S.No: 1	Exp. Name: ***Write the code to print the documents (syntax, description etc.) of Python built-in function(s).***	Date: 2023-02-23

ID: 21SCSE1011675

Aim:

Write a Python program to print the documents (syntax, description etc.) of Python built-in function(s).

Sample functions: abs(),complex(),len(),round(),sorted()

Source Code:

documents.py

```
print(abs.__doc__)
print(complex.__doc__)

print(len.__doc__)
print(round.__doc__)
print(sorted.__doc__)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Return the absolute value of the argument.
complex(real[, imag]) -> complex number
Create a complex number from a real part and an optional imaginary part.
This is equivalent to (real + imag*1j) where imag defaults to 0.
Return the number of items in a container.
round(number[, ndigits]) -> number
Round a number to a given precision in decimal digits (default 0 digits).
This returns an int when called with one argument, otherwise the
same type as the number. ndigits may be negative.
Return a new list containing all items from the iterable in ascending order.
A custom key function can be supplied to customize the sort order, and the
reverse flag can be set to request the result in descending order.

S.No: 2	Exp. Name: ***Program to find factorial of a number***	Date: 2023-02-23

Aim:

Write a program to find the factorial of a given number

Source Code:

Fac.py

```
# def fac(n):
#       if(n==0 | n==1):
#               return 1
#       return n*fac(n-1)

n=int(input("Enter a number: "))

res=1
for i in range(2,n+1):
        res*=i
print("Factorial of {0} is {1}".format(n,res))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a number:
4
Factorial of 4 is 24

Test Case - 2
User Output
Enter a number:
15
Factorial of 15 is 1307674368000

Test Case - 3
User Output
Enter a number:
1
Factorial of 1 is 1

S.No: 3	Exp. Name: ***Write the code which accepts the radius of a circle from the user and compute the area***	Date: 2023-02-23

Aim:

Write a Python program which accepts the radius of a circle from the user and compute the area

Note: Print the result up to 4 decimal places.

Source Code:

area_of_circ.py

```
from math import pi
radius=float(input("radius: "))
print("area:%.4f" % (pi*radius*radius))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
radius:
10
area:314.1593

Test Case - 2
User Output
radius:
12.45
area:486.9547

S.No: 4	Exp. Name: ***Write a Python program to print the calendar of a given month and year.***	Date: 2023-02-23

Aim:

Write a Python program to print the calendar of a given month and year.

Source Code:

year_month.py

```
import calendar
year=int(input("Input the year : "))
month=int(input("Input the month : "))
print(calendar.month(year,month))
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Input the year :
2012
Input the month :
4
     April 2012
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
```

Test Case - 2

User Output

```
Input the year :
1998
Input the month :
6
      June 1998
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
```

S.No: 5	Exp. Name: ***Write a Python Program for simple interest, and compound interest***	Date: 2023-02-24

Aim:

Write a Python Program for simple interest, and compound interest

Note: Print the result up to 3 decimal places.

Source Code:

S_C_interest.py

```
p=float(input("amount: "))
t=float(input("time: "))
r=float(input("rate: "))

print("Simple interest: {0:.3f}".format((p*r*t)/100))
amt=p*((1+r/100)**t)
print("Compound interest: {0:.3f}".format(amt-p))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
amount:
10000
time:
2
rate:
3
Simple interest: 600.000
Compound interest: 609.000

Test Case - 2
User Output
amount:
3450
time:
3
rate:
4
Simple interest: 414.000
Compound interest: 430.781

S.No: 6	Exp. Name: ***Write a Python program to calculate the length of a string.***	Date: 2023-02-23

Aim:

Write a Python program to calculate the length of a string.

Source Code:

length.py

```
str=str(input())
print("Length: {0}".format(len(str)))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
python
Length: 6

Test Case - 2
User Output
Hello World!!
Length: 13

S.No: 7	Exp. Name: ***Write a Python program to split and join a string***	Date: 2023-02-24

Aim:

Write a Python program to split and join a string

Source Code:

splitandjoin.py

```
l=str(input())
# l.split()
print(l.split(" "))
# l.join()
print("-".join(l.split()))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
sun rises in the east
['sun', 'rises', 'in', 'the', 'east']
sun-rises-in-the-east

Test Case - 2
User Output
welcome to programming!!
['welcome', 'to', 'programming!!']
welcome-to-programming!!

