Account(acno,acname,acntbal)
  acntdemoobj=AccountDemo()
  print(acntdemoobj.depositAmnt(acnt,depamnt))
  print(acntdemoobj.withdrawAmnt(acnt,withamnt))
```

Solution III

```
Class Account:
    def__init__(self,acno,acntname,accntbal):
        self.account_no=acno
        self.account_name=acntname
        self.account_balance=accntbal

   def depositAmnt(self,amount):
       self.account_balance+=amount

  def withdrawAmnt(self,amount):
      if(self.account_balance-amount<1000):
         return 0
      else:
         self.account_balance-=amount
      return 1


if __name__ == '__main__':
    acno=int(input())
    acname=input()
    acntbal=int(input())
   depamnt=int(input())
   withamnt=int(input())
   obj=Account(acno,acname,acntbal)
   obj.depositAmnt(depamnt)
```

```
print("Balance after deposit :",obj.account_balance)
res=obj.withdrawAmnt(withamnt)
if res == 1:
  print("Balance after withdrawal :",obj.account_balance)
else:
  print("Insufficient balance for withdrawal")
```

Question IV

Create a class Employee with the below attributes :

empno of numeric type

empname of string type

leaves of dictionary type

The leaves dictionary will have the leave type(EL ,SL,CL) as key and no of balance leaves (for the respective leave type) as values of the key

Create the __init__ method which takes all parameters in the above sequence and set them as the values of the attributes inside the method.

Create another class Company with below attributes:

cname of string type

emps of list type having employee objects

Create the __init__ method which takes all parameters in the above sequence and set them as the values of the attributes inside the method.

Create a method display_leave in the Company class, which takes the employee number and a leave type as input parameters and display the corresponding leave balance of the given leave type for the employee with the given employee number. Create another method leave_application in the Company class, which takes employee number, leave type and number of leaves applied by the employee in the given leave type. In this method,for the given employee, check whether the leave balance in the given leave type is more than or equal to the number of leaves applied, to grant the leave. Return "Granted" if all the conditions are matched. Otherwise return "Rejected".

To test the code against your customized Input through console:

The input data needs to be entered in the below order(as shown below in the sample input).

Example:

a. Lets assume that you want to add 3 Employee objects. Input the data is as shown in the sample input.

b. The first line (3) in the below sample input indicates the number of employee objects to be created.

Next five lines from second to 6th line , i.e.

```
1
Rajesh
5
10
```

20

in the below sample input represents employee number,employee name,leave balance for “CL”, leave balance for “EL” and leave balance for “SL”of one employee object.

c. Similarly seventh line to twelfth line i.e.

2

Venkat

5

20

30

from the below sample input represents employee number ,employee name,leave balance for “CL”, leave balance for “EL” and leave balance for “SL”of second employee object .

Likewise, next five lines represents the input data for third employee object.

d. The last 3 lines in the below sample input represents the employee number, leave type and number of leaves requested.<

Note:

The employee number and the leave type value provided will act as a input for display_leave method and as well as for leave_application method . The number of leaves provided will act as input for leave_application method.

Sample Input:

```
3
1
Rajesh
5
10
3
1
Rajesh
5
10
20
2
Venkat
5
20
30
3
```

```
Brahma
10
10
10
3
EL
10
```

Sample output:

```
10
Granted
```

here 10 is the number of leaves available with the employee of type EL and "Granted" is the status of the leave_application.

For more details, please refer to the below main section of the code

Note:

The below main section is already added in your console, You are requested not to modify anything in that section or it might result in failure of the test cases configured.

```python
if __name__=='__main__':
    n=int(input())
    c=Company("ABC",emps=[])
    for i in range(n):
        leaves={}
        eno=int(input())
        ename=input()
        leaves["CL"]=int(input())
        leaves["EL"]=int(input())
        leaves["SL"]=int(input())
        e=Employee(eno,ename,leaves)
        c.emps.append(e)
    empno=int(input())
    ltype=input()
    nol=int(input())
    print(c.display_leave(empno,ltype))
    print(c.leave_application(empno,ltype,nol))
```

Solution IV

```
class  Employee:

 def __init__(self,eno,ename,leaves):
   self.empno=eno
   self.empname=ename
   self.leaves=leaves

class Company:
  def __init__(self,cname,emps):
     self.cname=cname
     self.emps=emps

  def leave_application(self,eno,ltype,nol):
     chk=False
    for e in self.emps:
       if e.empno==eno:
       for lt,nl in e.leaves.items():
           if lt==ltype and nl>=nol:
              chk=True
   if chk:
      return "Granted"
  else:
      return "Rejected"

defdisplay_leave(self,eno,ltype):
fore inself.emps:
ife.empno==eno:
returne.leaves[ltype]


if__name__=='__main__':
 n=int(input())
 c=Company("ABC",emps=[])
foriin range(n):
leaves={}
eno=int(input())
ename=input()
leaves["CL"]=int(input())
leaves["EL"]=int(input())
leaves["SL"]=int(input())
 e=Employee(eno,ename,leaves)
c.emps.append(e)
empno=int(input())
```

