# M.Sc. (Five Year Integrated) in Computer Science (Artificial Intelligence & Data Science)

## First Semester

## Laboratory Record

## 21-805-0106: PYTHON PROGRAMMING LAB

*Submitted in partial fulfillment*
*of the requirements for the award of degree in*
*Master of Science (Five Year Integrated)*
*in Computer Science (Artificial Intelligence & Data Science) of*
*Cochin University of Science and Technology (CUSAT)*
*Kochi*



*Submitted by*

**ABHIN P T**
**(80521002)**

**DEPARTMENT OF COMPUTER SCIENCE**
**COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)**
**KOCHI-682022**

**MARCH 2022**

# DEPARTMENT OF COMPUTER SCIENCE
## COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
### KOCHI, KERALA-682022



*This is to certify that the practical laboratory record for **21-805-0106: Python Programming Lab** is a record of work carried out by **ABHIN P T(80521002)**, in partial fulfillment of the requirements for the award of degree in **Master of Science (Five Year Integrated)** in **Computer Science (Artificial Intelligence & Data Science)** of Cochin University of Science and Technology (CUSAT), Kochi. The lab record has been approved as it satisfies the academic requirements in respect of the first semester laboratory prescribed for the Master of Science (Five Year Integrated) in Computer Science degree.*

**Faculty Member in-charge**

Dr. Shailesh S                                                                Dr. Philip Samuel

Assistant Professor                                                      Professor and Head

Department of Computer Science            Department of Computer Science

CUSAT                                                                                      CUSAT

# Table of Contents

# ARITHMETIC OPERATORS

**AIM**

Develop a program to read a four-digit number and find its

a. Sum of digits

b. Reverse

c. Difference between the product of digits at the odd position and the product of digits at the even position.

**THEORY**

input (), Strings, Arithmetic operators

**PROGRAM**

```
def getnum():
    num=int(input('Enter a 4 digit number'))   # taking input from user
    return(num)


num = getnum()


def calculate(num):
    temp=num
    # assigning value of sum to temp variable
    sum=0                                   # assigning 0 to sum
    revnum=0
    evn=1
    odd=1
    x=temp%10                               # to get unit digit
    sum=sum+x                               # taking sum of digits
    revnum=(revnum*10)+x
    # storing individual digits in reverse order
    evn=evn*x                               # storing even position digit
    temp=temp//10
    # integer division by 10 to get quotient
    x=temp%10
    sum=sum+x
    revnum=(revnum*10)+x
    odd=odd*x                               # storing odd position digit
    temp=temp//10
    x=temp%10
    sum=sum+x
```

```
    revnum=(revnum*10)+x
    evn=evn*x
# taking product of even position numbers
    temp=temp//10
    x=temp%10
    sum=sum+x
    revnum=(revnum*10)+x
    odd=odd*x
# taking product of odd position numbers
    temp=temp//10
    diff=odd-evn
    return sum, revnum, diff
    # difference of odd position product and even position product


def display():
    sum, revnum, diff = calculate(num)
    print('Sum : ',sum)                          # printing sum of digit
    print('Reverse number : ',revnum)        # printing reverse number
    print('Difference between the product of digits at the odd position and
    the product of digits at the even position : ',diff)
    # printing product difference


display()
```

## SAMPLE INPUT-OUTPUT

```
Enter a 4 digit number1463
Sum :  14
Reverse number :  3641
Difference between the product of digits at the odd position and the product of digits at the even position :  -6
```

## TEST CASES

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Sum of digits | 1463 | 14 | 14 | True |
| 2 | . Reverse | 1463 | 3641 | 3641 | True |
| 3 | Difference between the product of digits at the odd position and product of digits at the even position | 1463 | -6 | -6 | True |

## GITHUB LINK

Click Here

## RESULT

Program executed successfully,result obtained.

# AREA OF TRIANGLE

**AIM**

Develop a program to read the three sides of two triangles and calculate the area of both. Define a function to read the three sides and call it. Also, define a function to calculate the area. Print the total area enclosed by both triangles and each triangle's contribution (towards it.

**THEORY**

Datatype, Functions, Expressions, Built-in functions

**PROGRAM**

```python
def getdata():     # function to read triangles side
  a=int(input("Enter the first side of triangle : "))
  b=int(input("Enter the second side of Triangle : "))
  c=int(input("Enter the third side of Triangle : "))
  return a,b,c

def area():       # function to calculate area of triangle
  a, b, c= getdata()
  s=(a+b+c)/2
  ar=(s*(s-a)*(s-b)*(s-c))**0.5
  return ar

a1=area()         # assigning area of triangle 1
a2=area()         # assigning area of triangle 2
print('Total area of 2 triangles : ', a1+a2)  # printing total area
con1=(a1/(a1+a2))*100                         # area contribution of triangle 1
con2=(a2/(a1+a2))*100                         # area contribution of triangle 2
print('Contribution of triangle 1 : ',con1)
print('Contribution of triangle 2 : ',con2)
```

**SAMPLE INPUT-OUTPUT**

```
Enter the first side of triangle : 3
Enter the second side of Triangle : 4
Enter the third side of Triangle : 5
Enter the first side of triangle : 3
Enter the second side of Triangle : 4
Enter the third side of Triangle : 5
Total area of 2 triangles :  12.0
Contribution of triangle 1 :  50.0
Contribution of triangle 2 :  50.0
```

**TEST CASES**

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Area of Triangle | 3,4,5 | 6 | 6 | True |
| 2 | Area of Triangle | 7,5,10 | 16.248 | 16.248 | True |
| 3 | Total Area | 6, 16.248 | 22.248 | 22.248 | True |
| 4 | Triangles Contribution 1 | 6 | 26.968 | 26.968 | True |
| 5 | Triangle's contribution 2 | 22.248 | 73.0313 | 73.0313 | True |

**GITHUB LINK**

Click Here

**RESULT**

Program executed successfully,result obtained.

# PAY SLIP

**AIM**

Develop a program to read the employee's name, code, and basic pay and calculate the gross salary, deduction, and net salary according to the following conditions. Define a function to find each of the components. Finally, generate a payslip.

**THEORY**

Conditional Branching

**PROGRAM**

```python
name=str(input("Enter employee's name : "))        # reading employees name
code=str(input("Enter employee's code : "))        # reading employees code
bp=float(input("Enter employee's basic pay :"))    # reading employees basic pay

def salary():
# function to calculate net salary
# using else if
  if(bp>50000):
    gs=bp+(0.25*bp)+(0.11*bp)+7000
    d=80+(0.12*bp)+(0.20*bp)
    # for calculating net salary in different range
    net_slry=gs-d
  elif(30000<bp<50000):
    gs=bp+(bp*0.11)+(bp*0.075)+5000
    d=60+(0.11*bp)(0.11*bp)
    net_slry=gs-d
  elif(10000<bp<30000):
    gs=bp+(0.075*bp)+(0.05*bp)+2500
    d=60+(0.08*bp)+0
    net_slry=gs-d
  else:
    gs=bp+(0.05*bp)+(0.025*bp)+500
    d=20+(0.08*bp)+0
    net_slry=gs-d


  return net_slry

print("Eployee's name : ",name)                    # printing payslip
```

```
print("Employee's code : ",code)
print("Employee's basic pay : ",bp)
print("Employee's Net Salary : ",salary())
```

## SAMPLE INPUT-OUTPUT

```
Enter employee's name : Abhin
Enter employee's code : 110382
Enter employee's basic pay :50001
Eployee's name :  Abhin
Employee's code :  110382
Employee's basic pay :  50001.0
Employee's Net Salary :  58921.04
```

## TEST CASES

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Net Salary | 10001 | 12891.045 | 12891.045 | True |
| 2 | Net Salary | 30010 | 33899.65 | 33899.65 | True |
| 3 | Net Salary | 50001 | 58921.04 | 58921.04 | True |
| 4 | Net Salary | 5000 | 5455.00 | 5455.00 | True |

## GITHUB LINK

Click Here

## RESULT

Program executed successfully,result obtained.

# HAPPY NUMBER OR NOT

## AIM

Develop a program to perform the following task: a. Define a function to check whether a number is happy or not. b. Define a function to print all happy numbers within a range. c. Define a function to print first N happy numbers. A happy number is a number defined by the following process: • Starting with any positive integer, replace the number with the sum of the squares of its digits. • Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. • Those numbers for which this process ends in 1 are happy. Note: if a number is not being happy after 100 iterations, consider it sad.