S.No: 8	Exp. Name: ***Write the code demonstrate various ways of accessing the string- By using Indexing (Both Positive and Negative)***	Date: 2023-02-24

Aim:

Write a Python program to demonstrate various ways of accessing the string- By using Indexing (Both Positive and Negative)

Source Code:

accessing.py

```
s=str(input())
pi=int(input("positive index: "))
try:
        print(s[pi])
except Exception as e:
        print("Index out of range")

ni=int(input("negative index: "))
try:
        print(s[ni])
except Exception as e:
        print("index out of range")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
python
positive index:
6
Index out of range
negative index:
-6
p

Test Case - 2
User Output
positive
positive index:
9
Index out of range
negative index:
-10
index out of range

Test Case - 3
User Output
bankalert
positive index:
5
1
negative index:
-6
k

S.No: 9	Exp. Name: ***Write a Python program to multiplies all the items in a list.***	Date: 2023-02-24

Aim:

Write a Python program to multiplies all the items in a list.

Source Code:

product.py

```
a=list(map(int,input().split(",")))

res=1
for i in a:
        res*=i
print(res)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1,2,3,4
24

Test Case - 2
User Output
10,20,30,40,50
12000000

S.No: 10	Exp. Name: ***Write a program to reverse a given List***	Date: 2023-02-24

Aim:

Create a List with the user-given elements. Write a program to reverse the given List, and print the result to the console as shown in the example.

Sample Input and Output:

```
data: Python,Java,Perl,Swift,R
reverse: ['R', 'Swift', 'Perl', 'Java', 'Python']
```

Source Code:

List12.py

```
a=list(map(str,input("data: ").split(",")))
a.reverse()
print("reverse:",a)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
data:
Python,Java,Perl,Swift,R
reverse: ['R', 'Swift', 'Perl', 'Java', 'Python']

Test Case - 2
User Output
data:
Django,Flask,Hibernate,Spring
reverse: ['Spring', 'Hibernate', 'Flask', 'Django']

S.No: 11	Exp. Name: ***Write a Python program to find smallest number in a list, and to find largest number in a list***	Date: 2023-02-24

Aim:

Write a Python program to find smallest number in a list, and to find largest number in a list

Source Code:

product.py

```
l=list(map(int,input().split(",")))
print("smallest number: {0}".format(min(l)))
print("largest number: {0}".format(max(l)))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1,2,3,4,5,6,7,8
smallest number: 1
largest number: 8

Test Case - 2
User Output
2,2,2,2,2,2,2,2
smallest number: 2
largest number: 2

Aim:

Write Python program to perform following operations on Dictionaries:
a) Update Dictionary
b) Looping through Dictionary
c) Delete Set

Note: Take keys as strings and values as integers.

Source Code:

dictionary.py

```
key=list(map(str,input("keys: ").split(",")))
value=list(map(int,input("values: ").split(",")))

d=dict(zip(key,value))
print(d)

l=list((input("key,value to be updated: ").split(",")))
l[1]=int(l[1])

it=iter(l)
d2=dict(zip(it,it))
d.update(d2)

print(d)

print("looping through dictionary")
for i in d:
        print(i,d[i])

print("After deletion")
d.clear()
print(d)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
keys:
a,b,c,d
values:
1,2,3,4
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
key,value to be updated:
e,5

{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
looping through dictionary
a 1
b 2
c 3
d 4
e 5
After deletion
{}

Test Case - 2
User Output
keys:
app,ap,a
values:
23,32,2
{'app': 23, 'ap': 32, 'a': 2}
key,value to be updated:
apa,3
{'app': 23, 'ap': 32, 'a': 2, 'apa': 3}
looping through dictionary
app 23
ap 32
a 2
apa 3
After deletion
{}

S.No: 13	Exp. Name: ***Write a Python script to sort (ascending and descending) a dictionary by value.***	Date: 2023-02-24

Aim:

Write a Python script to sort (ascending and descending) a dictionary by value.

Source Code:

sorted.py