```
ltype=input()
nol=int(input())
print(c.display_leave(empno,ltype))
print(c.leave_application(empno,ltype,nol))
```

Question V:

Create a function with the name count_words that take a string as input and return the count of occurrence of each word in the string. The word and count of occurrence of the word should be returned from the function as a dictionary. Consider the words to be separated by space in the string. Create another function with the name max_occurence_word that will take a string as input and return the word with maximum occurrence in the string.

Note : Use the function count_words inside this function to get the count of occurrence of each word. To test the code against your customized Input through console, please refer to the below instructions The input string needs to be entered in the console.

Consider the following sample input and expected output for reference.

Sample Input : Hello Hi Bye TCS Hello Welcome Hello Bye

Expected Output : Hello

Please use the below main program code to implement and to test/run your code and submit the complete code along with this main code.

```
if __name__ == '__main__':
    input_string=raw_input()
   dict=count_words(input_string)
   st=max_occurence_word(input_string)
   print(st)
```

Solution V

```
def count_words(str):
  words=str.split()
  res={}
  for i in words:
    res[i]=words.count(i)
  return res
```

```
def max_occurence_word(str):
  d=count_words(str)
  v = list(d.values())
  k = list(d.keys())
  return k[v.index(max(v))]

if __name__ == '__main__':
  input_string=input()
  dict=count_words(input_string)
  print(dict)
  st=max_occurence_word(input_string)
  print(st)
```

Question IV:

Create a class Book with following attributes:

book_id of type Number

book_name of type String

Create the __init__ method to take the above attributes as input in the given sequence.

Create another class Library with following attributes:

library_id of type Number

address of type String

book_list of tye List

Create a method in the Library class with the name count_books which takes a character as input and returns the count of books in the list of books that belongs to the library whose name starts with the given character.

Create another method in the same class with the name remove_books which takes a list of book names as input and removes the book from the book list of the Library if the name of the book matches with the name given in the input list of book names.

To test the code against your customized Input through console:

The input data needs to be entered in the below order(as in the below sample input).

Example:

a. The first input is the count of book objects to be added.

b. The second and the third lines in the sample input represents book id and book name of the a book object respectively. The next two lines are the book id and the book name of the next book object.. and so on for the other values of the book objects.

c. The next line represents the character that is to be searched.

d. The next line is the count of book names to be provided as the input list of book names.

e. The subsequent lines are the names of the books provided to be included in the list

Sample Input, inline to the above description :

```
3

100

C++ Programming

200

Introduction to SQL

300

Core Java

C

3

Odyssey

Core Java

Hello World
```

Output for the above sample input:

```
2 C++ Programming Introduction to SQL
```

Note: For more details on a. Input data entered from standard input b. How this data will be processed c. The order of the input data " , please refer the main program Please use the below main program code to implement and to test/run your code and submit the complete code along with this main code.

```
if __name__ == '__main__':

 books=[]

 count=int(input())

 for i in range(count):

    bid=int(input())

    bname=raw_input()

    b=Book(bid,bname)

    books.append(b)

  l=Library(123,'Mumbai',books)
 char=raw_input()
 ccount=int(input())
 names=[]
 for i in range(ccount):
   names.append(raw_input())
 print(l.count_books(char))
l.remove_books(names)
for i in l.book_list:
  print(i.book_name)
```

Solution VI

```
class Book:
  def __init__(self,bid,bname):
    self.book_id=bid
    self.book_name=bname


class Library:
  def __init__(self,id,address,blist):
    self.library_id=id
    self.address=address
    self.book_list=blist


  def count_books(self,char):
    count=0
    for i in self.book_list:
      if i.book_name.startswith(char)==True:
        count+=1
    return count


  def remove_books(self,blist):
        for book in blist:
            for b in self.book_list:
                if b.book_name==book:
                    self.book_list.remove(b)

if __name__ == '__main__':
    books=[]
    count=int(input())
    for i in range(count):
        bid=int(input())
        bname=input()
        b=Book(bid,bname)
        books.append(b)
    l=Library(123,'Mumbai',books)
    char=input()
    ccount=int(input())
    names=[]
    for i in range(ccount):
        names.append(input())
    print(l.count_books(char))
    l.remove_books(names)
    for i in l.book_list:
        print(i.book_name)
```

Question VII:

Create a function with the name check_prime which takes a number as input and checks if the given number is prime or composite.

The function returns 1 if the number is prime,0 if the number is composite and 0 otherwise.

Create another function with the name prime_composite_list which takes a list of numbers and checks whether the numbers in the list are prime or composite.

Include all the prime numbers in one list and all the composite numbers in another list.

Create a 3rd list and include the list of prime numbers and the list of composite numbers in the 3rd list and return the 3rd list from this function.

Note : use the function check_prime to find whether the number is prime or composite.

To test the code against your customized Input through console, please refer to the below instructions:

The first line in the sample input is the count of numbers to be provided in the input list
The next few lines are the numbers to be included in the list provided one by one

Sample Input :

```
4

11

7

90

44
```

Expected Output :

```
[[11, 7], [90, 44]]
```

Please use the below main program code to implement and to test/run your code and submit the complete code along with this main code.

