Introduction to Computer Science

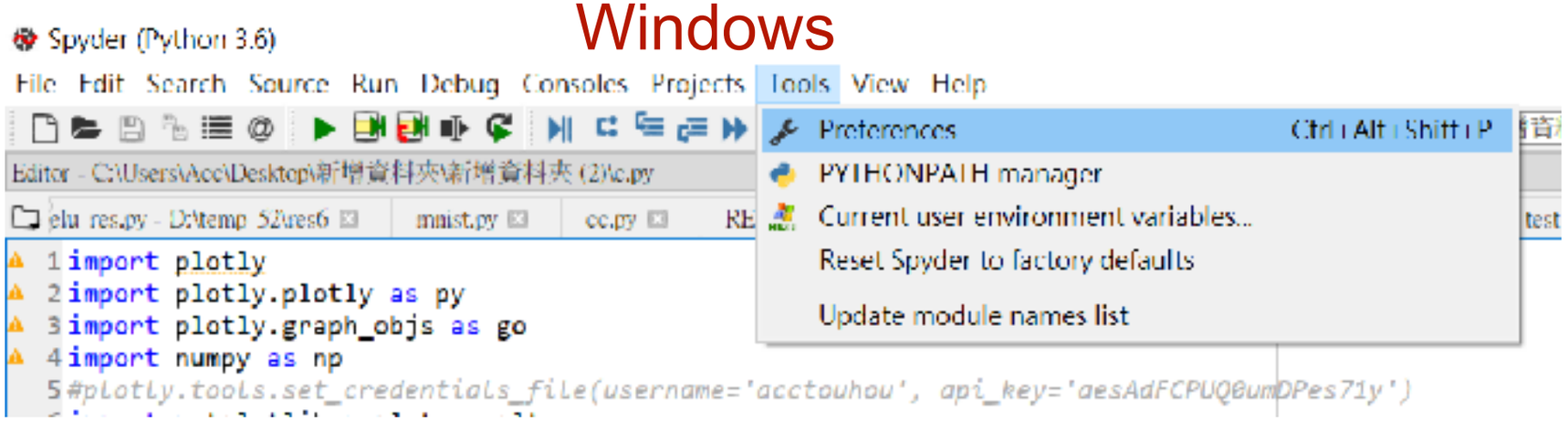
3D Plot

鄒年棣 (Nien-Ti Tsou)
楊仲齊

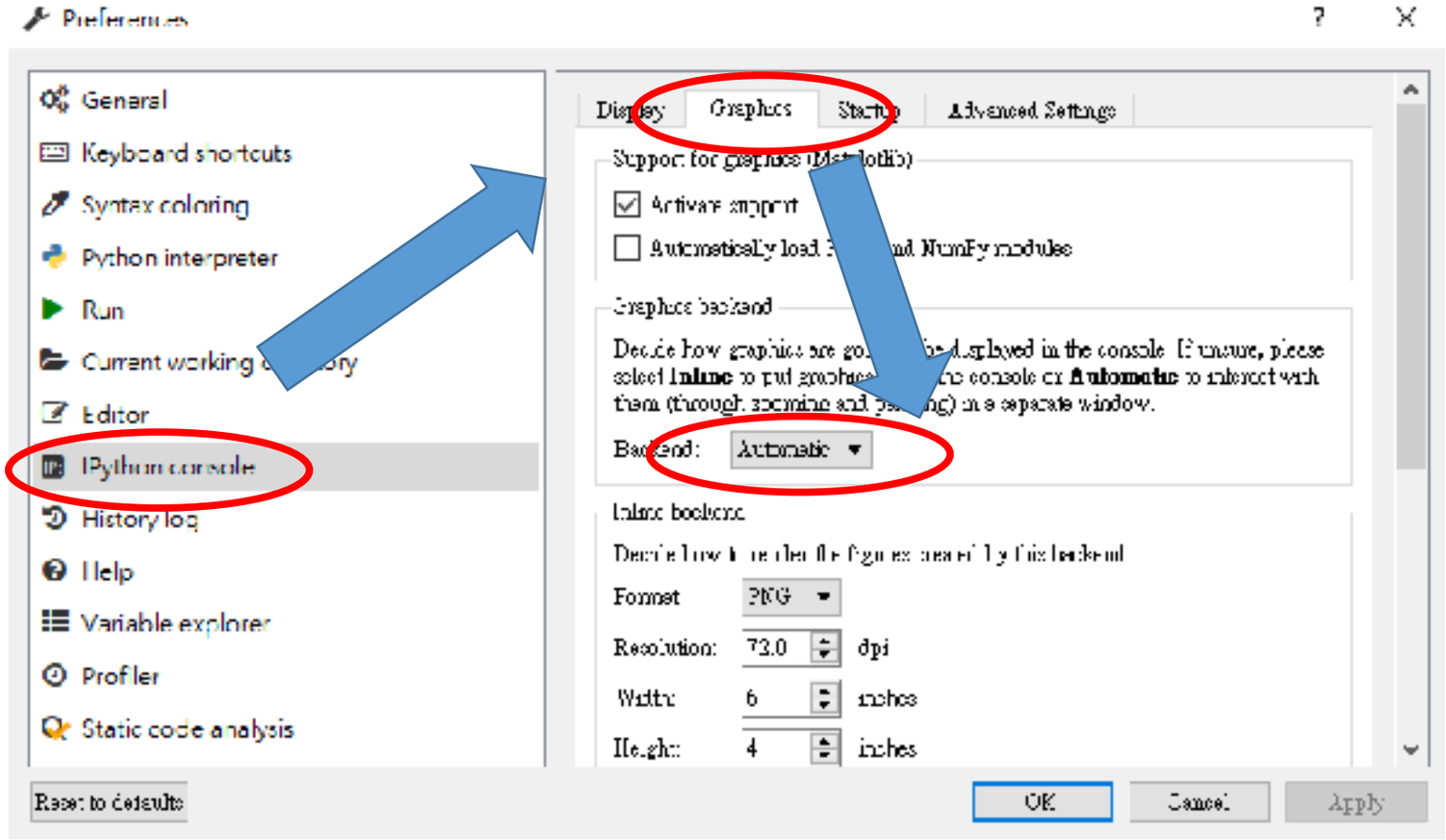