## THEORY

Loops – for, while, Nested loops

## PROGRAM

```
num=int(input("Enter a positive integer : "))  # reading a positive integer

def hppy_not(num):
# function to check the number happy or not
  is_happy = False
  n=0
  cnt=0
  while(n<=100):
    temp=num
    sum=0
    while(temp>0):
     temp=temp//10
     cnt=cnt+1
    for i in range(1, cnt+1):
      x=num%10
      sum=sum+(x**2)
      num=num//10
    if(sum==1):
      is_happy = True
      break
    else:
     n=n+1
     num=sum
```

```
    return is_happy
if hppy_not(num)==True:
# printing happy number or not by checking returned boolian value
    print("Happy number")
else:
    print("Not a Happy Number")



def hppy_range():
# function to print happy number with in a range
    lower_range=int(input("Enter lower range : "))
    upper_range=int(input("Enter higher range : "))
    for i in range(lower_range, upper_range+1):
        if hppy_not(i)==True:
            print(i)

hppy_range()                        # calling function

def hppy_n():                       # function to print first n happy number
    no_off=int(input("Enter N : "))
    count=1
    i=0
    while(count<=no_off):
        if hppy_not(i)==True:
            print(i)
            count=count+1
        i=i+1

hppy_n()                            # calling function
```

## SAMPLE INPUT-OUTPUT

```
Enter a positive integer : 13
Happy number
Enter lower range : 1
Enter higher range : 10
1
7
10
Enter N : 3
1
7
10
```

## TEST CASES

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Number is happy or not | 13 | Happy | Happy | True |
| 2 | Happy numbers within a range | 1, 10 | 1, 7, 10 | 1, 7, 10 | True |
| 3 | First N happy numbers | 3 | 1, 7, 10 | 1, 7, 10 | True |

## GITHUB LINK

Click Here

## RESULT

Program executed successfully,result obtained.

# STRING FUNCTIONS

**AIM**

Develop a program to read a string and perform the following operations: • Print all possible substrings. • Print all possible substrings of length K. • Print all possible substrings of length K with N distinct characters. • Print substring(s) of length maximum length with N distinct characters. • Print all palindrome substrings. Define function for each of the task.

**THEORY**

Strings, String functions, Slicing

**PROGRAM**

```python
strg=str(input("Enter a string : "))  # reading a string

def substring():                       # function to print substrings
  print("Substrings : ")
  for i in range(0,len(strg)+1):
    for j in range(i+1,len(strg)+1):
      sub=strg[i:j]
      print(sub)




def sub_length():
# function to print substring with specific length
  length=int(input("Enter the length of substring : "))
  print("Substring with length %d : "%length)
  for i in range(0,len(strg)+1):
    for j in range(i+1,len(strg)+1):
      sub=strg[i:j]
      if(len(sub)==length):
        print(sub)




def sub_length_distinct():
# function to print substring with specific length and
# specific distinct character
  length=int(input("Enter the length substring : "))
```

```python
    distinct=int(input("Enter the number of distinct characters : "))
    print("Substring with length %d and %d distinct characters : "%(length,distinct))
    for i in range(0,len(strg)+1):
      for j in range(i+1,len(strg)+1):
        sub=strg[i:j]
        if(len(sub)==length and len(set(sub))==distinct):
          print(sub)




def sub_max_distinct():
# function to print substring with specific distinct character
# with maximum length
    ls=[]
    distinct=int(input("Enter the number of distinct characters : "))
    print("Substring with %d distinct character with max length : "%distinct)
    for i in range(0,len(strg)+1):
      for j in range(i+1,len(strg)+1):
        sub=strg[i:j]
        if(len(set(sub))==distinct):
          ls.append(sub)
    #print(ls)
    max_leng=len(max(ls, key=len))
    for k in range(len(ls)):
      if(len(ls[k])==max_leng):
        print(ls[k])

def sub_pallian():
  print("Palindrome substrings : ")
# function to print palindrome sub strings
  for i in range(0,len(strg)+1):
    for j in range(i+1,len(strg)+1):
      sub=strg[i:j]
      rev=sub[::-1]
      if(sub==rev):
        print(sub)



substring()
sub_length()
sub_length_distinct()
```

sub_max_distinct()

sub_pallian()

