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

In [1]:
```
import math

class Cone:
    def __init__(self,radius,height):
        self.radius=radius
        self.height=height
    def calculate_area(self):
        area=(22.00/7)*self.radius*(self.radius+math.sqrt((self.height**2)+(self.
        print("Area of a cone with radius: {:0.2f} and height: {:0.2f} is {:0.2f}

print("Please enter the radius of the cone: ",end="")
radius = int(input())
print()
print("Please enter the height of the cone: ",end="")
height = int(input())
print()
c=Cone(radius,height)
c.calculate_area()
```

```
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 130.78
```

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
```

In [2]:
```
class MathOperation:
    def pow(self,num1,num2):
        return num1**num2

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
```

3) Write a python program that creates a class Base and Derived. Use inbuilt function issubclass and isinstance which gives 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

In [3]:
```
class Base:pass
class Derived(Base):pass
print(issubclass(Derived,Base))
print(issubclass(Base,Derived))
base=Base()
derived=Derived()
print(isinstance(base,Derived))
print(isinstance(derived,Base))
```

```
True
False
False
True
```

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 wit h last name

In [4]:
```
class Person:
    def __init__(self,first,last):
        self.first=first
        self.last=last
    def __str__(self):
        print(self.first + " " + self.last)

class Employee(Person):
    def __init__(self,first,last):
        super().__init__(first,last)
        super().__str__()

e=Employee("Manoj","Mishra")
```

```
Manoj Mishra
```