```
from operator import itemgetter
key=list(map(str,input("keys: ").split(",")))
value=list(map(int,input("values: ").split(",")))


d=dict(zip(key,value))
print(d)

sort_dic=dict(sorted(d.items(),key=itemgetter(1)))
print("Ascending order:  {0}".format(sort_dic))
sort_dic=dict(sorted(d.items(),key=itemgetter(1),reverse=True))
print("Descending order:  {0}".format(sort_dic))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
keys:
a,b,c,d,e
values:
2,3,1,5,4
{'a': 2, 'b': 3, 'c': 1, 'd': 5, 'e': 4}
Ascending order: {'c': 1, 'a': 2, 'b': 3, 'e': 4, 'd': 5}
Descending order: {'d': 5, 'e': 4, 'b': 3, 'a': 2, 'c': 1}

Test Case - 2
User Output
keys:
we,re,ty,re,sd,yt,gt
values:
23,54,32,12,12,10,9
{'we': 23, 're': 12, 'ty': 32, 'sd': 12, 'yt': 10, 'gt': 9}
Ascending order: {'gt': 9, 'yt': 10, 're': 12, 'sd': 12, 'we': 23, 'ty': 32}
Descending order: {'ty': 32, 'we': 23, 're': 12, 'sd': 12, 'yt': 10, 'gt': 9}

S.No: 14	Exp. Name: ***Write a python code to Create a dictionary and apply the following methods create 1) Print the dictionary items 2) access items 3) use get() 4)change values 5) use len()***	**Date: 2023-02-24**

Aim:

Write a python code to Create a dictionary and apply the following methods create 1) Print the dictionary items 2) access items 3) use get() 4)change values 5) use len().

Source Code:

dicop.py

```
n=int(input("Enter No of Elements : "))
d={}
for i in range(0,n):

        key=str(input("Enter Key : "))
        value=str(input("Enter Value : "))
        d[key]=value

l=list(zip(d.keys(),d.values()))
print("Dictionary Items :  dict_items({0})".format(l))

acc=str(input("Enter Key to Access item : "))
print("Item for key  {0}  is  {1}".format(acc,d[acc]))

get=str(input("Enter Key to Get Element : "))
print("Item for key  {0}  is  {1}".format(get,d.get(get)))

chg=str(input("Enter Key to change that element : "))
valc=str(input("Enter Value to change : "))
d[chg]=valc
print("After Changing value Updated dictionary :  {0}".format(d))
print("Length :  {0}".format(len(d)))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter No of Elements :
3
Enter Key :
1
Enter Value :
codetantra
Enter Key :
2
Enter Value :

mallareddy
Enter Key :
3
Enter Value :
abc
Dictionary Items : dict_items([('1', 'codetantra'), ('2', 'mallareddy'), ('3', 'abc')])
Enter Key to Access item :
1
Item for key 1 is codetantra
Enter Key to Get Element :
2
Item for key 2 is mallareddy
Enter Key to change that element :
2
Enter Value to change :
mallareddy university
After Changing value Updated dictionary : {'1': 'codetantra', '2': 'mallareddy university', '3': 'abc'}
Length : 3

S.No: 15	Exp. Name: ***Write a python program to Create a tuple and perform the following methods create 1) Add items 2) len() 3) check for the item in tuple 4)Access items.***	**Date: 2023-02-24**

Aim:

Write a python program to Create a tuple and perform the following methodscreate 1) Add items 2) len() 3) check for the item in tuple 4)Access items.

Source Code:

tupleop.py