## SAMPLE INPUT-OUTPUT

```
Enter a string : aaabcd
Substrings :
a
aa
aaa
aaab
aaabc
aaabcd
a
aa
aab
aabc
aabcd
a
ab
abc
abcd
b
bc
bcd
c
cd
d
Enter the length of substring : 4
Substring with length 4 :
aaab
aabc
abcd
Enter the length substring : 2
Enter the number of distinct characters : 2
Substring with length 2 and 2 distinct charectors :
ab
bc
cd
Enter the number of distinct characters : 2
Substring with 2 distinct charector with max length :
aaab
Palindrome substrings :
a
aa
aaa
a
aa
a
b
c
d
```

**TEST CASES**

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | All possible substrings | abb | a,ab,abb,b,bb,b | a,ab,abb,b,bb,b | True |
| 2 | Substrings of length K | 2 | ab,bb | ab,bb | True |
| 3 | Substrings of length K with N distinct characters | 1, 1 | a, b, b | a, b, b | True |
| 4 | Length maximum length with N distinct characters | 2 | abb | abb | True |

**GITHUB LINK**

Click Here

**RESULT**

Program executed successfully,result obtained.

# RABBIT PAIRS

**AIM**

Suppose a newly born pair of rabbits, one male and one female, are put in a field. Rabbits can mate at the age of one month so that at the end of its second month, a female has produced another pair of rabbits. Suppose that our rabbits never die and that the female always produces one new pair every month from the second month. Develop a program to show a table containing the number of pairs of rabbits in the first N months.
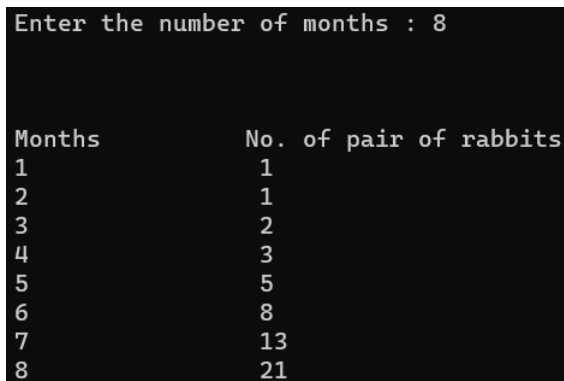
**THEORY**

Critical thinking, Loops, formatted io.

**PROGRAM**

```python
def num_rabbits():
  n=int(input('Enter the number of months : ')) #reading number of months
  a = 0
  b = 1
  c=0
  print("\n\n")
  print("Months\t\tNo. of pair of rabbits")
  for i in range(1,n+1):
  #calculating number of pair of rabbits
  #it is fibinocci series
    c = b
    b = a+b
    a = c
    print(i,"\t\t",a)


num_rabbits()
```

**SAMPLE INPUT-OUTPUT**

```
Enter the number of months : 8



Months          No. of pair of rabbits
1               1
2               1
3               2
4               3
5               5
6               8
7               13
8               21
```

**TEST CASES**

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Number of pairs of Rabbits | 1 | 1 | 1 | True |
| 2 | Number of pairs of Rabbits | 2 | 1 | 1 | True |
| 3 | Number of pairs of Rabbits | 3 | 2 | 2 | True |

**GITHUB LINK**

Click Here

**RESULT**

Program executed successfully,result obtained.

# LIST OF INTEGERS

**AIM**

Write a program to read a string containing numbers separated by a space and convert it as a list of integers. Perform the following operations on it. 1. Rotate elements in a list by 'k' position to the right 2. Convert the list into a tuple using list comprehension 3. Remove all duplicates from the tuple and convert them into a list again. 4. Create another list by putting the results of the evaluation of the function $() = 2 -$ with each element in the final list 5. After sorting them individually, merge the two lists to create a single sorted list.

**THEORY**

List, tuple, set, list comprehension

**PROGRAM**

```
s = input(str('Enter the numbers : '))
#reading a string containing numbers separated by a space
#s = '1 2 3 4 5 2 2 1'
l = [int(i) for i in s.split()]          #converting it as a list of integers
print(s)
print(l)



def rotate(l):
  k = int(input("Enter the 'k' position to rotate elements in a list : "))
  for i in range (k):
    l.insert(0,l.pop())                    #rotating elements k position to right
  print("Rotate elements in a list by 'k' position to the right : ",l)


rotate(l)

def conversion(l):
  t = tuple([i for i in l])
  #converting to tuple using list comprehension
  print("Convert the list into a tuple using list comprehension : ",t)
  return(t)


t=conversion(l)

def duplicate(t):
  st = set(t)                            #remove all duplicates from the tuple
```

```
  print("Remove all duplicates from the tuple : ",st)
  l = list(st)                          #converting to list
  print("Remove all duplicates from the tuple and convert them into a list again : ",l)
  return(l)


l=duplicate(t)


def lsfuntion(l):
  l2 = [((i**2)-i) for i in l]
  #forming a list by evaluation of the function () = 2 {  with each elem
  ent in the final list
  print("Create another list by putting the results of the evaluation
  of the function () = 2 {   : ",l2)
  return(l2)


l2=lsfuntion(l)


def sort_merge(l,l2):
  l.sort()                #sorting each list
  l2.sort()
  l3 = l + l2             #merging lists
  l4=l3.sort()              #sorting list
  print("Sorting them individually, merge the two lists to create
  a single sorted list : ",l3)


sort_merge(l,l2)
```

**SAMPLE INPUT-OUTPUT**

```
Enter the numbers : 1 2 3 1 1 2
1 2 3 1 1 2
[1, 2, 3, 1, 1, 2]
Enter the 'k' position to rotate elements in a list : 3
Rotate elements in a list by 'k' position to the right :  [1, 1, 2, 1, 2, 3]
Convert the list into a tuple using list comprehension :  (1, 1, 2, 1, 2, 3)
Remove all duplicates from the tuple :  {1, 2, 3}
Remove all duplicates from the tuple and convert them into a list again :  [1, 2, 3]
Create another list by putting the results of the evaluation of the function f(x) = x2 - x  : [    0, 2, 6]
Sorting them individually, merge the two lists to create a single sorted list :  [0, 1, 2, 2, 3, 6]
```

**TEST CASES**

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Rotate elements 2 position | 1,2,2,3 | 2, 3, 1, 2 | 2, 3, 1, 2 | True |
| 2 | Remove all duplicates | 1, 2, 2, 3 | 1, 2, 3 | 1, 2, 3 | True |
| 3 | Evaluating function | 1,2,3 | 0,2,6 | 0,2,6 | True |
| 4 | Merging & Sorting | 1,2,3,0,2,6 | 0,1,2,2,3,6 | 0,1,2,2,3,6 | True |

**GITHUB LINK**

Click Here

**RESULT**

Program executed successfully,result obtained.

# FILE HANDLING

**AIM**

Read the file 'iris.json' as a text file : 1. Create a list having each line of the file as an element 2. Convert it into a list of dictionary objects. 3. Show the details of all flowers whose species is "setosa". 4. Print the minimum petal area and max sepal area in each species 5. Sort the list of dictionaries according to the total area are sepal and petal.

