Write a python program which creates a class named Cone and write a function calculate_area which calculates the area of the Cone

In [1]:
```
    class cone():

        def __init__(self,r,h):

            self.r=float(r)
            self.h=float(h)

        def area(self):
            pi=3.14
            return(pi*self.r*(self.r+(self.r**2+self.h**2)**0.5))
r=input("Enter the radius of cone\t")
h=input("Enter the height of cone\t")
t=cone(r,h)

print("Area of cone with radius {} and height {} is {}" .format(r,h,t.area()))
```

```
Enter the radius of cone        5.2
Enter the height of cone        4.5
Area of cone with radius 5.2 and height 4.5 is 197.18954136010728
```

Define a class MathOperation which implements pow(x,n) without using python's in-built pow() method

In [2]:
```
class py_solution:
    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

print(py_solution().pow(2, -3));
print(py_solution().pow(3, 5));
print(py_solution().pow(100, 0));
```

```
0.125
243
1
```

Write a python program that creates a class Base and Derived. Use inbuilt function issubclass and isinstance which gives boolean results (True or False)

In [3]:
```
class Base():
    pass   # Empty Class

class Derived(Base):
    pass   # Empty Class

# Driver Code
print(issubclass(Derived, Base))
print(issubclass(Base, Derived))

d = Derived()
b = Base()

# b is not an instance of Derived
print(isinstance(b, Derived))

# But d is an instance of Base
print(isinstance(d, Base))
```

```
True
False
False
True
```

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 wit h last name

In [4]:
```
class Person:
    def __init__(self,first,last):
        self.firstname=first
        self.lastname=last
    def __str__(self):
        return "{}""{}".format(self.firstname,self.lastname)

class Employee(Person):
    def __init__(self,first,last):
        super().__init__(first,last)
t=Employee("santosh","Kumar")
print(t)
```

```
santoshKumar
```

In [ ]: