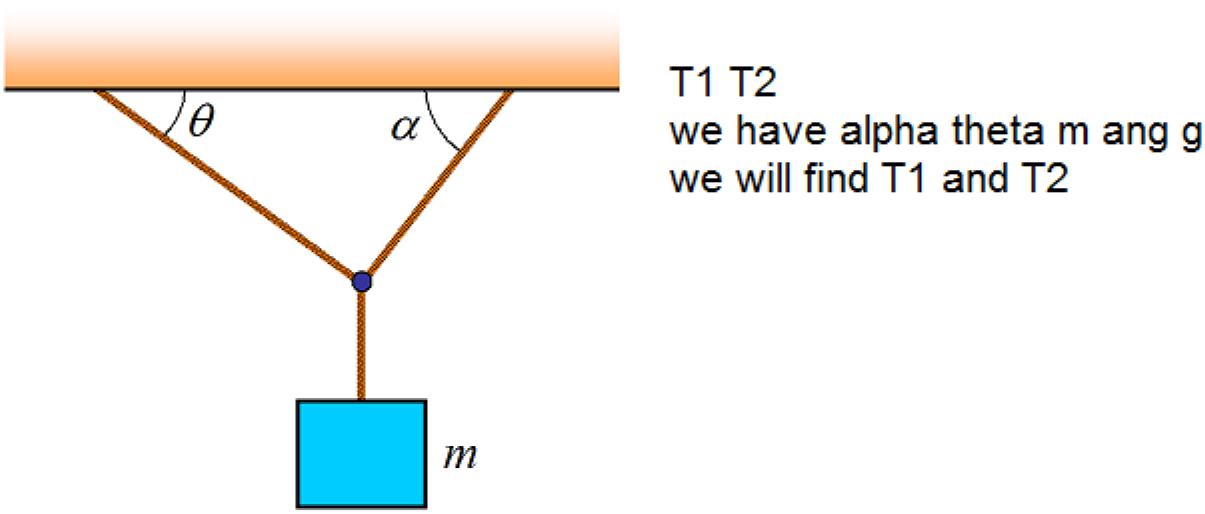
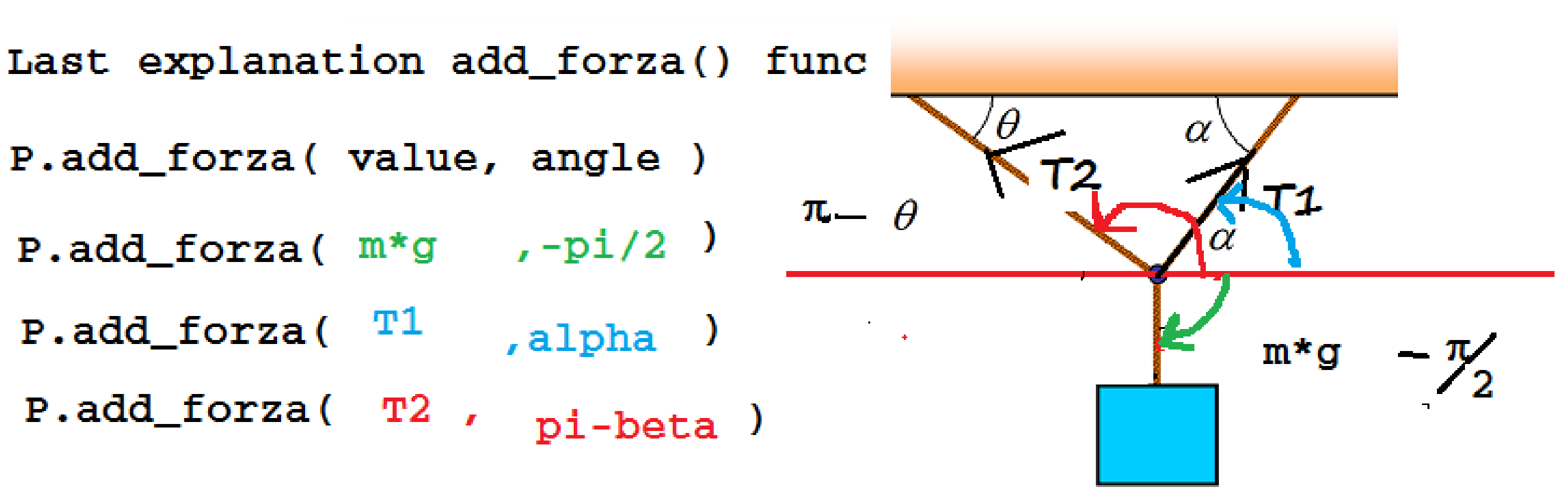
```python
from sympy import *
from polyclass import *
from libaldo_math import *
from libaldo_show import *
from physic_lib import *
from IPython.display import display, Math
init_printing()
```

