**PYTHON DECORATOR EXAMPLE**

In [11]:

**def** divide(a,b):

**return**(a**/**b)

**def** smart\_div(func):

**def** validator(a,b):

**if**(b**==**0):

print("division by Zero undefined")

**else**:

**return** func(a,b)

**return** validator

div**=**smart\_div(divide)

div(10,20)

division by Zero undefined

**multiple inheritance**

In [19]:

**class** aquaticAnimal:

**def** \_\_init\_\_(self,watertype,noofgills,nooffins):

self**.**watertype**=**watertype

self**.**noofgills**=**noofgills

self**.**nooffins**=**nooffins

**def** swim(self):

print("Swimming")

**class** terrestrialAnimal:

**def** \_\_init\_\_(self,nooflegs,noofhands,livesin):

self**.**nooflegs**=**nooflegs

self**.**noofhands**=**noofhands

self**.**livesin**=**livesin

**def** walk(self):

print("walking")

**def** climb(self):

print("climb")

**class** amphibians(aquaticAnimal,terrestrialAnimal):

**def** \_\_init\_\_(self,**\***args,**\*\***kwargs):

aquaticAnimal**.**\_\_init\_\_(self,**\***args)

terrestrialAnimal**.**\_\_init\_\_(self,**\*\***kwargs)

crocodile**=**amphibians("fresh",2,2,nooflegs**=**2,noofhands**=**2,livesin**=**"land and water")

**abstarct classes**

In [21]:

**from** abc **import** ABC, abstractmethod

**class** Polygon(ABC):

@abstractmethod

**def** Area(self):

**pass**

**class** Triangle(Polygon):

**def** \_\_init\_\_(self,breadth,height):

self**.**noofsides**=**3

self**.**breadth**=**breadth

self**.**height**=**height

**def** Area(self):

area**=**(self**.**breadth**\***self**.**height)**/**2

**return** area

**Replacing the content of the file**

In [ ]:

**class** File\_Modifier:

**def** \_\_init\_\_(self,filename):

self**.**filename**=**filename

**def** getFileContent(self):

**try**:

fhandler**=**open(self**.**filename,"r")

fcontent**=**fhandler**.**read()

fhandler**.**close()

contentList**=**fcontent**.**split(' ')

**return** contentList

**except**:

print("file not found")

**def** WriteNewContent(self,newcontent):

**try**:

fhandler**=**open(self**.**filename,"w")

fhandler**.**write(newcontent)

fhandler**.**close()

**except**:

print("Write Failed")

**def** ManipulateContent(self,contentList,string,newstring):

newcontent**=**""

**if**(string **in** contentList):

index**=**contentList**.**index(string)

contentList[index]**=**newstring

**for** token **in** contentList:

newcontent**=**newcontent**+**" "**+**token

**return** newcontent

**def** modify(self,string,newstring):

contentList**=**self**.**getFileContent()

newcontent**=**self**.**ManipulateContent(contentList,string,newstring)

self**.**WriteNewContent(newcontent)

file**=**File\_Modifier("myfile.txt")

file**.**modify("placement","screening")