```
t=tuple(map(str,input("Enter Input Separated By , (Comma) : ").split(",")))
print("Tuple :  {0}".format(t))

t2=tuple(map(str,input("Enter Input Separated By , (Comma) : ").split(",")))
res=t+t2
print("After Adding items tuple :  {0}".format(res))
print("Length : ",len(res))

strr=str(input("Enter item to Search : "))
if strr in res:
        print("{0}  is Found".format(strr))
else:
        print("{0}  is not Found".format(strr))


ind=int(input("Enter index to access item : "))
try:
        print("Item at index {0}  is  {1}".format(ind,res[ind]))
except Exception as e:
        # print(e)
        pass
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Input Separated By , (Comma) :
1,2,3
Tuple : ('1', '2', '3')
Enter Input Separated By , (Comma) :
4,5,6
After Adding items tuple : ('1', '2', '3', '4', '5', '6')
Length : 6
Enter item to Search :
4
4 is Found

```
Enter index to access item :
3
Item at index 3  is  4
```

S.No: 16	Exp. Name: ***Write a python program to define a module to find Fibonacci Numbers and import the module to another program.***	**Date: 2023-02-24**

Aim:

Write a python program to define a module to find Fibonacci Numbers and import the module to another program.

Source Code:

fibonacci.py

```
import math

def fib(phi,n):
        for i in range(0,n+1):
                res=round(pow(phi,i)/math.sqrt(5))
                print(res,end=" ")

n=int(input("Enter the max value: "))
if n==1:
        print("Fibonacci series upto 1 :")
        print("0")
else:

        phi=(1+math.sqrt(5))/2
        if(n>=0):
                print("Fibonacci series upto {0} :".format(n))
                fib(phi,n-1)
        else:
                print("Please enter a positive integer")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the max value:
10
Fibonacci series upto 10 :
0 1 1 2 3 5 8 13 21 34

Test Case - 2
User Output
Enter the max value:
-9
Please enter a positive integer

Test Case - 3
User Output
Enter the max value:
1
Fibonacci series upto 1 :
0

S.No: 17	Exp. Name: ***Write a program to double a given number and add two numbers using lambda()***	Date: 2023-02-24

Aim:

Write a program to double a given number and add two numbers using lambda()

Source Code:

lamda.py

```
num=int(input("Enter a Number : "))
num2=num*2
print("After Doubling Num1 and assigning to Num2  =  {0}".format(num2))
add=lambda num,num2:num+num2
print("Num 1 + Num 2 =  {0}".format(add(num,num2)))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a Number :
10
After Doubling Num1 and assigning to Num2 = 20
Num 1 + Num 2 = 30

S.No: 18	Exp. Name: ***Import a specific function***	Date: 2023-02-24

Aim:

Write a python program to define a module and import a specific function in that module to another program.

Source Code:

import.py

```
import math
print("The value of pi is {0}".format(math.pi))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
The value of pi is 3.141592653589793

S.No: 19	Exp. Name: ***Unique words in the file in alphabetical order***	Date: 2023-02-24

Aim:

Write a program that inputs a text file. The program should print all of the unique words in the file in alphabetical order

Source Code:

alphabetical.py

```
# Write your code here
fname=input("Enter file name: ")
f=open(fname)
lst=list()
words=[]

for line in f:
        words+=line.split()
words.sort()

for word in words:
        if word in lst:
                continue
        else:
                lst.append(word)
                print(word,"1")
```

out.txt

```
hi good morning
hi good morning
hi good morning
how are u
```

out1.txt

```
India
India
America
Africa
Amazon
Amazing
xyz
hello how are you
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter file name:
out.txt
are 1
good 1
hi 1
how 1
morning 1
u 1

Test Case - 2
User Output
Enter file name:
out1.txt
Africa 1
Amazing 1
Amazon 1
America 1
India 1
are 1
hello 1
how 1
xyz 1
you 1

S.No: 20	Exp. Name: ***Write a python program to open and write "hello world" into a file?***	Date: 2023-02-24

Aim:

Write a python program to open and write "hello world" into a file?

