



```
# -*- coding: utf-8 -*-  
"""
```

```
Created on Tue Jan 3 15:16:32 2023
```

```
@author:
```

```
Groupe16 :
```

```
LABULU IBAM Danny
```

```
ONKETU ANTEMBA Beni
```

```
KABANGU MWATA Olivier
```

```
"""
```

```
"""RESOLUTION 1"""
```

```
from abc import ABCMeta, abstractmethod #importation du module abc qui gère les  
classes abstraites
```

```
from math import pi, sqrt #importation des fonctions pi et sqrt depuis math
```

```
class Geo_Form(metaclass = ABCMeta): #definition de la classe mère
```

```
    @abstractmethod #mention d'appel pour rendre la methode abstraite
```

```
    def perimetre(): #methode abstraite pour perimetre sans retour
```

```
        pass
```

```
    @abstractmethod
```

```
    def surface():
```

```
        pass
```

```
    def decris_toi(self): #methode de la description complete d'une figure
```

```
        print("Pour la figure {}\nPerimetre : {}\nSurface : {}".format(self.nomF,  
self.perimetre(), self.surface()))
```

```
"""RESOLUTION 2"""
```

```
#classe Rectangle qui herite de la classe mère Geo_Form
```

```
class Rectangle(Geo_Form):
```

```
    try: #gestion des exceptions
```

```
        def __init__(self,nomF, longueur, largeur): #initialisation de la classe
```

```
Rectangle avec nom, long, largeur
```

```
        #initialisation des variables internes de la classe
```

```
        self.nomF = nomF
```

```
        self.longueur = longueur
```

```
        self.largeur = largeur
```

```
    #methode de calcul du perimetre
```

```
    def perimetre(self):
```

```
        return 2*self.longueur + 2*self.largeur
```

```
    #methode de calcul de la surface
```

```
    def surface(self):
```

```
        return self.longueur*self.largeur
```

```
    except: #gestion des exceptions
```

```
        print("Parametres non pris en charge")
```

```
#classe Cercle qui herite de la classe mère Geometrie_Forme
```

```
class Cercle(Geo_Form):
```

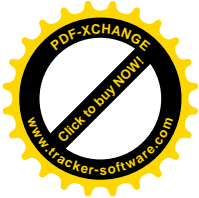
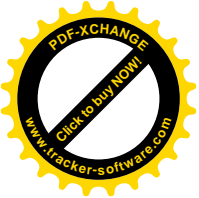
```
    try:
```

```
        def __init__(self, nomF, rayon):
```

```
            self.nomF = nomF
```

```
            self.rayon = rayon
```

```
        def perimetre(self):
```



```
        return 2*pi*sel f.rayon
    def surface(sel f):
        return pi*(sel f.rayon**2)
except:
    print("Parametres non pris en charge")

#classe Triangle qui herite de la classe mère Geo_Form
class Triangle(Geo_Form):
    try:
        def __init__(sel f, nomF, CA, CB, CC):
            sel f.nomF = nomF
            sel f.CB = CB
            sel f.CA = CA
            sel f.CC = CC
        def perimetre(sel f):
            return sel f.CB + sel f.CA + sel f.CC

        def surface(sel f):
            p = sel f.perimetre()/2
            aire = sqrt(p*(p - sel f.CA)*(p - sel f.CB)*(p - sel f.CC))
            aire = aire.real
            return aire
    except:
        print("Parametres non pris en charge")

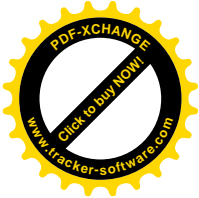
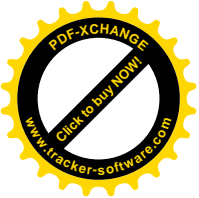
"""RESOLUTION 4"""

#classe Carre qui herite de la classe Rectangle
class Carre(Rectangle):
    try:
        def __init__(sel f, nomF, cote):
            Rectangle.__init__(sel f, nomF, cote, cote)
    except:
        print("Parametres non pris en charge ")

#classe TriangleRectangle qui herite de la classe Triangle
class TriangleRectangle(Triangle):
    try:
        def __init__(sel f, nomF, base, hauteur):
            hyp = sqrt(base**2+hauteur**2)
            Triangle.__init__(sel f, nomF, base, hauteur, hyp)
    except:
        print("Parametres non pris en charge ")

"""RESOLUTION 5"""

#classe GeoFig exploite toutes les autres classes de la Geometrie_Forme
class GeoFig():
    try:
        def __init__(sel f):
            sel f.LGeo_rep = []
        def add(sel f, fi g):
            sel f.LGeo_rep.append(fi g)
        def decris_toi(sel f):
            for L in sel f.LGeo_rep:
```



```
print("Resultat de la figure{}\nle Perimetre est de : {}m\nla  
Surface est de : {} u.s".format(L.nomF, L.perimetre(), L.surface()))  
except:  
    print("Parametres non pris en charge")
```