Tension, angles, pulleys, etc...




```python
# creating variables....
alpha,beta,T1,T2,m,g=symbols('alpha beta T1 T2 m g')
```

```python
# creating physical object class P.. I like P for the name PUTAAAA  in spanish but you can change ..
P=mparticle(m=m,g=g)
```




```python
P.add_forza(m*g,-pi/2)
P.add_forza(T1,alpha)
```

```
P.add_forza(T1,alpha)
P.add_forza(T2,pi-beta)
```

```
# take note  default x_res() value
P.x_res()
```

$$T_1 \cos(\alpha) - T_2 \cos(\beta)$$

```
# take note  default y_res() value
P.y_res()
```

$$T_1 \sin(\alpha) + T_2 \sin(\beta) - gm$$

```
# ok we will find T1 from x_res()=0 and stores this value on internal object P
P.store_val(T1,csolve(P.x_res(),T1)) # or.. find T1 from x_res()=cero Eq and storeddddd....
```

```
# if we want to know thw value of T1 call object.value(variable)... like this
P.value(T1)
```

$$\frac{T_2 \cos(\beta)}{\cos(\alpha)}$$

```
# now work whit y_res() that we know that is cero and from now all T1 value will take from store
# see the different answer compared whit command lines before
P.y_res()
```

$$\frac{T_2 \sin(\alpha)\cos(\beta)}{\cos(\alpha)} + T_2 \sin(\beta) - gm$$

```
#now we will  find T2 from y_res()=0 and stores this value on internal object P
P.store_val(T2,csolve(P.y_res(),T2)) # or.. find T2 from y_res()=cero Eq and storeddddd....
```

```
P.value(T2)
```

$$\frac{gm\cos(\alpha)}{\sin(\alpha+\beta)}$$

---

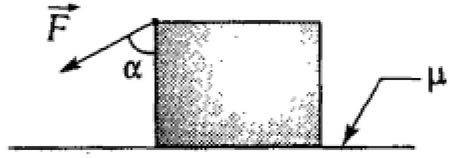
remeber class object to getting info.. ...

```
P.kvalue # kvalue mparticle store find valuessss
```

$$\left(\left[T_2,\ T_1 \right],\ \left[\frac{gm\cos(\alpha)}{\sin(\alpha+\beta)},\ \frac{T_2\cos(\beta)}{\cos(\alpha)} \right] \right)$$

Find alpha just start move

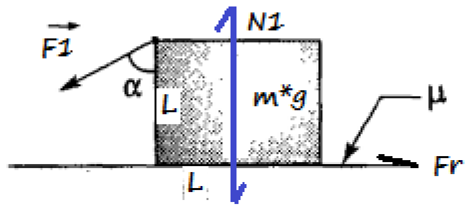
cube-->




```
# Steep 1 : Creating sympy variables
m,g,mu,N1,F1,fr,alpha,L=symbols('m g mu N1 F1 fr alpha L')
```

Find alpha just start move

cube-->




```
# Steep 2 : Creating physic object P
P=mparticle()
```

```
# adding forzas vector but including possitiones
# adding weight
P.add_forza(m*g,-pi/2,L/2,L/2)
# adding F1
P.add_forza(F1,pi+alpha,0,L)
# adding N1
P.add_forza(N1,pi/2,L/2,L/2)
# adding fr
P.add_forza(fr,0,L/2,0)
```

```
# get resultamt in x and y :
P.torque()
```

$$F_1 L \cos(\alpha) + \frac{L N_1}{2} - \frac{L g m}{2}$$

```
P.store_val(N1,0)
```

```
#
P.torque()
```

$$F_1 L \cos(\alpha) - \frac{L g m}{2}$$

```
csolve(P.torque(),F1)
```

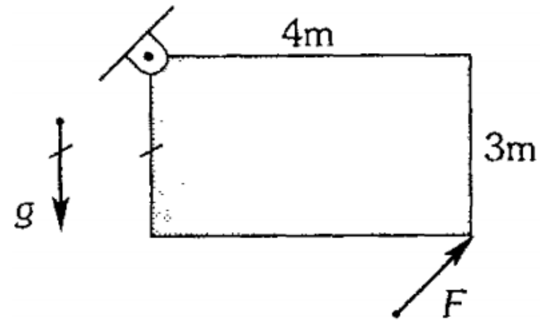
$$\frac{g m}{2 \cos(\alpha)}$$

```
# tha same answer but nice answer omly add option like....
F1=csolve(P.torque(),F1,'F1')
```

$$F1 = \frac{gm}{2\cos(\alpha)}$$

Se tiene una placa homogénea rectangular de 1 kg, que permanece en la posición mostrada. Determine el mínimo módulo de la fuerza $\vec{F}$ ($g=10$ m/s^2).