Source Code:

filehandhello.py

```
s=str(input("Enter Data : "))
f=open("file1.txt","w")

f.write(s)

f=open("file1.txt","r")
print(f.read())
f.close()
```

file1.txt

```
hello world
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Data : hello world hello world

S.No: 21	Exp. Name: ***Write a python program to write the content "hi python programming" for the existing file.***	Date: 2023-02-24

Aim:

Write a python program to write the content "hi python programming" for the existing file.

Note: If the choice is **1 (**File 1) file name is **file.txt**.
If the choice is **2 (**File 2) file name is **file2.txt**.

Source Code:

filewrite.py

```
print("Select")
print("1.File 1")
print("2.File 2")
ch=int(input("Enter your choice : "))


if(ch==1):
        print("Before Appending the Data : ")
        f=open("file.txt","r")
        print(f.read())


        f=open("file.txt","a")
        f.write("\nHi Python Programming")

        print("After Appending the Data : ")
        f=open("file.txt","r")
        print(f.read())
        f.close()

elif ch==2:
        print("Before Appending the Data : ")
        f=open("file2.txt",'r')
        print(f.read())

        f=open("file2.txt",'a')
        f.write("\nHi Python Programming")

        print("After Appending the Data : ")
        f=open("file2.txt","r")
        print(f.read())
        f.close()
```

file.txt

```
Welcome to CodeTantra!
```

file2.txt

```
Welcome to Mrcet Python Lab!
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Select
1.File 1
2.File 2
Enter your choice :
1
Before Appending the Data :
Welcome to CodeTantra!
After Appending the Data :
Welcome to CodeTantra!
Hi Python Programming

S.No: 22	Exp. Name: ***Write a Python class to convert an integer to a Roman numeral.***	Date: 2023-02-24

Aim:

Write a Python class to convert an integer to a Roman numeral.

Source Code:

roman_num.py

```
def printRoman(number):
        num=[1,4,5,9,10,40,50,90,100,400,500,900,1000]
        sym=["I","IV","V","IX","X","XL","L","XC","C","CD","D","CM","M"]
        i=12
        strr=""
        while number:
                div=number//num[i]
                number%=num[i]
                while div:
                        # print(sym[i],end="")
                        strr+=sym[i]
                        div-=1
                i-=1
        return strr

if __name__=="__main__":
        number=int(input())
        print(printRoman(number))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
12
XII

Test Case - 2
User Output
100
C

Test Case - 3
User Output
56
LVI

S.No: 23	Exp. Name: ***Write the code to display welcome to MRCET and find sum of two numbers using class and methods.***	Date: 2023-02-24

Aim:

1. Write a python Program to display welcome to MRCET by using classes and objects.
2. Write a program to find sum of two numbers using class and methods.

Source Code:

display.py

```
class Disp:
        def welcome():
                print("welcome to mrcet")


class Addition:
        def add(x,y):
                return x+y


d=Disp
d.welcome()

x=int(input())
y=int(input())

a=Addition
summ=a.add(x,y)

print("{0} + {1} = {2}".format(x,y,summ))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
welcome to mrcet
1
2
1 + 2 = 3

Test Case - 2
User Output
welcome to mrcet
22
33
22 + 33 = 55

S.No: 24	Exp. Name: ***Write a python Program to call data member and function using classes and objects.***	Date: 2023-02-24

Aim:

Create a Python class called **Birthdayboy** with three data members:

- An integer called the **currentyear**
- An integer called the **yourBirthyear**
- An integer called the **yourSiblingBirthYear**
- Initialized all data members with the help of parameterized constructor.

Within the class **BirthdayBoy,** create a method called **age** that finds the age gap between you and your sibling by the given inputs.

Once this class has been created, create an instance of the class and call the age method.

Constraint:

birthyear <= currentyear.

Note:

- Refer to the displayed test cases for better understanding.
- Use object oriented approach to solve the listed problem.

