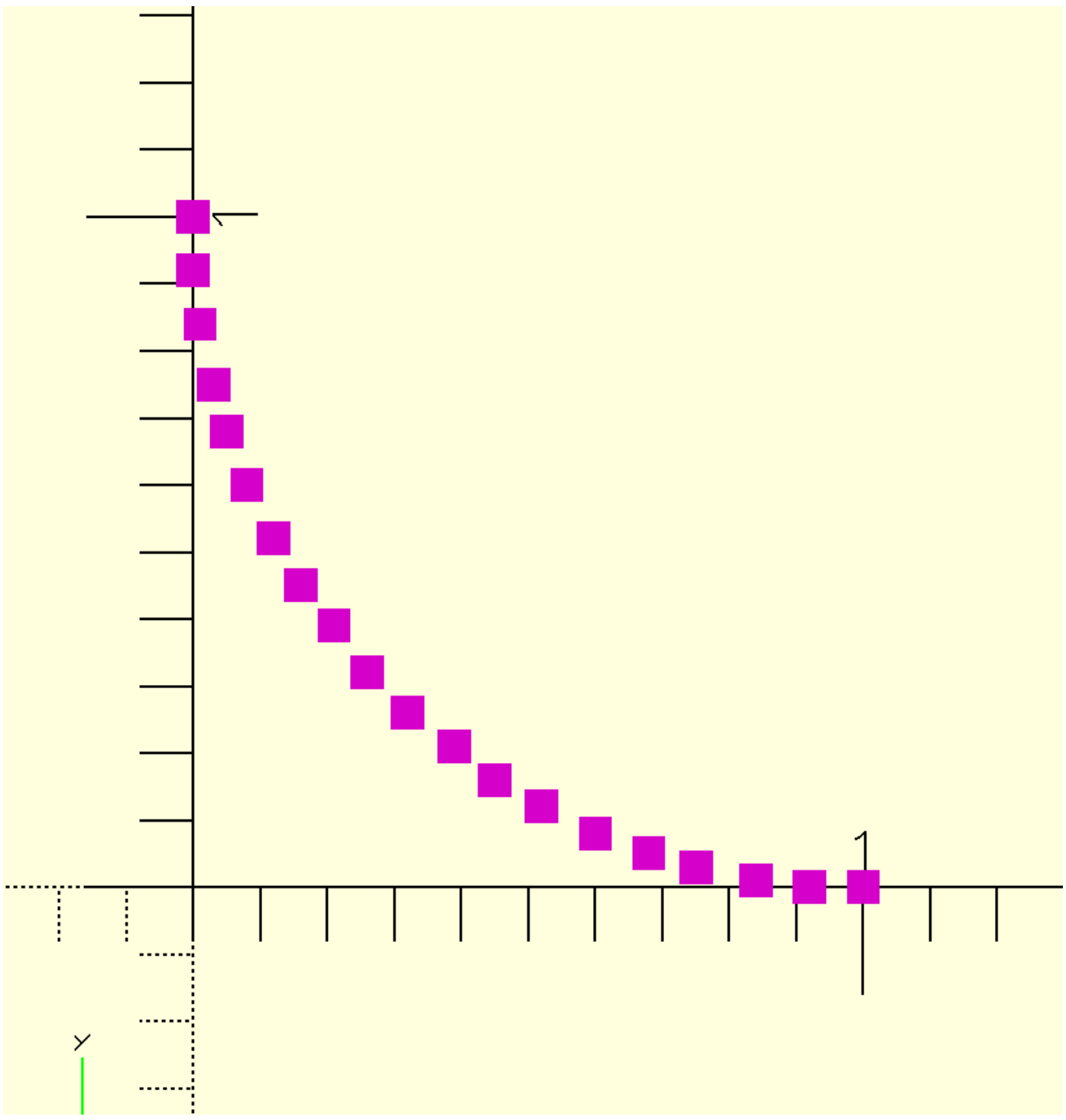
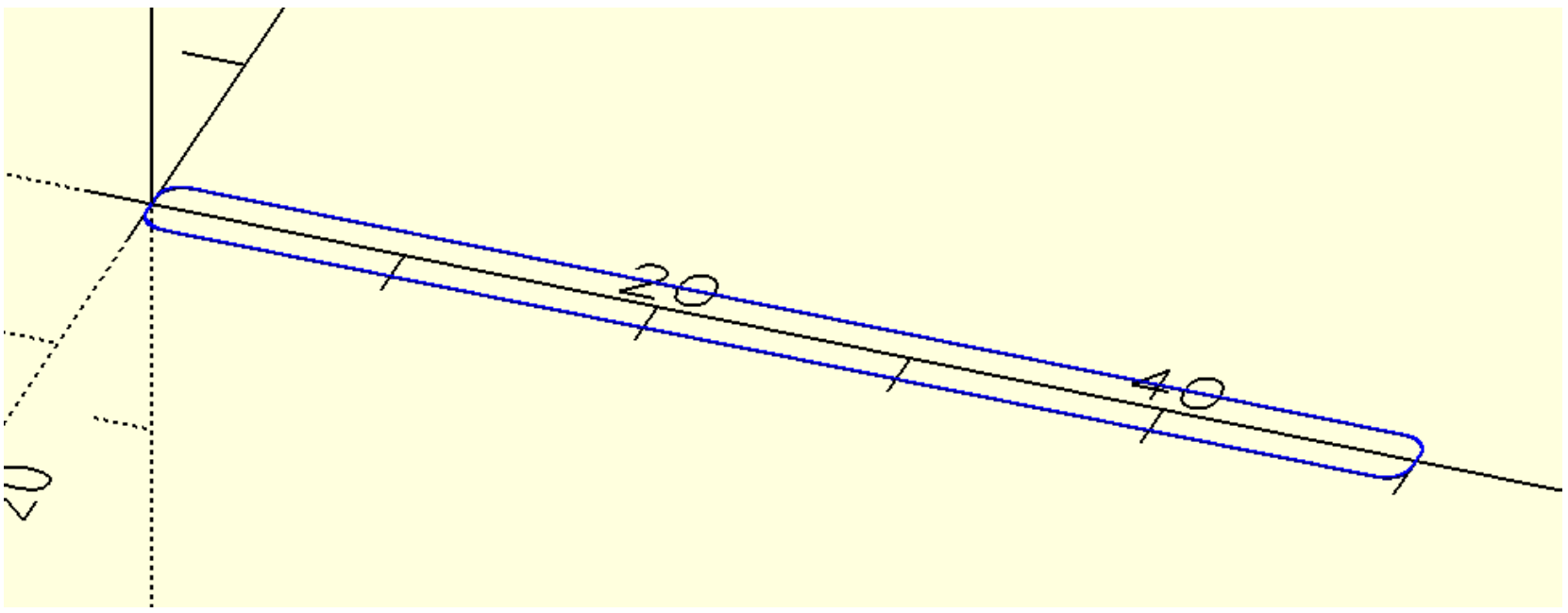
In [1]:
```
%load_ext autoreload
%autoreload 2
from openscad3 import *
```

In [18]:
```
# Step1: Create an Arc of the fillet radius required
r=1
a=l_(round([[r+r*cos(d2r(i)),r+r*sin(d2r(i))] for i in linspace(180,270,20)],2))
with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("magenta")points({a},.05);

    ''')
```