**THEORY**

JSON, dictionary

**PROGRAM**

```
def read_file(filename):
  a = open(filename,"r")                  #reading the file iris.json
  data = a.readlines()                    #a list having each line of the file as an element
  a.close()
  return data


data = read_file("iris.json")


#print(data)
import json

def read_dictionary1(file):
  a = open(file,"r")
  dictionary1 = json.load(a)              #Convert into dictionary1 data objects.
  dictionary11 = dictionary1
  a.close()
  print(dictionary1)
  return(dictionary1)


dictionary1 = read_dictionary1("iris.json")


#[print(i['species']) for i in dictionary1]


#def print_setosa(dictionary1):
for i in dictionary1:
  if i['species'] == 'setosa':
  #printing the details of all flowers whose species is "setosa".
```

```
      print(i)

#print_setosa(dictionary1)

l_species = []

def species(dictionary1):
  for i in dictionary1:                    #forming a list of type of species
    if i['species'] not in l_species:
      l_species.append(i['species'])
  return(l_species)

l_species = species(dictionary1)
#print(l_species)

l_area_sepal1 = []
l_area_petal1 = []
l_area_sepal2 = []
l_area_petal2 = []
l_area_sepal3 = []
l_area_petal3 = []

def area(dictionary1,l_area_petal1,l_area_petal2,l_area_petal3,l_area_sepal1,
l_area_sepal2,l_area_sepal3,l_species):

  for i in dictionary1:
    ptlarea = (i['sepalLength']*i['sepalWidth'])
    #calculating petal area of each flower
    splarea = (i['petalLength']*i['petalWidth'])
    #calculating sepel area of each flower
    totalarea = ptlarea + splarea
    #calculating total area of each flower
    i.update({'totalArea' : totalarea})
    #adding total area to the dictionary1
    if i['species'] == l_species[0]:
      l_area_sepal1.append(i['sepalLength']*i['sepalWidth'])
      l_area_petal1.append(i['petalLength']*i['petalWidth'])
    elif i['species'] == l_species[1]:
    #calculating sepal and petal area of different species
      l_area_sepal2.append(i['sepalLength']*i['sepalWidth'])
      l_area_petal2.append(i['petalLength']*i['petalWidth'])
```

```
    elif i['species'] == l_species[2]:
      l_area_sepal3.append(i['sepalLength']*i['sepalWidth'])
      l_area_petal3.append(i['petalLength']*i['petalWidth'])


  # print(l_area_sepal1)
  # print(l_area_petal1)
  # print(l_area_sepal2)
  # print(l_area_sepal2)
  # print(l_area_sepal3)
  # print(l_area_sepal3)


  print(l_species[0])
  print("Greatest sepal area : ",max(l_area_sepal1))
  print("Minimum petal area  : ",min(l_area_petal1))
  print(" ")
  print(l_species[1])
  #printing the max sepal and minimum petal area of different species
  print("Greatest sepal area : ",max(l_area_sepal2))
  print("Minimum petal area  : ",min(l_area_petal2))
  print(" ")
  print(l_species[2])
  print("Greatest sepal area : ",max(l_area_sepal3))
  print("Minimum petal area  : ",min(l_area_petal3))
  print("    ")
  #print(dictionary1)
  print("    ")
  return(dictionary1)


dictionary1 = area(dictionary1,l_area_petal1,l_area_petal2,
l_area_petal3,l_area_sepal1,l_area_sepal2,l_area_sepal3,l_species)


def sorted_dictionary1(dictionary1):
  sorteddictionary1 = (sorted(dictionary1, key = lambda i:i ['totalArea'] ))
  #Sort the list of dictionaries according to the total area of sepal and petal
  print("                                Details of sorted list")
  print("----------------------------------------------------------------
  --------------------------------")
  print('Sepal Length    Sepal Width    Petal Length     Petal Width
  Species       Total Area     ')


  for i in sorteddictionary1:
```

```
print(' ',i["sepalLength"]," \t","\t",i["sepalWidth"],'\t\t',i['petalLength'],
    '\t\t',i['petalWidth'],'\t\t',i['species'],'\t',i['totalArea'])
```

```
sorted_dictionary1(dictionary1)
```

```
# print(sorteddictionary1)
```