```
if __name__=='__main__':

  inp=[]

  count=int(input())

  for i in range(count):

     inp.append(int(input()))

  print(check_prime(inp[1]))
  result=prime_composite_list(inp)
  print(result)
```

Solution VII:

```
def check_prime(num):
  flag=-1
  if num > 1:
    for i in range(2, num//2):
      if (num % i) == 0:
        flag=0
        break
      else:
        flag=1
  return flag

def prime_composite_list(li):
  prime=[]
  composite=[]
  for i in li:
    if check_prime(i)==1:
      prime.append(i)
    elif check_prime(i)==0:
      composite.append(i)
  res=[]
  res.append(prime)
  res.append(composite)
  return res

if __name__=='__main__':
  inp=[]
  count=int(input())
```

```
for i in range(count):
   inp.append(int(input()))
print(check_prime(inp[1]))
result=prime_composite_list(inp)
print(result)
```

Question VIII

Create a class Item with following attributes:

item_id of type Number
item_name of type String
item_price of type Number
quantity_available of type Number

Create the __init__ method to take the above attributes as input in the given sequence

create a method calculate_price which will take a number as input which represnts the qunatty of the item to be checked. the method checks if the item has available quantity greater than or equal to the given quantity. If it is available, calculate the total price of the item by considering the item price and the quantity(Total price=Item price * Quantity). Return the total price from the method if the qunatoty is available or else return 0.

Create another class Store with following attributes:

item_list of type List

Create the __init__ method to take the above attributes as input in the given sequence
Create a method in the Store class with the name generate_bill which takes a dictionary as input. The dictionary includes the item name as the key and the quantity required as the value. This method calculates the total price of each item whose item_name is provided as key of the dictionary and returns the bill amount.
Note : use the calculate_price method of the Item class to calculate total price of each item

To test the code against your customized Input through console:
The input data needs to be entered in the below order(as in the below sample input).

Example:

a. The first input is the count of Item objects to be added.
b. The second,third,fourth and fifth line is the item id, item name, item price and the quantity available fot the 1st Item object. The subsequent lines have the same for the other item objects.
c. The next line is the count of values to be included in the input dictionary
d. The next line is name of the 1st required item and the subsequent line is the rquired quantity and so on for the other dictionary values.
For more details, refer to the sample input an its output with reference to the given main section below

Sample Input:
```
3
1
Bread
30
5
2
Milk
50
10
3
Cookies
100
2
3
Bread
2
Butter
1
Milk
2
```

Output for the above sample input:
```
60
160
```

Note:
For more details on
a. Input data entered from standard input
b. How this data will be processed
c. The order of the input data "" ,
please refer the main program

Please use the below main program code to implement and to test/run your code and submit the complete code along with this main code.

```
if __name__ == '__main__':
  ilist=[]
  icount=int(input())
  for i in range(icount):
    id=int(input())
    name=input()
    price=int(input())
    quant=int(input())
    i=Item(id,name,price,quant)
    ilist.append(i)
  s=Store(ilist)
  inp={}
  inp_count=int(input())
  for i in range(inp_count):
    name=input()
    quantity=int(input())
    inp[name]=quantity
  print(ilist[0].calculate_price(2))
  print(s.generate_bill(inp))
```

Solution VIII:

```
class Item:
  def __init__(self,id,name,price,quant):
    self.item_id=id
    self.item_name=name
    self.item_price=price
    self.quantity_available=quant

  def calculate_price(self,quantity):
    if self.quantity_available >= quantity:
      self.quantity_available-=quantity
      return self.item_price*quantity
    else:
      return 0
```

```
class Store:
  def __init__(self,ilist):
    self.item_list=ilist

  def generate_bill(self,req):
    bill=0
    for i in req:
      for j in self.item_list:
        if i == j.item_name:
          bill+=j.calculate_price(req[i])
    return bill
if __name__ == '__main__':
  ilist=[]
  icount=int(input())
  for i in range(icount):
    id=int(input())
    name=input()
    price=int(input())
    quant=int(input())
    i=Item(id,name,price,quant)
    ilist.append(i)
  s=Store(ilist)
  inp={}
  inp_count=int(input())
  for i in range(inp_count):
    name=input()
    quantity=int(input())
    inp[name]=quantity
  print(ilist[0].calculate_price(2))
  print(s.generate_bill(inp))
```