A) 1 N
B) 2 N
C) 3 N
D) 4 N
E) 5 N



Find minimum value of F to equilibrium system

we will solve this problem whit large method only for didactic explicity

```
# decrared sympy variables neccesary
F1,alpha =symbols('F1 alpha') #
```

```
# Creating physic object P
P=mparticle()
```

```
# adding forzas
P.add_forza(F1,alpha,4,-3) # F1 ,angle, pos x ,pos y > we take hinge as 0,0
P.add_forza(10,-pi/2,2,-1.5) # weight ,pi/2 > down direcc.., x,y in center mass
```

```
# getting torque value and we know that is cero
P.torque()
```

$$4F_1\sin(\alpha) + 3F_1\cos(\alpha) - 20$$

```
# Now we get F1 from torque = cero , is better designe new val in F1 .. sympy is crazy somwtimes
# using csolve to find F1 in torque Eq
F2=csolve(P.torque(),F1,'F1')
```

$$F1 = \frac{20}{4\sin(\alpha) + 3\cos(\alpha)}$$

```
# F2  will be minimun when his derivate is cero and store in new val F3
F3=diff(F2,alpha);F3
```

$$\frac{20\,(3\sin(\alpha) - 4\cos(\alpha))}{(4\sin(\alpha) + 3\cos(\alpha))^2}$$

remember some function from lib_math numer() denom().. sympy function

```
# simple call F3
F3
```

$$\frac{20\left(3\sin\left(\alpha\right)-4\cos\left(\alpha\right)\right)}{\left(4\sin\left(\alpha\right)+3\cos\left(\alpha\right)\right)^{2}}$$

```
numer(F3)
```

$$60\sin\left(\alpha\right)-80\cos\left(\alpha\right)$$

```
denom(F3)
```

$$\left(4\sin\left(\alpha\right)+3\cos\left(\alpha\right)\right)^{2}$$

```
F3=denom(F2);F3  # geting part i
```

$$4\sin\left(\alpha\right)+3\cos\left(\alpha\right)$$

```
# F2 will be minimum when numer(F3) =0
numer(F3)
```

$$4\sin\left(\alpha\right)+3\cos\left(\alpha\right)$$

```
# see the default value of -4/3 whit sympy
-4/3
```

$$-1.33333333333333$$

```
# see the default value of -4/3 whit frs(numer,denom) atb function is niceee....
frs(-4,3)
```

$$-\frac{4}{3}$$

```
# remeber that sE(['tex',math valu,'tex2',.....]) is used instead of Latex that I hate it...
sE(['if ',numer(F3),'= 0 then ', tan(alpha),'=',frs(-4,3),' also',sin(alpha),'=',frs(4,5),'and ',cos(al
```

$$if\ 4\sin\left(\alpha\right)+3\cos\left(\alpha\right)\ =\ 0\ then\ \tan\left(\alpha\right)\ =\ -\frac{4}{3}\ also\ \sin\left(\alpha\right)\ =\ \frac{4}{5}\ and\ \cos\left(\alpha\right)\ =\ -\frac{3}{5}$$

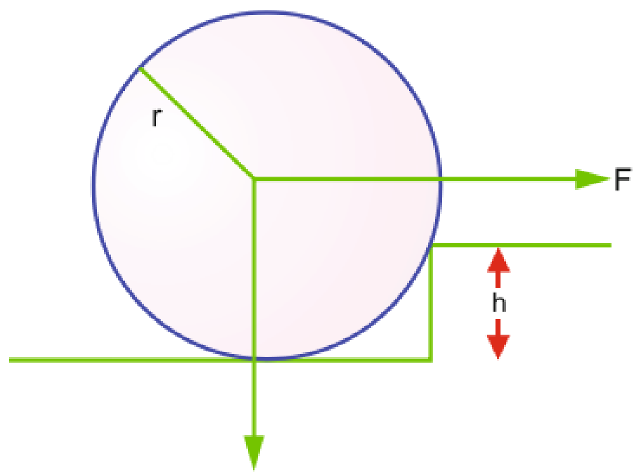
```
# calling F2 that is the final value that we need
F2
```