## SAMPLE INPUT-OUTPUT

```
{'sepalLength': 5.5, 'sepalWidth': 3.5, 'petalLength': 1.3, 'petalWidth': 0.2, 'species': 'setosa'}
{'sepalLength': 4.9, 'sepalWidth': 3.6, 'petalLength': 1.4, 'petalWidth': 0.1, 'species': 'setosa'}
{'sepalLength': 4.4, 'sepalWidth': 3.0, 'petalLength': 1.3, 'petalWidth': 0.2, 'species': 'setosa'}
{'sepalLength': 5.1, 'sepalWidth': 3.4, 'petalLength': 1.5, 'petalWidth': 0.2, 'species': 'setosa'}
{'sepalLength': 5.0, 'sepalWidth': 3.5, 'petalLength': 1.3, 'petalWidth': 0.3, 'species': 'setosa'}
{'sepalLength': 4.5, 'sepalWidth': 2.3, 'petalLength': 1.3, 'petalWidth': 0.3, 'species': 'setosa'}
{'sepalLength': 4.4, 'sepalWidth': 3.2, 'petalLength': 1.3, 'petalWidth': 0.2, 'species': 'setosa'}
{'sepalLength': 5.0, 'sepalWidth': 3.5, 'petalLength': 1.6, 'petalWidth': 0.6, 'species': 'setosa'}
{'sepalLength': 5.1, 'sepalWidth': 3.8, 'petalLength': 1.9, 'petalWidth': 0.4, 'species': 'setosa'}
{'sepalLength': 4.8, 'sepalWidth': 3.0, 'petalLength': 1.4, 'petalWidth': 0.3, 'species': 'setosa'}
{'sepalLength': 5.1, 'sepalWidth': 3.8, 'petalLength': 1.6, 'petalWidth': 0.2, 'species': 'setosa'}
{'sepalLength': 4.6, 'sepalWidth': 3.2, 'petalLength': 1.4, 'petalWidth': 0.2, 'species': 'setosa'}
{'sepalLength': 5.3, 'sepalWidth': 3.7, 'petalLength': 1.5, 'petalWidth': 0.2, 'species': 'setosa'}
{'sepalLength': 5.0, 'sepalWidth': 3.3, 'petalLength': 1.4, 'petalWidth': 0.2, 'species': 'setosa'}
setosa
Greatest sepal area :  25.080000000000002
Minimum petal area  :  0.11000000000000001

versicolor
Greatest sepal area :  22.400000000000002
Minimum petal area  :  3.3

virginica
Greatest sepal area :  30.02
Minimum petal area  :  7.5
```

## TEST CASES

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Printing particular details of a json file | json file | Minimum petal area and maximum sepal area in each species | Minimum petal area and maximum sepal area in each species | True |
| 2 | Adding total area to dictionary | json file | Adding total area to dictionary | Adding total area to dictionary | True |

## GITHUB LINK

Click Here

## RESULT

Program executed successfully,result obtained.

# VOLUME: AREA RATIO

## AIM

Write a program to create a class Box with data members length, breadth, height, area, and volume. Provider constructor that enables initialization with one parameter (for cube), two parameters (for square prism) three parameters (rectangular prism). Also, provide functions to calculate area and volume. Create a list of N boxes with random measurements and print the details of the box with maximum volume: area ratio.

## THEORY

Class, objects, constructor

## PROGRAM

```python
class Box:                                #creating class box
  __length = None
  __breadth = None
  __height = None
  __area = None
  __volume = None


  def __init__(self, *p):
  #multiple constructor to take different number of arguments
    if len(p) == 1:
      self.__length = p[0]
      self.__breadth = p[0]
      self.__height = p[0]
    elif len(p) == 2:
      self.__length = p[0]
      self.__breadth = p[0]
      self.__height = p[1]
    elif len(p) == 3:
      self.__length = p[0]
      self.__breadth = p[1]
      self.__height = p[2]


  def volume(self):
    self.__volume = self.__length*self.__breadth*self.__height

    #function to calculate area and volume
```

```python
    def area(self):
      self.__area = 2*(self.__length*self.__length + self.__breadth*self.__breadth
      + self.__height*self.__height)


    def display(self):
      print("Box Details")
      #function to print details of box
      print("Length  = ",self.__length)
      print("Breadth = ",self.__breadth)
      print("Height  = ",self.__height)
      print("Area    = ",self.__area)
      print("Volume  = ",self.__volume)


    def ratio(self):
    #function to calculate volume area ratio
      r= (self.__volume)/(self.__area)
      return r

import random

n=int(input("Enter the number of objects that want to made : "))

def putdata():
  ls1 = []
  count=0
  for i in range(0,n,3):
  #providing random inputs
    while(count!=n):
      ls1.insert(i,Box(random.randrange(1,50,1)))
      count = count+1
      ls1.insert(i+1,Box(random.randrange(1,50,1),random.randrange(1,50,1)))
      count = count+1
      ls1.insert(i+2,Box(random.randrange(1,50,1),random.randrange(1,50,1),
      random.randrange(1,50,1)))
      count = count+1
      break;
  return(ls1)

ls1=putdata()

putdata()
```

```python
for i in range(0,n):          #sorting area and volume
  ls1[i].area()
  ls1[i].volume()


ls2 = []

def display_fn():
  for i in range(0,n):
    ls2.append(ls1[i].ratio())
    ls1[i].display()
    print("Ratio   : ",ls1[i].ratio())
    print(" ")

display_fn()

def display_max():
  for i in range(0,n):
  #details of the box with maximum volume: area ratio.
    if max(ls2) == ls1[i].ratio():
      print(" ")
      ls1[i].display()
      print("Greatest Ratio",ls1[i].ratio())

display_max()
```