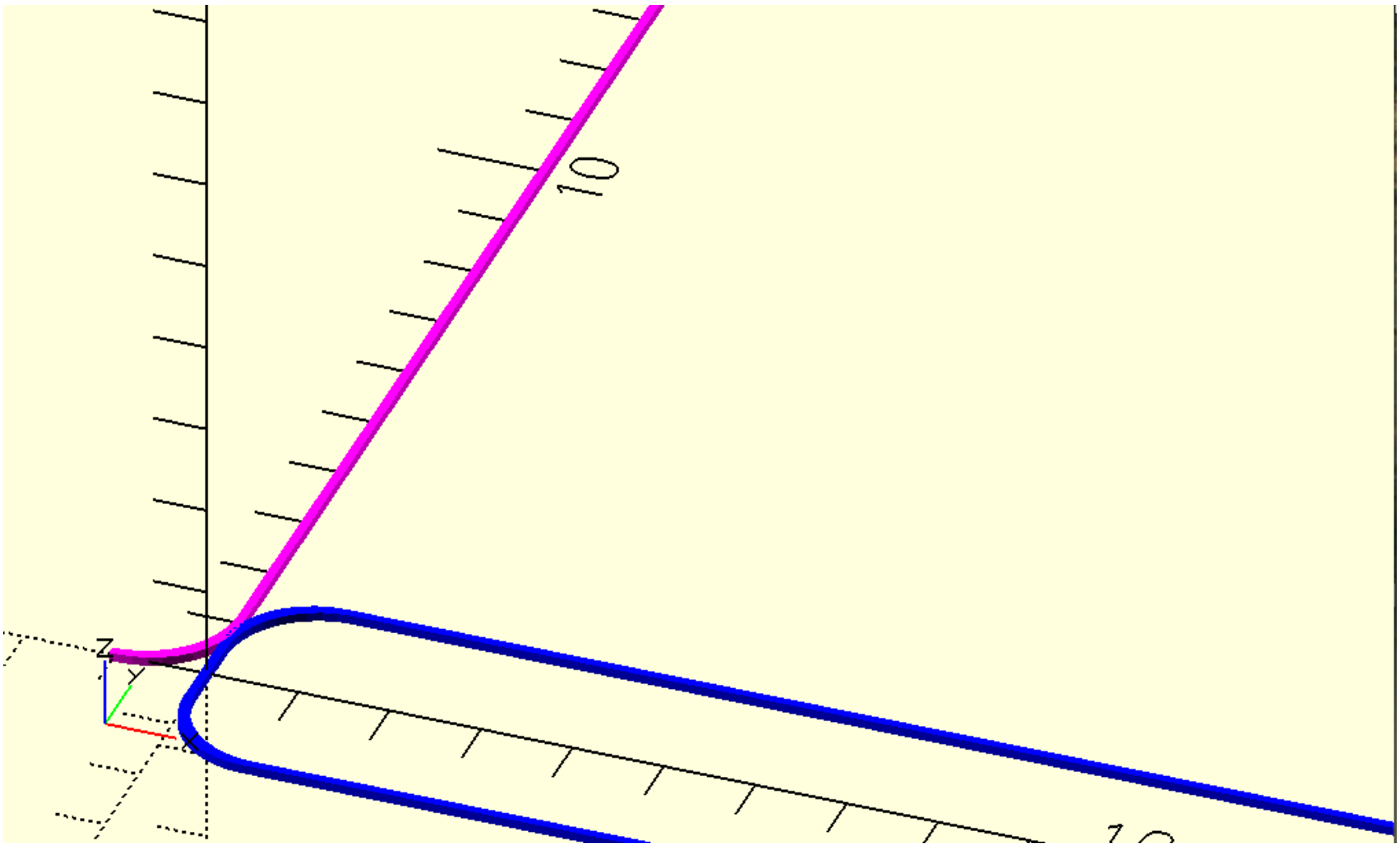


In [20]:
```
# To create a solid create a 2d section of the solid
sec=corner_radius_with_turtle([[0,-1.5,1],[50,0,1],[0,3,1],[-50,0,1]],10)

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3dc({sec},.1,1);

    ''')
```