$$\frac{20}{4\sin(\alpha) + 3\cos(\alpha)}$$

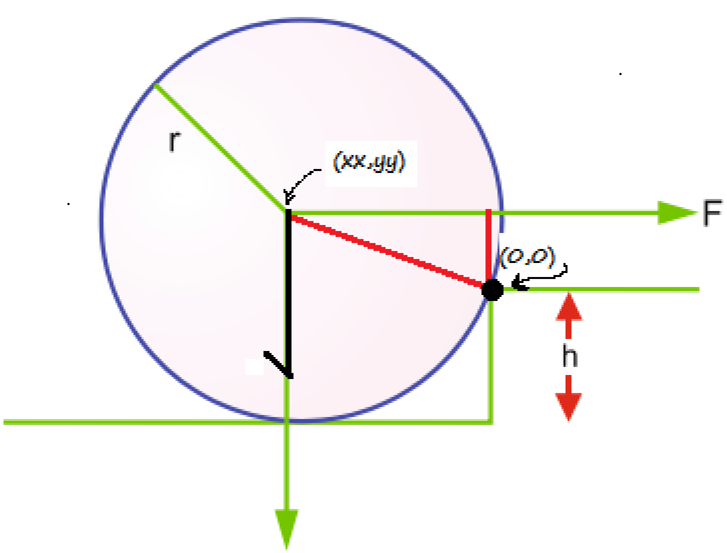
```
# now we will store F1,sin and cos value to use internal

P.store_val(sin(alpha),frs(4,5))
P.store_val(cos(alpha),frs(-3,5))
P.store_val(F1,F2)


P.value(F1)
```

$$\frac{100}{7}$$

1.42 A wheel of radius r and weight W is to be raised over an obstacle of height h by a horizontal force F applied to the centre. Find the minimum value of F




```
# decrared sympy variables neccesary
W,F1,r,h =symbols('W F1 r h ')
```




```
# Creating  object
```

```
# Creating  object
P2=mparticle()
```

```
# rpow() and kpow from atb math lib because is more clear whit exponet proccedure.. example
rpow(r)
```

$$\sqrt{r}$$

```
kpow(r,2)
```

$$r^{2}$$

```
kpow(r,frs(2,3)) # niceeeeeeeeeeeeee.......
```

$$r^{\frac{2}{3}}$$

```
rpow(r-h,2)
```

$$\sqrt{-h + r}$$

```
# ok ,, adding value o xx and yy respect (0,0)
xx,yy = symbols('xx yy',positive=True)
xx=r-h
yy=rpow(kpow(r,2)-kpow(r-h,2))
```

```
P2.add_forza(F1,0,xx,yy)
P2.add_forza(W,-pi/2,xx,yy)
```