## SAMPLE INPUT-OUTPUT

```
Enter the number of objects that want to made : 3
Box Details
Length  =  3
Breadth =  3
Height  =  3
Area    =  54
Volume  =  27
Ratio   :  0.5

Box Details
Length  =  10
Breadth =  10
Height  =  3
Area    =  418
Volume  =  300
Ratio   :  0.7177033492822966

Box Details
Length  =  33
Breadth =  44
Height  =  22
Area    =  7018
Volume  =  31944
Ratio   :  4.551724137931035


Box Details
Length  =  33
Breadth =  44
Height  =  22
Area    =  7018
Volume  =  31944
Greatest Ratio 4.551724137931035
```

## TEST CASES

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Generate random values | None | Random Values | Random Values | True |
| 2 | Area | 10,10,3 | 418 | 418 | True |
| 3 | Volume | 10,10,3 | 300 | 300 | True |
| 4 | Ratio | 418,300 | 0.717 | 0.717 | True |

## GITHUB LINK

Click Here

## RESULT

Program executed successfully,result obtained.

# 3D Shapes

**AIM**

Write a program to create a parent class, 3DShapes, with methods printVolume() and printArea(), which prints the Volume and Area, respectively. Create classes Cylinder and Sphere by inheriting 3DShapes class. Using these child classes, calculate and print the volume and area of a cylinder and sphere.

**THEORY**

Inheritance

**PROGRAM**

```
class Shapes:                          #parent class shapes
  _volume = None
  _area   = None

  def printVolume(self):
    print("Volume = ",self._volume)    #public member functions of class shapes
  def printArea(self):
    print("Area = ",self._area)



class Cylinder(Shapes):                #class cylinder by inheriting shapes
  __height = None
  __radius = None

  def __init__(self,r,h):              #constructor
    self.__height=h
    self.__radius=r

  def calcArea(self):
    self._area= 6.28*self.__radius*(self.__height + self.__radius)
    #functions to calculate area and volume of cylinder

  def calcVolume(self):
    self._volume=3.14*self.__radius*self.__radius*self.__height



class Sphere(Shapes):                  #class sphere by inheriting shapes
```

```
    __radius = None

  def __init__(self,r):              #constructor
    self.__radius = r

  def calcVolume(self):
    self._volume = 4.19*self.__radius*self.__radius*self.__radius
    #function to calculate area and volume of sphere

  def calcArea(self):
    self._area = 12.56*self.__radius*self.__radius

r1=float(input("Enter the radius of Sphere : "))
#reading values from user
print("---Sphere---")
s = Sphere(r1)
s.calcVolume()
s.calcArea()
s.printVolume()
s.printArea()
print(" ")


r2=float(input("Enter the radius of Cylinder : "))
#reading values from user
h=float(input("Enter the height of cylinder : "))
print("Cylinder---")
c= Cylinder(r2,h)
c.calcArea()
c.calcVolume()
c.printArea()
c.printVolume()
```

**SAMPLE INPUT-OUTPUT**

```
Enter the radius of Sphere : 5
---Sphere---
Volume =  523.7500000000001
Area =  314.0

Enter the radius of Cylinder : 10
Enter the height of cylinder : 5
Cylinder---
Area =  942.0000000000001
Volume =  1570.0
```

**TEST CASES**

| Test Cases No. | Description | Input | Expected Outcome | Actual Outcome | Result |
|---|---|---|---|---|---|
| 1 | Volume of Sphere | 5 | 525.75 | 525.75 | True |
| 2 | Area of Sphere | 5 | 314 | 314 | True |
| 3 | Volume of Cylinder | 10, 5 | 1570 | 1570 | True |
| 4 | Area of Cylinder | 10, 5 | 942 | 942 | True |

**GITHUB LINK**

Click Here

**RESULT**

Program executed successfully,result obtained.