Sample Test case :

2022 ----> current year
2001 ----> yourBirthYear
2003 ----> yourSiblingBirthYear
2 -----> Agegap

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Source Code:

Class.py

```
class BirthdayBoy:
        currentYear=0
        yourBirthyear=0
        yourSiblingBirthYear=0

        def __init__(self,currentYear,yourBirthyear,yourSiblingBirthYear):
                self.currentYear=currentYear
                self.yourBirthyear=yourBirthyear
                self.yourSiblingBirthYear=yourSiblingBirthYear

        def age(self,y,s):
                return abs(y-s)

a=int(input())
b=int(input())
c=int(input())









# a is name, b is the age
obj= BirthdayBoy(a,b,c)
print(obj.age(b,c))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
2022
2001
2003
2

Test Case - 2
User Output
2022
2001
1995
6

Test Case - 3

User Output
2022
2022
2003
19

Test Case - 4
User Output
2022
2022
2024
2

S.No: 25	Exp. Name: ***Write the code to read 3 subject marks and display pass or failed***	Date: 2023-02-24

Aim:

Write a program to read 3 subject marks and display pass or failed using class and object.(pass marks = 40)

Source Code:

subject_marks.py

```
class Result:
        def check(m):
                if m<40:
                        print("Fail")
                else:
                        print("Pass")


m1=int(input())
m2=int(input())
m3=int(input())

print("Enter the mark1:",m1)
print("Enter the mark2:",m2)
print("Enter the mark3:",m3)

obj=Result
obj.check(m1)
obj.check(m2)
obj.check(m3)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
67
45
30
Enter the mark1: 67
Enter the mark2: 45
Enter the mark3: 30
Pass
Pass
Fail

Test Case - 2

User Output
23
24
35
Enter the mark1: 23
Enter the mark2: 24
Enter the mark3: 35
Fail
Fail
Fail

S.No: 26	Exp. Name: ***Write a python program using polymorphism with inheritance concepts***	Date: 2023-02-24

Aim:

Create a parent(Base) class **A** and define 2 methods:

3. **__init()__** method (Constructor) with no arguments used to initialize a string type class attribute named **value** as input from the user.
4. **function()** method with no arguments used to print the **Reverse** of second half of the class attribute **value**

Create a child(Derived) class **B** and define 2 methods:

5. **__init()__** method (Constructor) with no arguments used to initialize a string type class attribute named **value** as input from the user.
6. **function()** method with no arguments is used to print **thrice** the class attribute **value.**

In the driver code,

7. create an instance/object of the parent class **A** and call the method using the object of the parent class.
8. create an instance/object of the child class **B** and call the method using the object of the child class.

Constraints:

1 <= length of the string <= 10

Note:

- Refer to the displayed test cases for better understanding.

Sample Test case:

inheritance
ecnat
Tool
ToolToolTool

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Source Code:

inheritance.py

```
class A:
#define the methods.
        pass
        def function(self):
                pass



class B:
#define the methods.
        pass
        def function(self):
                pass



a = A()
#create an instance of parent class.
str1=str(input())

a.function()
#call the method using parent class object.
n=len(str1)
m=n//2
if(n%2!=0):
        m+=1
rev=""
for i in range(m,n):
        rev+=str1[i]
print(rev[::-1])
b = B()
#create an instance of child class.
str2=str(input())

b.function()
#call the method using child class object.
print(str2*3)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
inheritance
ecnat
Tool
ToolToolTool

S.No: 27	Exp. Name: ***Write a program for Multiple Inheritance.***	Date: 2023-02-24

Aim:

Write a program to understand **Multiple Inheritance**.

Fill the **missing code** in the below program by following the below instructions:

- Create a `base class` **vehicle** with `__init__` method, and pass **self**, **name**, **price** and **regno** as arguments
- Derive a `class` **car** from `base class` **vehicle**
- Define a method `__init__` in the `class` **car** which contains the `parameters` **self**, **name**, **price**, **regno** and **gear**
- Derive a `class` **boat** from `class` **vehicle**
- Derive a `class` **hover** from `class` **car** and **boat**

Sample Input and Output:

```
car      toyota      1500000     car2121     auto
boat     maruti      1000000     boat0121
hover    toyota      1500000     hover1212   manual
class car inherits from class vehicle
```

Source Code:

MultipleInher.py