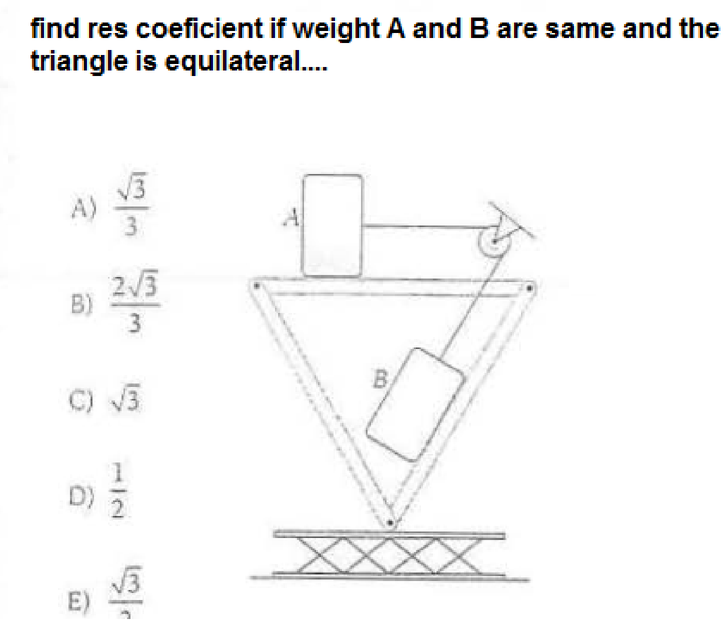
```
P2.torque()
```

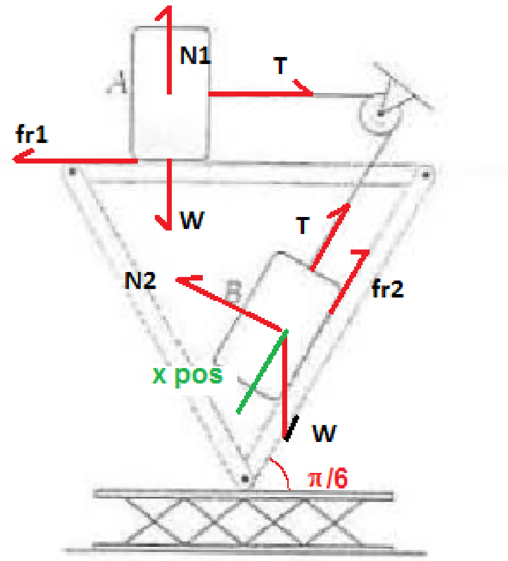
$$- F_{1} \sqrt{r^{2} - \left(- h + r\right)^{2}} - W \left(- h + r\right)$$

```
F1=csolve(P2.torque(),F1,'F')
```

$$F = \frac{W \left(h - r\right)}{\sqrt{h \left(- h + 2 r\right)}}$$

Please, take note that from now we assume that you know physics, and if not, is better that you return to books ... because the idea is know how we use this library, and not teach physics, Ok Carajooooo.






```
# Steep 1 : Creating symbols
W,N1,N2,T,fr1,fr2,mu=symbols('W N1 N2 T fr1 fr2 mu')


# Steep 2 : Creating A mparticle() Physic class and adding forces
A=mparticle()
A.add_forza(T,0)
A.store_val(fr1,W*mu)
A.add_forza(fr1,pi)


# Steep 2 : get x_res and know equal cero...
A.x_res()
```

$$T - W\mu$$

```
# Now get T in other function variable


T1=csolve(A.x_res(),T,'T') # why not T istead of T1.. because mayby sympy put crazy..
```

$$T = W\mu$$

```
# work whit B
B=mparticle()
B.add_forza(T1,pi) # we put T1 instwaed T because we finding before.. see up
B.add_forza(W,-pi/6)
B.add_forza(N2,pi/2) # before add fr2 we store value N2*mu and then use this val
B.store_val(fr2,N2*mu)
B.add_forza(fr2,pi)


# Now we will try know N2 value because Y_res equal to cero
B.y_res()
# ok simple N2=W/2 later store this value
```

```
# ok simple N2 W/2 later store this value
```

$$N_2 - \frac{W}{2}$$

```
# first see the value of B.x_res() before...
B.x_res()
```

$$-N_2\mu - W\mu + \frac{\sqrt{3}W}{2}$$

```
# ok what happend if now change N2 whit store_val and call again y_res..
B.store_val(N2,W/2)
B.x_res()
```

$$-\frac{3W\mu}{2} + \frac{\sqrt{3}W}{2}$$

```
# Now how we know that y_res equal to cero aply csolve and find mu
mu=csolve(B.x_res(),mu,'m_u')
```

$$m_u = \frac{\sqrt{3}}{3}$$

```
### Method 2 aplied polyclass and Mpolyclass
A=mparticle()
A.add_forza(T,0)
A.store_val(fr1,W*mu)
A.add_forza(fr1,pi)
B=mparticle()
B.add_forza(T,pi) # we put T1 instwaed T because we finding before.. see up
B.add_forza(W,-pi/6)
B.add_forza(N2,pi/2) # before add fr2 we store value N2*mu and then use this val
B.store_val(fr2,N2*mu)
B.add_forza(fr2,pi)


e1=polyclass(fr1,W*mu)
e2=polyclass(A.x_res(),0)
e3=polyclass(B.y_res(),0)
e4=polyclass(fr2,N2*mu)
e5=polyclass(B.x_res(),0)
vec=[e1,e2,e3,e4,e5]
P=Mpolyclass(vec);P.showQ()
```

$$\begin{aligned}
fr_1 &= W\mu \\
T - W\mu &= 0 \\
N_2 - \frac{W}{2} &= 0 \\
fr_2 &= N_2\mu \\
-N_2\mu - T + \frac{\sqrt{3}W}{2} &= 0
\end{aligned}$$

FUNCTION in Mpolyclass..... try_solve()

$$fr_1 = W\mu$$

$$T - W\mu = 0$$

$$N_2 - \frac{W}{2} = 0$$

$$fr_2 = N_2\mu$$

$$-N_2\mu - T + \frac{\sqrt{3}W}{2} = 0$$

function try_solve(symbols[], num_eq[])