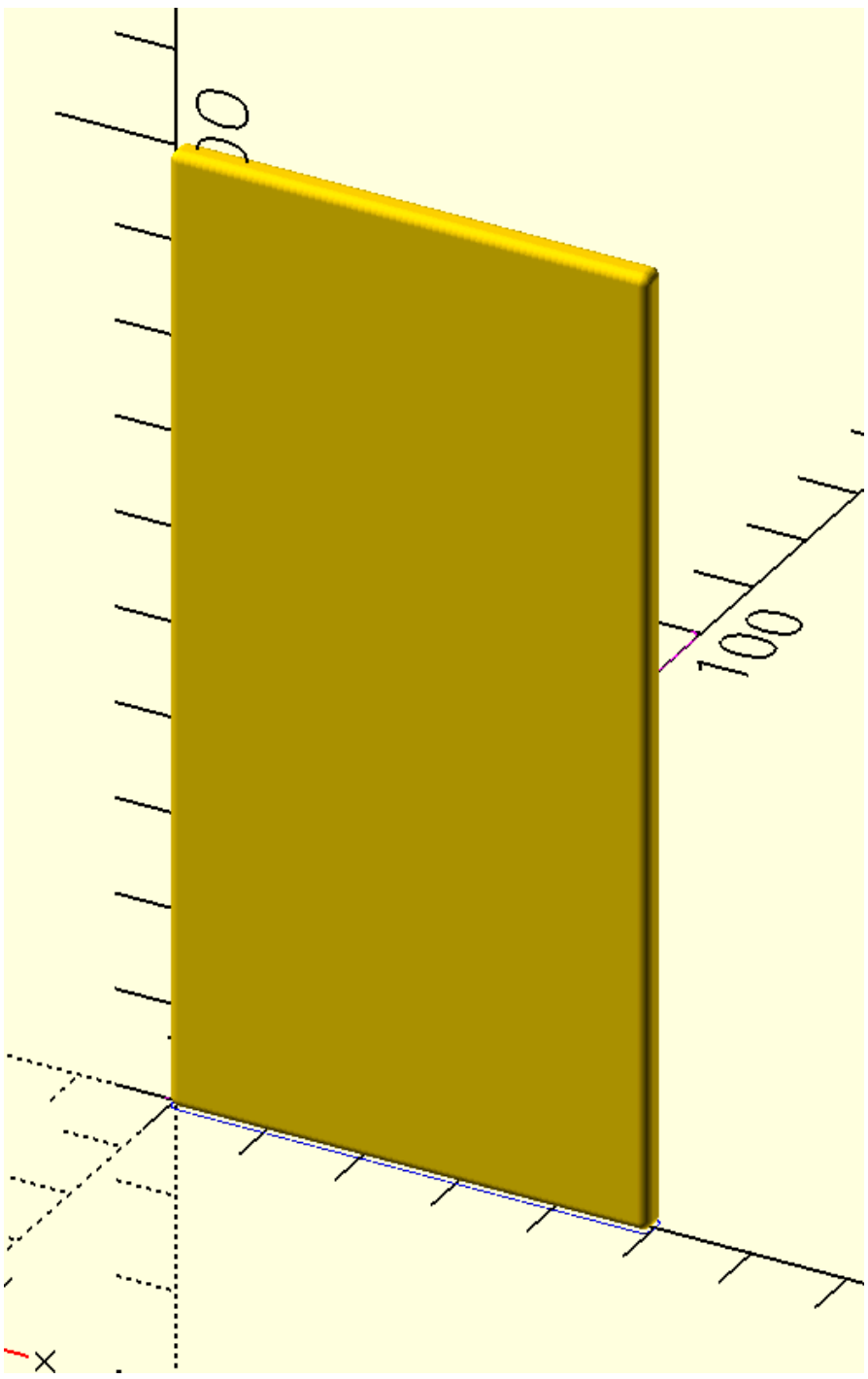


In [21]:
```
# Create a 2d path to shape the above section 'sec'

path=corner_radius_with_turtle([[-1,0],[1,0,1],[0,100,1],[-1,0]],10)

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3dc({sec},.1,1);
```

```
color("magenta")p_line3d({path},.1,1);
    ''')
```

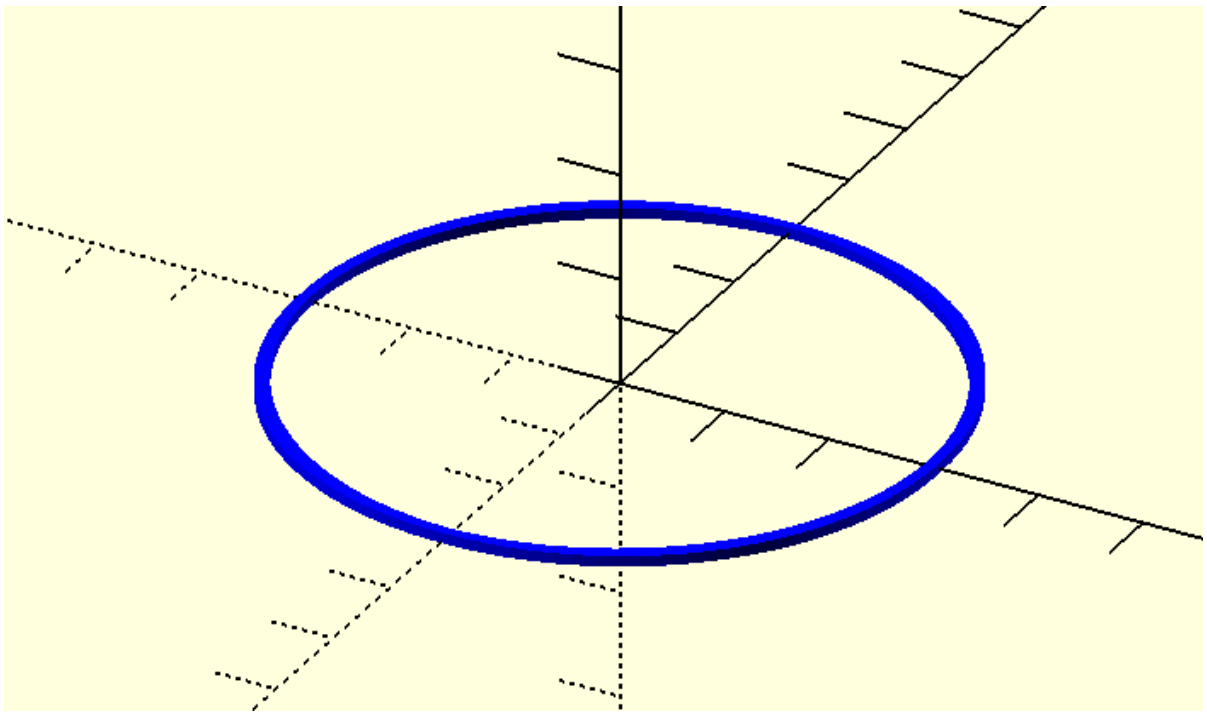


In [22]:
```
# Use function prism to create a solid 'sol1'

sol1=prism(sec,path)

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3dc({sec},.1,1);
color("magenta")p_line3d({path},.1,1);
{swp(sol1)}
    ''')
```

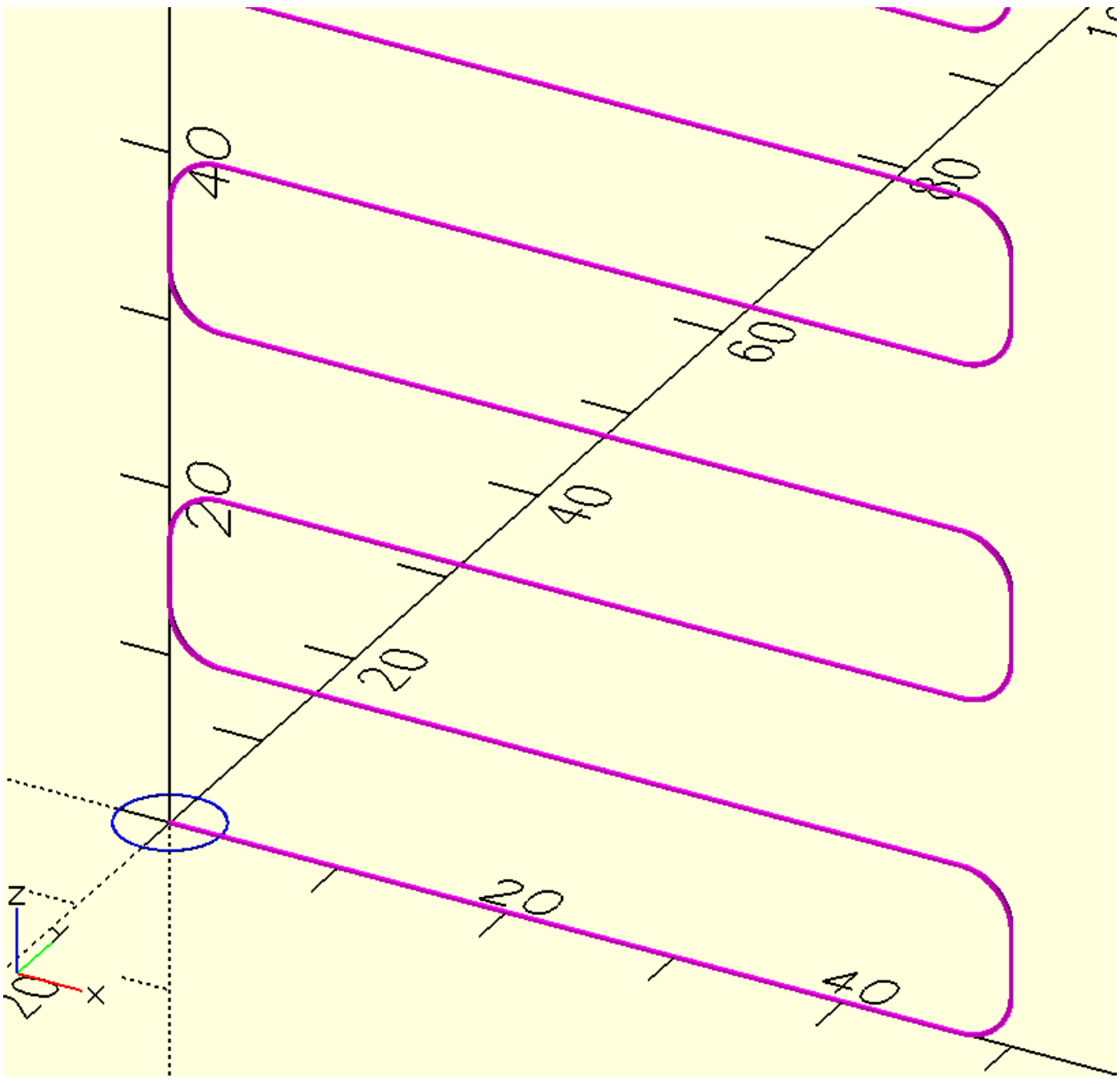


In [23]:
```
# Create another 2D section in this case a circle

sec1=circle(3,s=100)

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
```

