In [1]:

```
"""
Write a python program which creates a class named Cone and write a
function calculate_area which calculates the area of the Cone.
Sample Execution:
Please enter the radius of the cone: 4
Please enter the height of the cone: 5
Area of a cone with radius: 4.00 and height: 5.00 is: 83.73

"""
import math



class Cone:
    def __init__(self, Radius, Hight):
        self.rad = Radius
        self.hight = Hight
        #print ("Triagle sides are Initialised in super class [" +  str(side1)
    def  get_area(self):
        return ((1.0/3) * math.pi * self.rad * self.rad * self.hight)
rad = float(input('Please Enter the Radius of a Cone: '))
hght = float(input('Please Enter the Height of a Cone: '))
instance = Cone(rad,hght)
print ("Area of a cone with radius:{} and height: {} is: ".format(rad,hght) +
```

```
Please Enter the Radius of a Cone: 4
Please Enter the Height of a Cone: 5
Area of a cone with radius:4.0 and height: 5.0 is: 83.7758040957278
```

In [2]:

```
"""
2) Define a class MathOperation which implements pow(x,n) without using
python's in-built pow() method
Sample Execution:
M = MathOperation()
print(M.pow(2, 3))
8
print(M.pow(5, -3))
0.008
print(M.pow(-2, 5))
-32
print(M.pow(-5, -3))
-0.008
print(M.pow(20000,0))
1

"""


class MathOperation:
    def pow(self, x, n):
        if x==0 or x==1 or n==1:
            return x

        if x==-1:
            if n%2 ==0:
                return 1
            else:
                return -1
        if n==0:
            return 1
        if n<0:
            return 1/self.pow(x,-n)
        val = self.pow(x,n//2)
        if n%2 ==0:
            return val*val
        return val*val*x
M = MathOperation()
print(M.pow(2, 3))
print(M.pow(5, -3))
print(M.pow(-2, 5))
print(M.pow(-5, -3))
print(M.pow(20000,0))
```

```
8
0.008
-32
-0.008
1
```

In [3]:

```
"""
3) Write a python program that creates a class Base and Derived. Use inbuilt f
boolean results.(True or False)
Check:
Derived class is a subclass of Base class which will return true
Base class is a subclass of Derived class which will return false
Base class is an instance of Derived class which will return false
Derived class is an instance of Base class which will return true

"""

class Base:
    print ("Base class")

class Derived(Base):
    print ("Derived class")
b=Base()
d=Derived()
result = issubclass(Derived, Base)
print('Derived class is a subclass of Base class:', result)
result = issubclass(Base, Derived)
print('Base class is a subclass of Derived class:', result)
result = isinstance(b, Derived)
print('Base class is an instance of Derived class:', result)
result = isinstance(d, Base)
print('Derived class is an instance of Base class:', result)

```

```
Base class
Derived class
Derived class is a subclass of Base class: True
Base class is a subclass of Derived class: False
Base class is an instance of Derived class: False
Derived class is an instance of Base class: True
```

In [4]:

```
"""
4) Write a python program that creates base class Person which has two methods
def __init__(self, first, last)
def __str__(self)
Also create a derived class named Employee which uses the base class method
“def __str__(self)” using “super()” to concatenate first name with last name

"""

class Person:
    # constructor initialization
    def __init__(self,first,last):
        self.FirstName=first
        self.LastName=last
    def __str__(self):
        return "Concatination of first andlast name with super class: {} {}".f
class Employee(Person):
    def __init__(self,fname,lname):
        super().__init__(fname,lname)

emp=Employee("Santosh","Chidura")
print(emp)
```

```
Concatination of first andlast name with super class: Santosh Chidura
```

In [ ]:

```

```