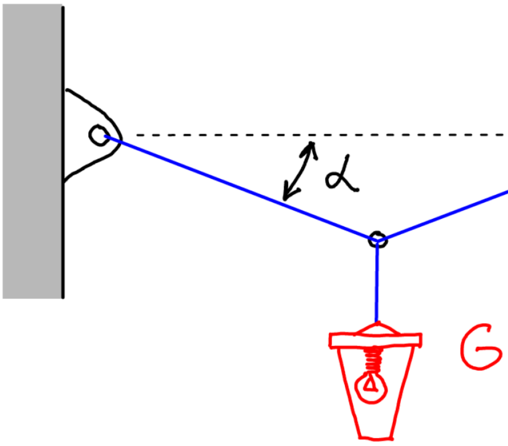
# 1 Példa 1.1

Egy $G = 700\,\mathrm{N}$ súlyú lámpatestet szeretnénk felfüggeszteni a merevnek tekintett falak közé egy $\O d = 3\,\mathrm{mm}$ átmérőjű acélhuzallal, melyre húzás esetén a megengedett feszültség $\sigma_{\mathrm{meg}} = 285\,\mathrm{MPa}$. Határozzuk meg, hogy mekkora lehet minimálisan a huzal vízszintessel bezárt szöge annak érdekében, hogy a huzalban ébredő normálfeszültség a megengedett érték alatt maradjon.




# 2 Megoldás

A megoldás során szimbolikus számításokat fogunk végezni (azaz a konkrét értékeket csak a végén helyettesítjük be, előtte a képleteket írjuk fel és rendezzük át). Ehhez szükségünk van a `sympy` modulra.

In [1]:

```
import sympy as sp #betöltjük a sympy modul összes függvényét, és sp-ként hivat
# ami függvényt a sympyból használunk azt sp.függvény formában hívjuk meg
```

executed in 548ms, finished 09:30:08 2020-02-18

Definiálnunk kell a később haszált szimbólumokat. Az átláthatóság kedvéért mi most a kód legelején definiáljuk őket.

A szintaktika: `valtozonev = sp.symbols("kiirt_nev")` . A programkódban a szimbólumra a `valtozonev` -vel hivatkozunk. A `"kiirt_nev"` (a " kell az elejére és végére) az a karaktersor, amit kiír a program, mint a szimbólum neve, amikor ki akarunk íratni egy végeredményt. A `valtozonev` és `"kiirt_nev"` bármi lehet, de célszerű, hogy megegyezzenek.

In [2]:

```
G = sp.symbols("G")
d = sp.symbols("d")
σ_meg = sp.symbols("σ_meg") # sigma: \sigma + TAB
α = sp.symbols("α") # alpha: \alpha + TAB
# a görög szimbólumok a "LaTeX jelölésük" + TAB segítségével hívhatók meg
```

executed in 5ms, finished 09:30:08 2020-02-18

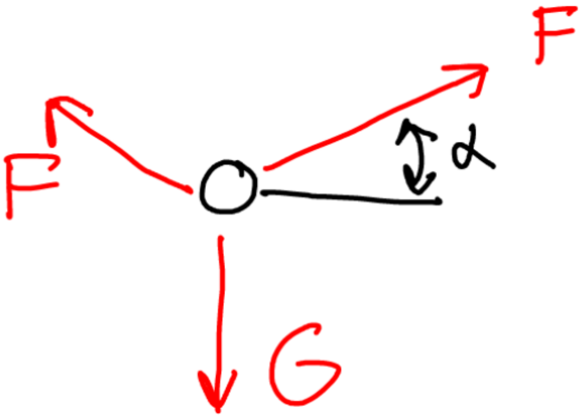
A feladat megad néhány konkrét értéket, amit később behelyettesíthetünk. Ezeket az átláthatóság kedvéért itt, a feladat elején definiáljuk.

In [3]:

```
G_adat = 700 #N
d_adat = 3 #mm
σ_meg_adat = 285 #MPa
```

executed in 18ms, finished 09:30:08 2020-02-18

Írjuk fel a szükséges összefüggéseket a szabadtest ábra alapján.




$$2F\sin\alpha = G.$$

A fenti egyenletből kell kifejeznünk $F$-et. Ekkor az alábbi összefüggést kapjuk:

$$F = \frac{G}{2\sin\alpha}.$$

(Természetesen az átrendezést megcsinálhatnánk a `sympy` segítségével - akár sokkal bonyolultabb összefüggés esetén is -, ezt a félév későbbi részében mutatjuk be.)

Írjuk be a fenti összefüggést $F$-re a programkódba is! A szinusz függvény a `sympy` modul része, ezért a $\sin\alpha$-t `sp.sin(α)` -ként programozhatjuk be. Fontos megjegyezni, hogy Pythonban a szorzásjeleket ki **kell** tenni. Nem írhatunk `2 sp.sin(α)` -t a kódban, kell közé a `*` : `2*sp.sin(α)` .

In [4]:

```
F = G/(2*sp.sin(α)) # az összefüggést eltároljuk az 'F' nevű változóban
F # megnézzük, hogy mi van az 'F' változóban
  #(ennek az utasításnak a hatására csak akkor kapunk kimenetet, ha ez a cella
```

Out[4]:

$$\frac{G}{2\sin(\alpha)}$$

Tudjuk, hogy a huzal keresztmetszete:

$$A = \frac{d^2\pi}{4}.$$

Programozzuk be ezt is, az $F$-hez hasonló módon! Figyeljünk arra, hogy **Pythonban a hatványozás jele a** `**` . A $\pi$ ismét a `sympy` modul része, ezért `sp.pi` -ként adhatjuk meg.

In [5]:

```
A = d**2*sp.pi/4 # az összefüggést eltároljuk az 'A' nevű változóban
A # megnézzük, hogy mi van az 'A' változóban (ez csak akkor működik, ha ez a ce
```

Out[5]:

$$\frac{\pi d^2}{4}$$

A normálerő és feszültség között az alábbi összefüggés áll fent:

$$\sigma = \frac{F}{A}.$$

Ebbe ha behelyettesítjük az $F$-re kapott összefüggést, akkor a következőt kaphatjuk:

$$\frac{G}{2\sin\alpha} = A\sigma_{meg}.$$

Ebből a keresett $\alpha$ szög:

$$\alpha = \arcsin\frac{G}{2A\sigma_{meg}}.$$

Az $\arcsin$ a `sympy` modul része, `sp.asin()` -ként érjük el.

In [6]:

```
α_megoldas = sp.asin(G/(2*A*σ_meg))
α_megoldas # nézzük meg, hogy mit kaptunk
```

Out[6]:

$$\operatorname{asin}\left(\frac{2G}{\pi d^2 \sigma_{meg}}\right)$$

Láthatjuk, hogy visszakaptunk az összefüggést, nem kaptunk numerikus eredményt, hiszen nem helyettesítettük be az adatokat.

Egy összefüggésbe behelyettesíteni az `osszefugges.subs(regi,uj)` szintaktikával tudunk, ahol `regi` az a változó, amelyet az `uj` -ra szeretnénk kicserélni. Ez esetünkben egy szimbolikus változó kicserélése egy numerikus értéket tároló változóra. Egy sorban több értéket is behelyettesíthetünk több `.subs()` utasítással.

In [7]:

```
α_behely = α_megoldas.subs(G,G_adat).subs(d,d_adat).subs(σ_meg,σ_meg_adat)
```

Ennél általában olvasatóbb egy alternatív szintaktika. Ebben a behelyettesítendő `(regi,uj)` párokat egy listában tároljuk el, majd a listát írjuk bele a `.subs()` -ba. Ez főként akkor előnyös, ha több kifejezésbe is be kell helyettesítenünk, hiszen a listát csak egyszer kell létrehozni.

In [8]:

```
adatok = [(G,G_adat),(d,d_adat),(σ_meg,σ_meg_adat)]
α_behely = α_megoldas.subs(adatok)
α_behely # megnézzük, hogy mit kapunk a behelyettesítés után
```

Out[8]:

$$\operatorname{asin}\left(\frac{280}{513\pi}\right)$$

Láthatjuk, hogy most sem kaptunk számszerű eredményt. Ennek az oka, hogy a `sympy` egy szimbolikus csomag, ezért magától csak azokat az átalakításokat végzi el, amik matematikailag teljesen ekvivalensek. Azaz magától nem ad numerikus eredményt, hiszen a kerekítések és véges számú tizdesjegy miatt óhatatlanul sérülne a matematikai egzaktság.

Ilyenkor ki kell adni egy utasítást, hogy a kedvünkért mégis adjon numerikus eredményt. Az `.evalf(N)` utasítással értékelhetünk ki egy kifejezést N értékesjegyre. Az `.evalf()` nem olyan "függvény", mint amiket korábban használtunk, ezért a szintaktikája `kifejezes.evalf()` azaz a kiértékelendő dolog után írjuk.

*Kíváncsiaknak: ez azért van így, mert az `.evalf()` önmagában nem létezik, nincsen `sp.evalf()` függvény. Az `evalf()` csak bizonyos típusokhoz - pontosabban osztályokhoz - kötötten létezik, mint azok saját magukon alkalmazható "függvénye". Ezt úgy precízen mondjuk, hogy az `.evalf()` ezeknek az osztályoknak metódusa. Mivel a Szilárdságtan tárgyhoz készült segédleteknek nem a programozási szaknyelv megtanítása a fő célja, ezért mi gyakran nem fogunk különbséget tenni a szövegben függvény és metódus között.*

Értékeljük ki a kapott kifejezést 5 értékesjegyre! Az eredményt radiánban kapjuk, a `sympy` mindig radiánban számol.

In [9]:

```
α_behely.evalf(5)
```

Out[9]:

$$0.17462$$

A legegyszerűbben úgy kaphatunk ebből fokot, ha az ismert módon megszorozzuk $\frac{180}{\pi}$-vel. Ezután írjuk ki az eredmény 5 értékesjegyre, fokban.

In [10]:

```
α_fok = α_behely*180/sp.pi # eltároljuk a beszorzott értéket az α_fok változóban
α_fok.evalf(5) # kiírjuk az eredményt 5 értékesjegyre °-ban
```

Out[10]:

$$10.005$$

Megjegyzés: az eredményt elvileg akárhány értékesjegyre kiírhatnánk, ennek viszont sok értelme nincs a gyakrolatban. A különböző közelítések használata jó eséllyel nagyobb hibát okoz, mint hogy csak 5 értékesjegyre értékelünk ki. A jelen feldatban például elhanyagoltuk az acélhuzal saját súlyát.