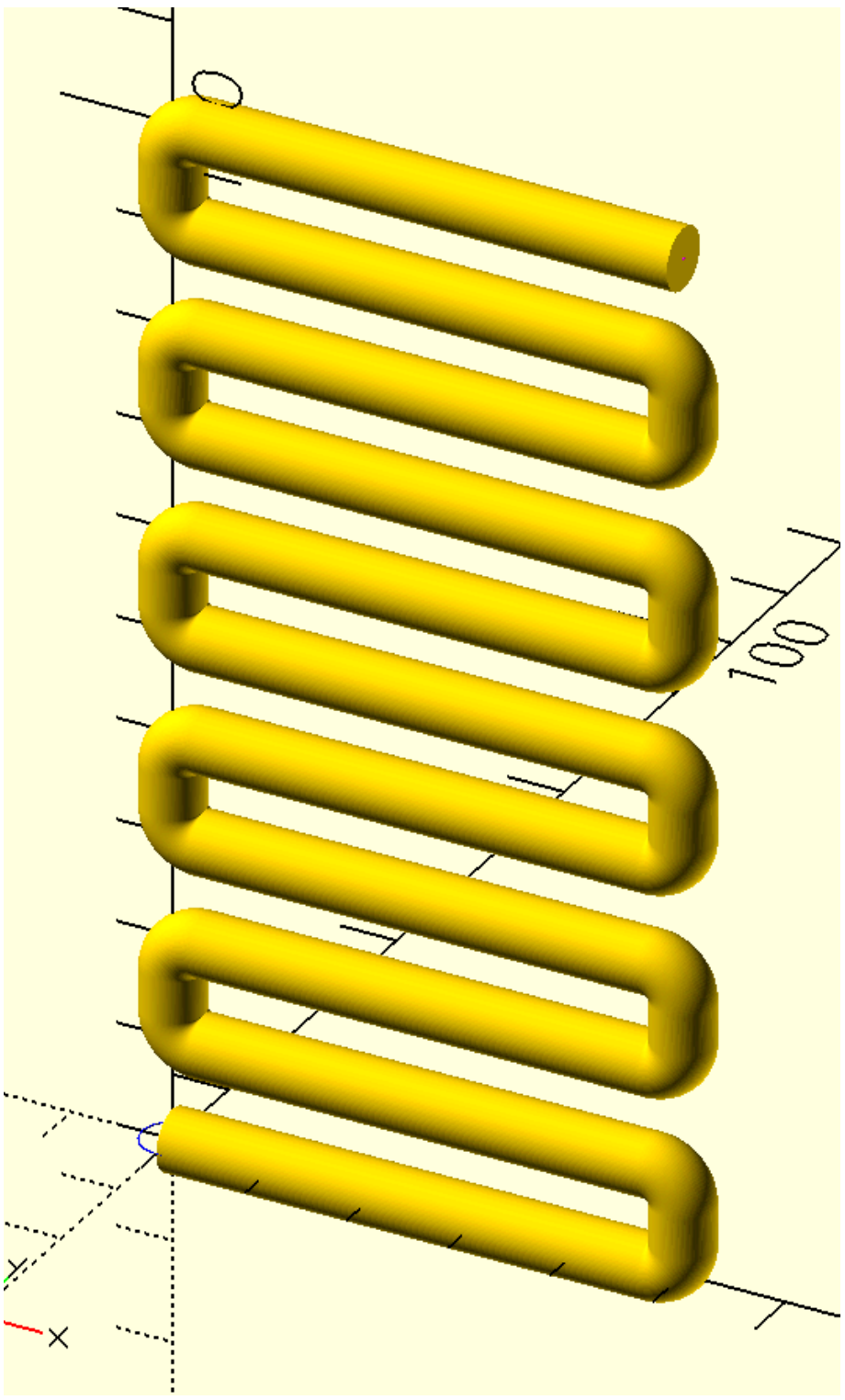
```
color("blue")p_line3dc({sec1},.1,1);

    ''')
```



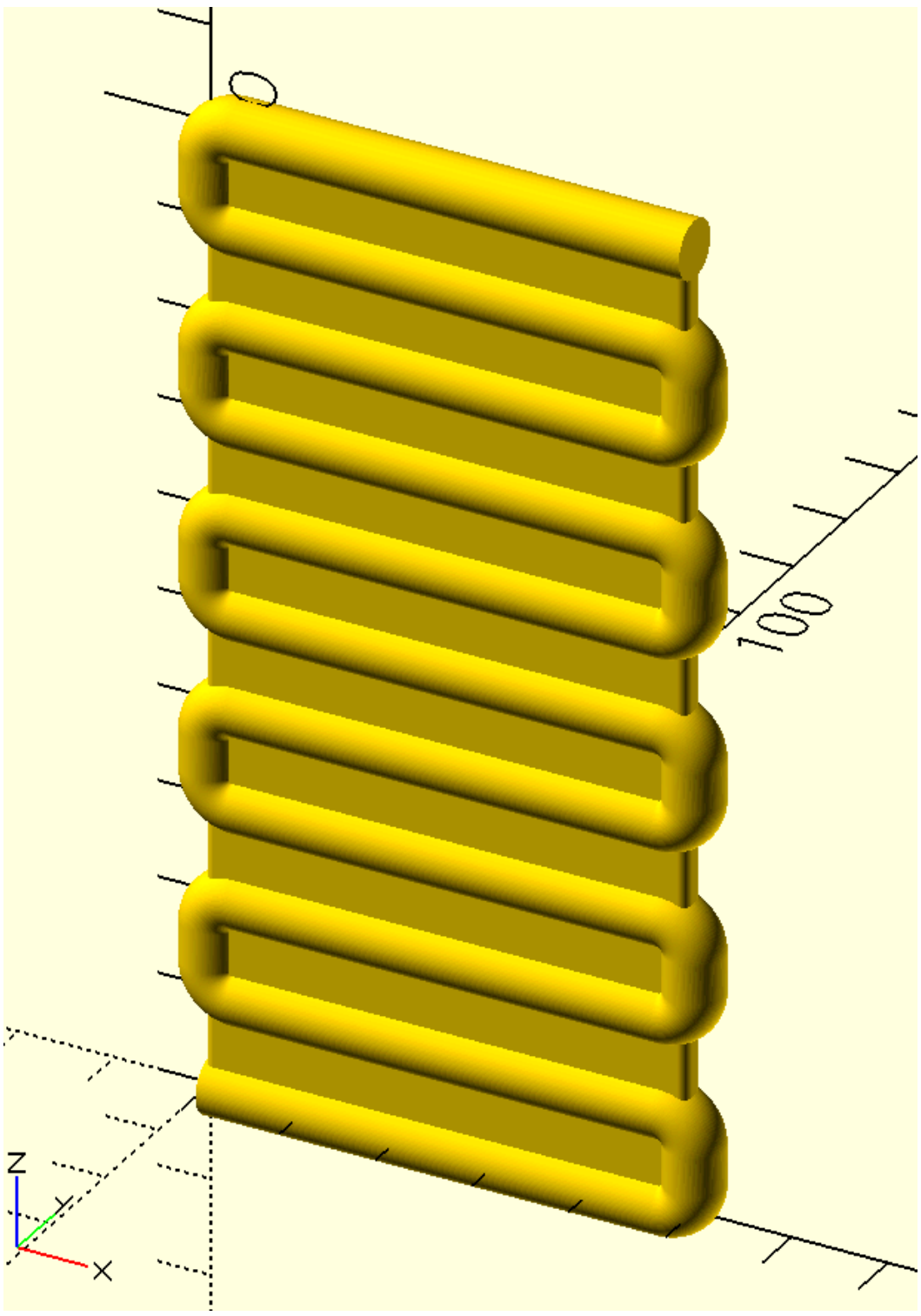
In [24]:
```
# Create a 3D path to extrude the above section 'sec1'

path1=corner_radius3d_with_turtle([[0,0,0],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                   [0,0,10,3],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                   [0,0,10,3],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                   [0,0,10,3],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                     [0,0,10,3],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                    [0,0,10,3],[50,0,0]
                                  ],30)

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3dc({sec1},.1,1);
color("magenta")p_line3d({path1},.2,1);
    ''')
```



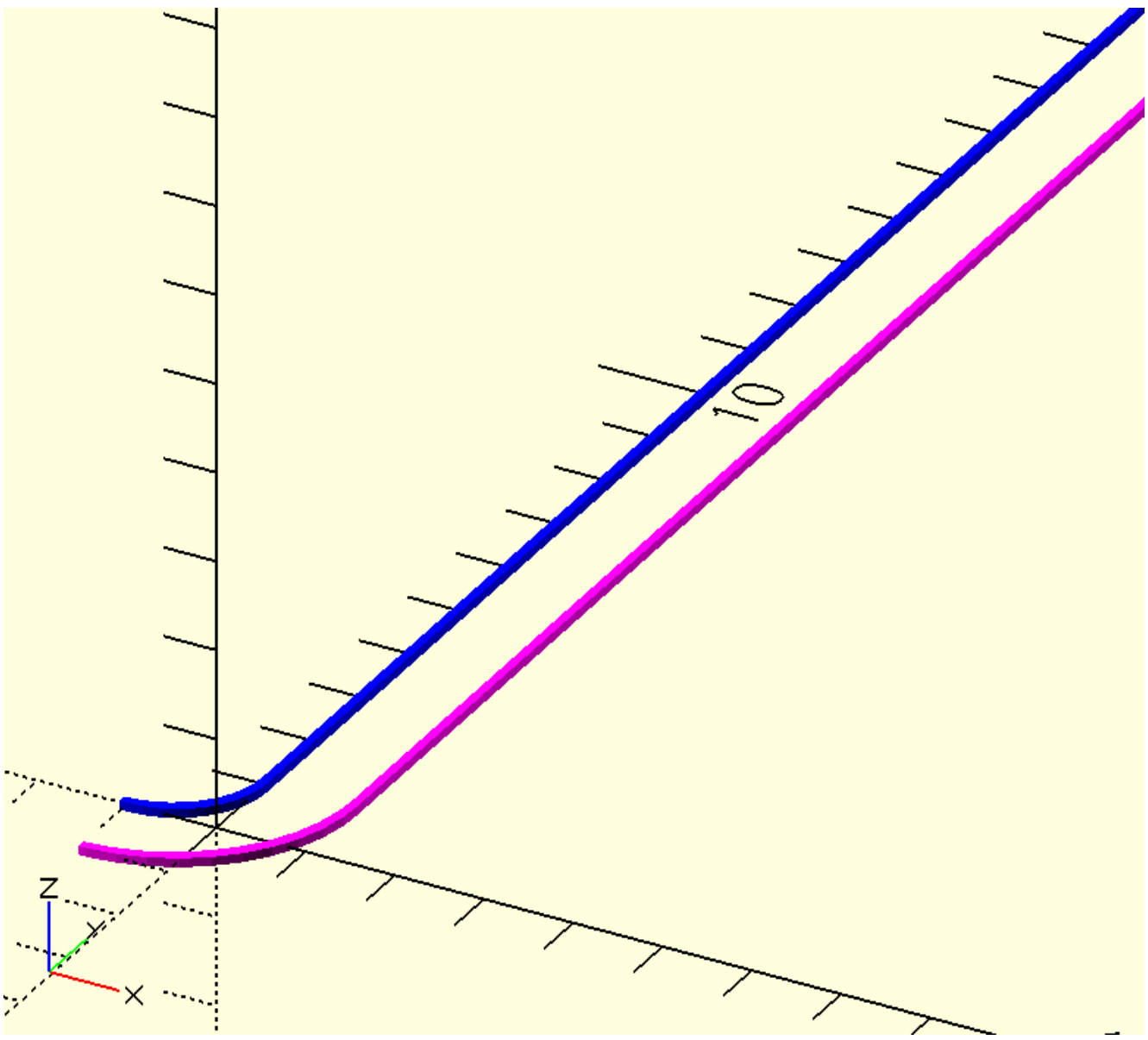
In [25]:
```
# Use function path_extrude_open to create a solid 'sol2'.
# another function align_sol_1 is used here to avoid twisted sections while extruding

sol2=align_sol_1(path_extrude_open(sec1,path1))

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3dc({sec1},.1,1);
color("magenta")p_line3d({path1},.2,1);
{swp(sol2)}
    ''')
```



In [26]:
```
# Both the solids together on which the fillet needs to be created

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
{swp(sol1)}
{swp(sol2)}
    ''')
```

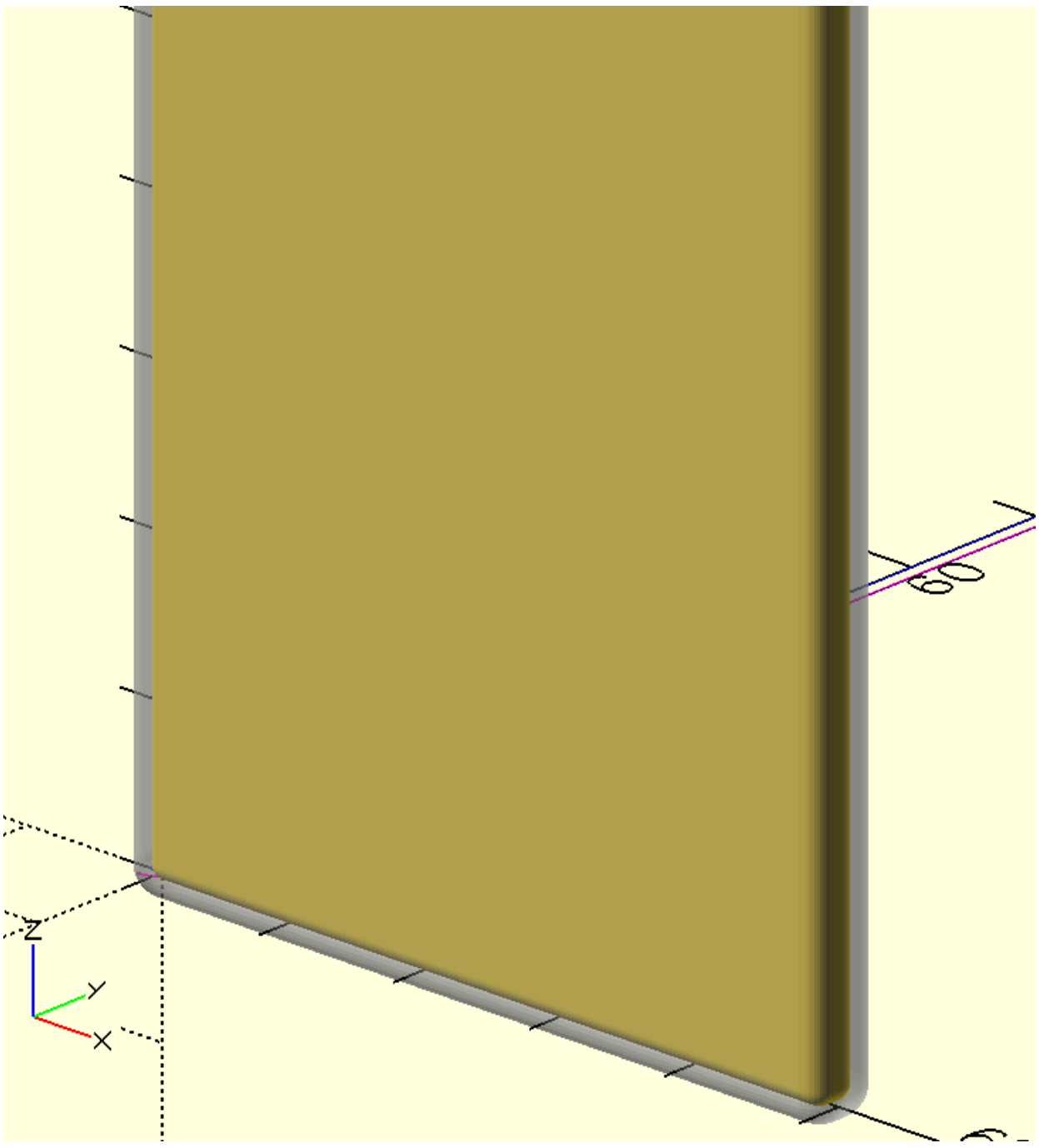


In [27]:
```
# method to offset the 'sol1'
# create an offset path for shaping the 'sec'
p_1=path_offset_n(path,r)
with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3d({path},.1,1);
color("magenta")p_line3d({p_1},.1,1);
    ''')
```

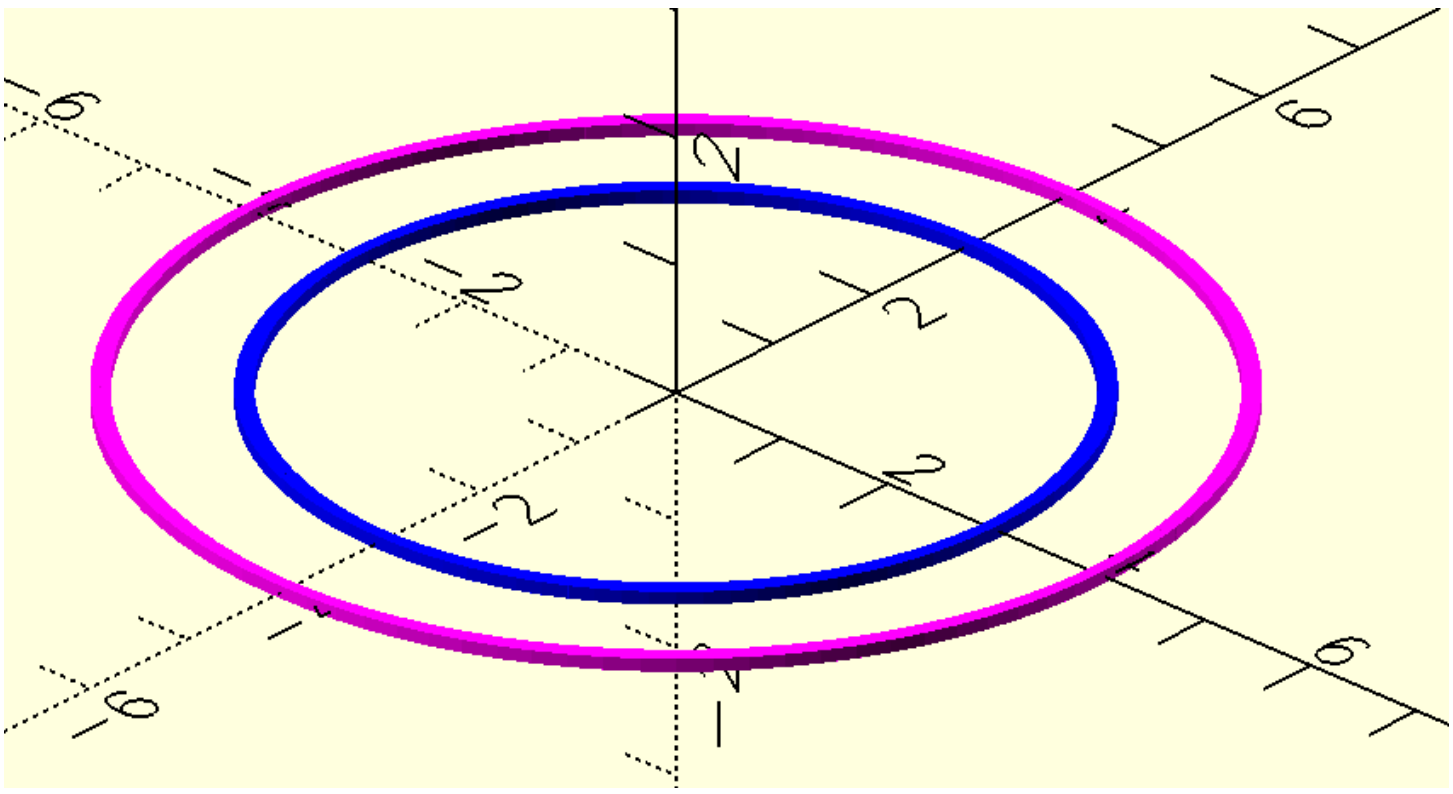


In [28]:
```
# with above offset path, an offset solid can be created
# with the same prism function
```

```
s_1=prism(sec,p_1)

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3d({path},.1,1);
color("magenta")p_line3d({p_1},.1,1);
{swp(sol1)}
%{swp(s_1)}
    ''')
```

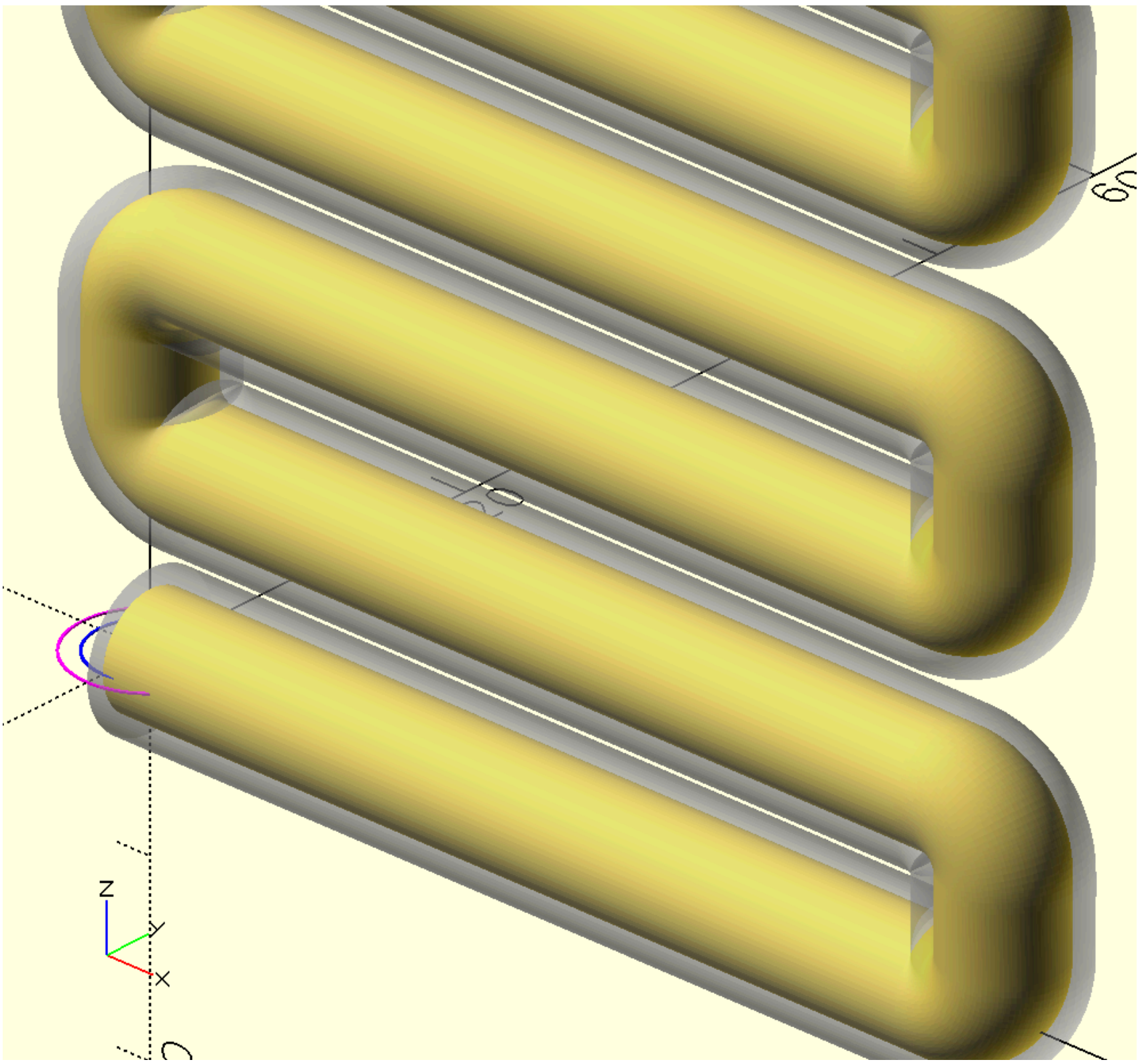


In [29]:
```
# method to offset 'sol2'
# offset the 2D section 'sec1'. Please note here the path offset will not work as that
# will not create the offset solid

sec2=offset(sec1,1)

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3dc({sec1},.1,1);
color("magenta")p_line3dc({sec2},.1,1);

    ''')
```



In [30]:
```
# with the above offset 'sec2' extrude it to the 3D path1 to create an offset solid

s_2=align_sol_1(path_extrude_open(sec2,path1))
with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>
color("blue")p_line3dc({sec1},.1,1);
```

```
color("magenta")p_line3dc({sec2},.1,1);
{swp(sol2)}
%{swp(s_2)}

    ''')
```

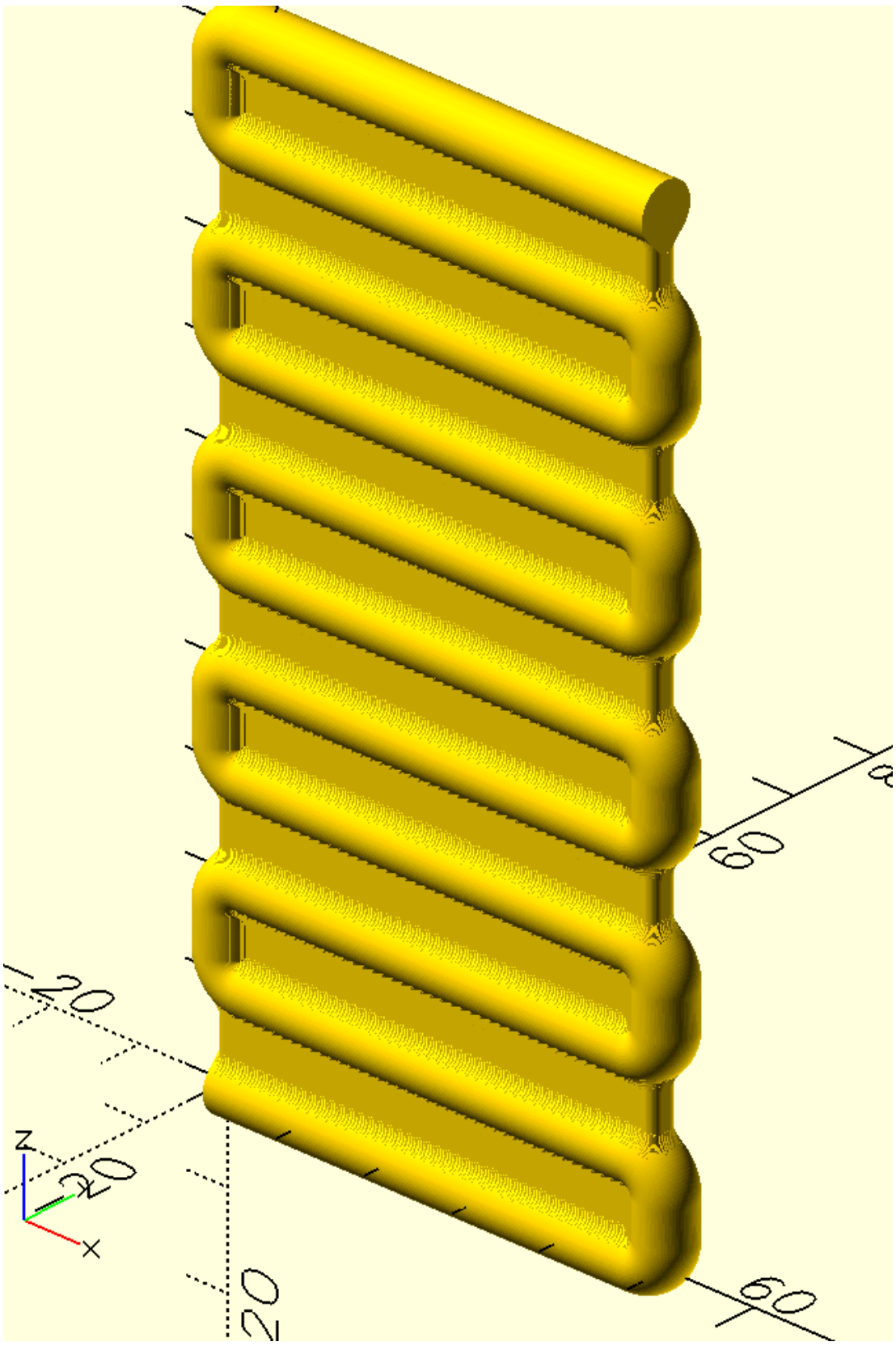


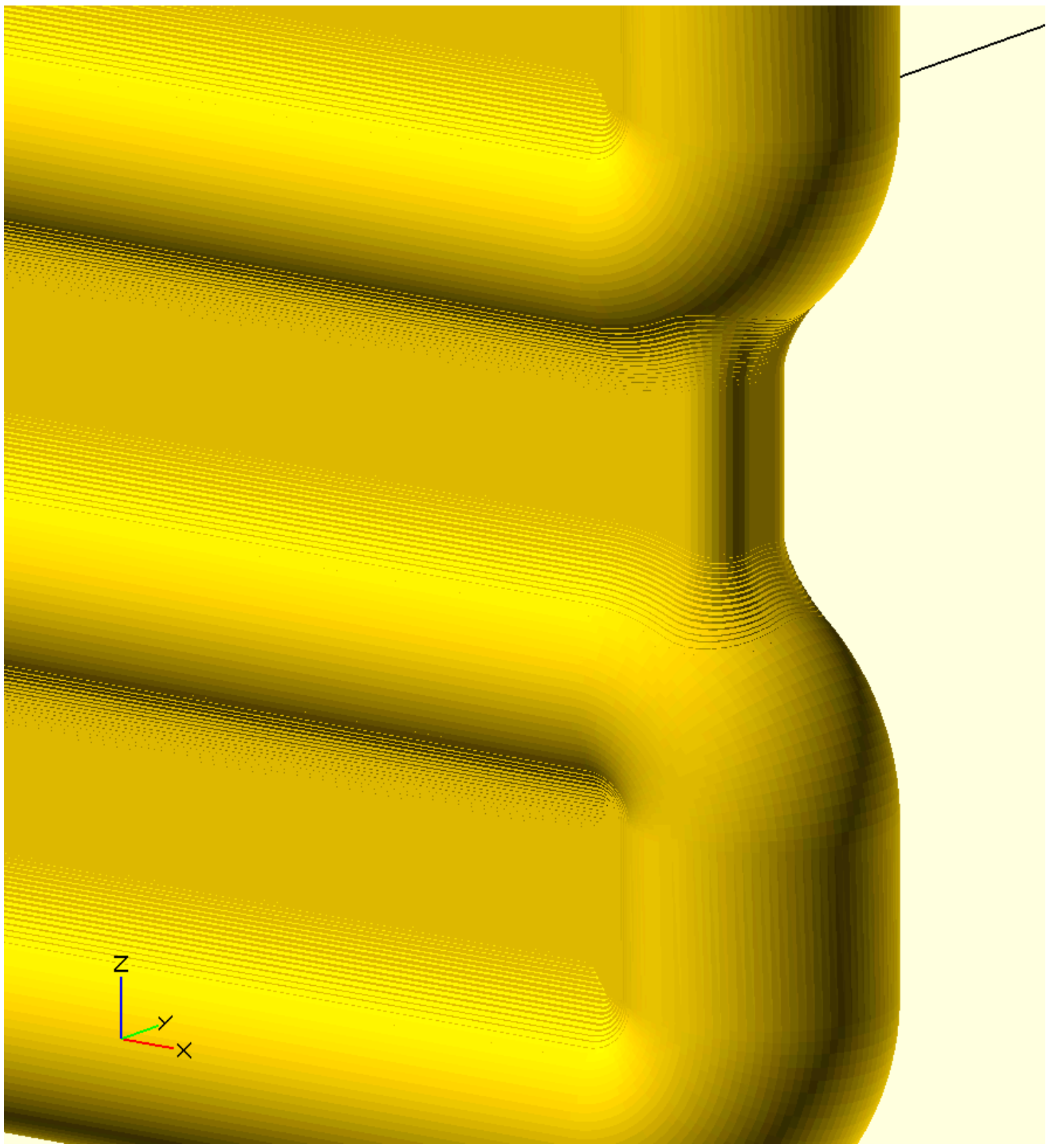
In [ ]:
```
# with the same method mentioned above for sol1, create various copies of offset solid
# Here each point in the arc 'a' x-coordinate value is used to create the offset

l1=[prism(sec,path_offset(path,i[0])) for i in a]
```

In [ ]:
```
# with the same method mentioned above for sol2, create various copies of offset solid
# Here each point in the arc 'a' y-coordinate value is used to create the offset

l2=[align_sol_1(path_extrude_open(offset(sec1,i[1]),path1)) for i in a]
```

In [31]:
```
# Finally create intersections for each pair of sol1 and sol2 offset by the x,y
# coordinates of each point in the arc 'a' and this will create a stepped shape
# of fillet. Smooth fillet is not possible in this case as this is a concave surface
# and the chain hull of the 2 intersections would not result in a fillet.
# the complete code is as below

r=1
a=round([[r+r*cos(d2r(i)),r+r*sin(d2r(i))] for i in linspace(180,270,20)],2)

sec=corner_radius_with_turtle([[0,-1.5,1],[50,0,1],[0,3,1],[-50,0,1]],10)
# sec=m_points1(sec,100,.1)
path=corner_radius_with_turtle([[-1,0],[1,0,1],[0,100,1],[-1,0]],10)

sec1=circle(3,s=100)
path1=corner_radius3d_with_turtle([[0,0,0],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                   [0,0,10,3],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                   [0,0,10,3],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                   [0,0,10,3],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                     [0,0,10,3],[50,0,0,3],[0,0,10,3],[-50,0,0,3],
                                    [0,0,10,3],[50,0,0]
                                   ],30)

sol1=prism(sec,path)
sol2=align_sol_1(path_extrude_open(sec1,path1))

l1=[prism(sec,path_offset(path,i[0])) for i in a]
l2=[align_sol_1(path_extrude_open(offset(sec1,i[1]),path1)) for i in a]

with open('trial.scad','w+') as f:
    f.write(f'''
    include<dependencies2.scad>

{swp(sol1)}
{swp(sol2)}

for(i=[0:len({l1})-1])
intersection(){{
```

```
swp({l1}[i]);
swp({l2}[i]);
}}

    ''')
```




Z
Y
X