```
class vehicle:
        '''General Vehicle class'''
        def __init__(self,name,price,regno):
                self.name=name
                self.price=price
                self.price=price
                self.regno=regno



class car(vehicle):
        ''' Class car inherits from Vehicle'''
        def __init__(self,name,price,regno,gear):
                self.name=name
                self.price=price
                self.regno=regno
                self.gear=gear

class boat(vehicle):
        pass

class hover(car, boat):
        pass

c1 = car('toyota', 1500000, 'car2121', 'auto')
b1 = boat('maruti', 1000000, 'boat0121')
h1 = hover('toyota', 1500000, 'hover1212', 'manual')
print(type(c1).__name__, "\t", c1.name, "\t", c1.price, "\t", c1.regno, "\t", c1.gear)
print(type(b1).__name__, "\t", b1.name, "\t", b1.price, "\t", b1.regno, "\t")
print(type(h1).__name__, "\t", h1.name, "\t", h1.price, "\t", h1.regno, "\t", h1.gear)
print(c1.__doc__)
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
car     toyota          1500000         car2121         auto
boat    maruti          1000000         boat0121
hover   toyota          1500000         hover1212       manual
 Class car inherits from Vehicle
```

S.No: 28	Exp. Name: ***Write the code to implement single inheritance concept.***	Date: 2023-02-24

Aim:

Your task is to:

- Define a Base class **Calculation** with methods **addition** and **subtraction.**
- **Addition:** Adds two integers.
- **Subtraction:** subtracts the second integer from the first one.
- Derive a child class from the base class **My_Calculation** with the method **multiplication.**
- **Multiplication:** Multiplies two integers.
- Using the child class to call all the methods of both Parent and child classes.

Constraints:

- -10000 <= Integers <= 10000

Note:

- **Refer to the displayed Test cases for better understanding.**

Sample Test case:

10 ----> Enter integer 1
20 ----> Enter integer 2
30 ----> Display the addition.
-10 ----> Display the subtraction.
200 ----> Display the multiplication.

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Source Code:

inheritance.py

```
class Calculation:
    # def __init__(self,x,y):
    #   self.x=x
    #   self.y=y

    def addition(self,x,y):

        ''' Missing code '''
        return (x+y)

    def subtraction(self,x,y):

        return (x-y)

class My_Calculation(Calculation):
    def multiplication(self, x,y):
        return x*y

x=int(input())
y=int(input())
b=Calculation()
a=My_Calculation()
print(b.addition(x,y))
print(b.subtraction(x,y))
print(a.multiplication(x,y))
''' Missing code '''
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
10
20
30
-10
200

Test Case - 2
User Output
1
2
3
-1
2

Test Case - 3
User Output

-200
-300
-500
100
60000

S.No: 29	Exp. Name: ***Implement data-abstraction in python, try to create object of abstract class.***	Date: 2023-02-24

Aim:

Your task is to:

- Define the **abstract class Animal** which is derrived from **ABC class of the abc module** and inherits the properties of the **ABC class**.
- Define the abstract method **move()** in the abstract class **Animal**.
- Create various subclasses which are derrived from the base **class Animal**. The subclasses are as follows :
- **Human:** Define a method **move()** that prints the **"I can walk and run"**
- **Snake:** Define a method **move()** that prints **"I can crawl".**
- **Lion:** Define a method **move()** that prints **"I can roar".**

Create the object/instance of the Base class**(Animal)** as well as the derrived classes **Human, Snake, and Lion** and with their respective objects invoke (Call) the **move()** method.

Note:

- Here the test cases are not displayed to you, write your code step by step as mentioned in the problem statement to achieve the result.

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Source Code:

interface.py

```
from abc import ABC

class Animal(ABC):
        def move():
                pass

class Human(Animal):
        def move():
                print("I can walk and run")


class Snake(Animal):
        def move():
                print("I can crawl")


class Lion(Animal):
        def move():
                print("I can roar")

#create the objects of Animal class and all other derrived classes (subclasses)
# ani-Animal
h=Human
s=Snake
l=Lion
#Invoke the the move() method for all the classes.

h.move()
s.move()
l.move()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
I can walk and run
I can crawl
I can roar