1 is ok because fr1 alone ,remeber that it is Eq num 0

2 solve T in Eq 1

3 solve N2 in Eq 2

4 fr2 is ok because alone

5 final solve mu in 4

```
P.try_solve([T,N2,mu],[1,2,4])
```

$$fr_1 = \frac{\sqrt{3}W}{3}$$

$$T = \frac{\sqrt{3}W}{3}$$

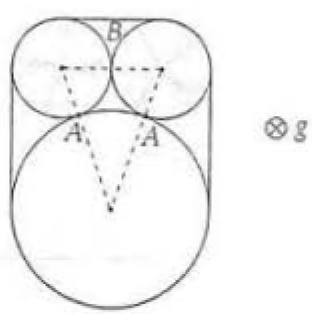
$$N_2 = \frac{W}{2}$$

$$fr_2 = \frac{\sqrt{3}W}{6}$$

$$\mu = \frac{\sqrt{3}}{3}$$

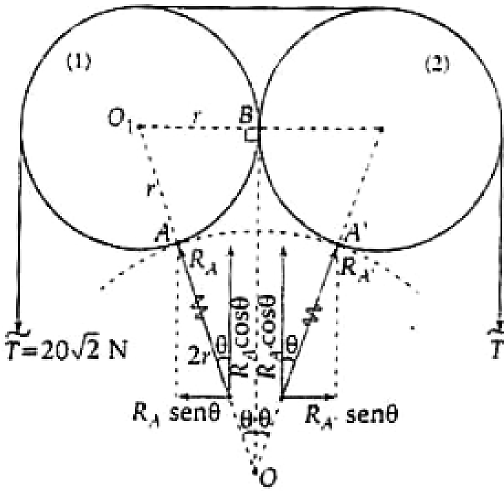
Remember .. if you think that the procedure is long .. is due tutorial teach.....

On a horizontal table are placed the 3 rollers, of which one has twice the diameter than other one and the rope that supports a modulus tension $20\sqrt{2}$ N. Determine the moduli of the reactions in A and B contact...

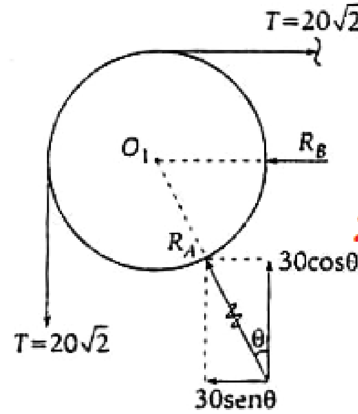


A) $R_A = 50$ N; $R_B = 19$ N
B) $R_A = 30$ N; $R_B = 18{,}2$ N
C) $R_A = 33{,}2$ N; $R_B = 15$ N
D) $R_A = 30$ N; $R_B = 20$ N
E) $R_A = 50$ N; $R_B = 18{,}2$ N

```
# creating symbols
Ra,T,alpha,r=symbols('Ra T alpha r',positive=True) #mayby sqrt appear,better make sure ..
```

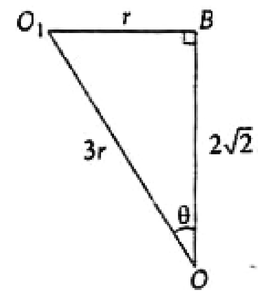


1



2

- Del triángulo O_1BO



3

Please, take note that from now we assume that you know physics, and if not, is better that you return to books ... because the idea is know how we use this library, and not teach physics, Ok Carajooooo.

```
T=20*rpow(2,2)


# First  Eq usin polyclass in last image part 1,2
e1=polyclass(Ra*cos(alpha)+Ra*cos(alpha),2*T)
# then  Eq usin polyclass in last image part 3
e2=polyclass(cos(alpha),2*rpow(2)*r/(3*r))
P=Mpolyclass([e1,e2]);P.showQ()
```

$$
\begin{aligned}
2Ra\cos(\alpha) &= 40\sqrt{2} \\
\cos(\alpha) &= \frac{2\sqrt{2}}{3}
\end{aligned}
$$

```
# apply simple and super neeeerddddd intuition   order to solve Mpolyclass...

P.try_solve([cos(alpha),Ra],[1,0])
```

$$
\begin{aligned}
Ra &= 30 \\
\cos(\alpha) &= \frac{2\sqrt{2}}{3}
\end{aligned}
$$