Important settings

- In order to have interactive screen for Python3.

Windows



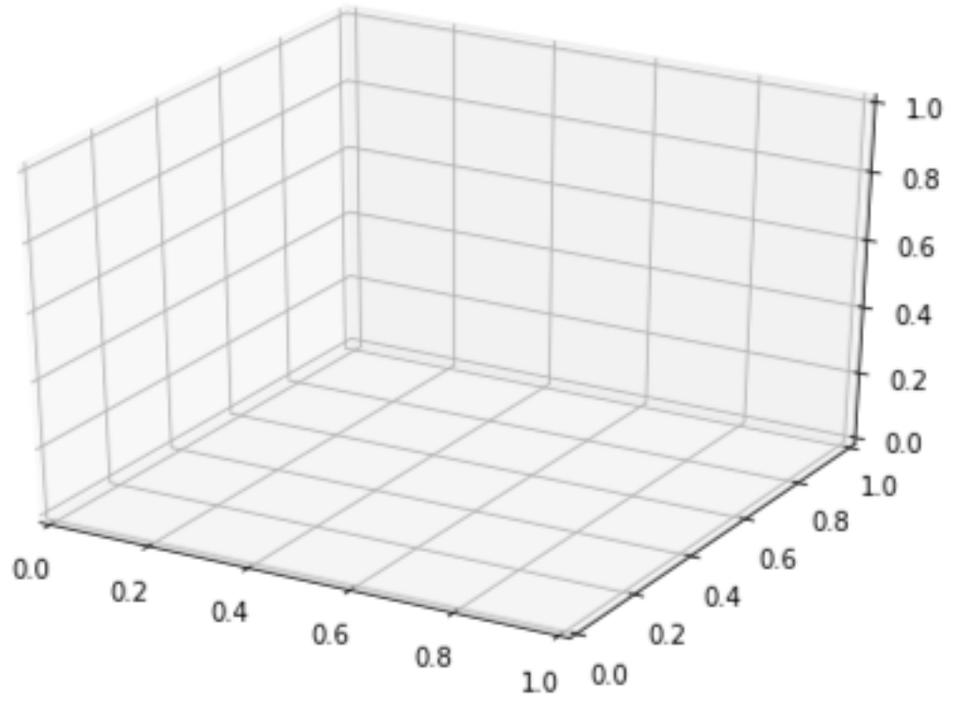
Mac



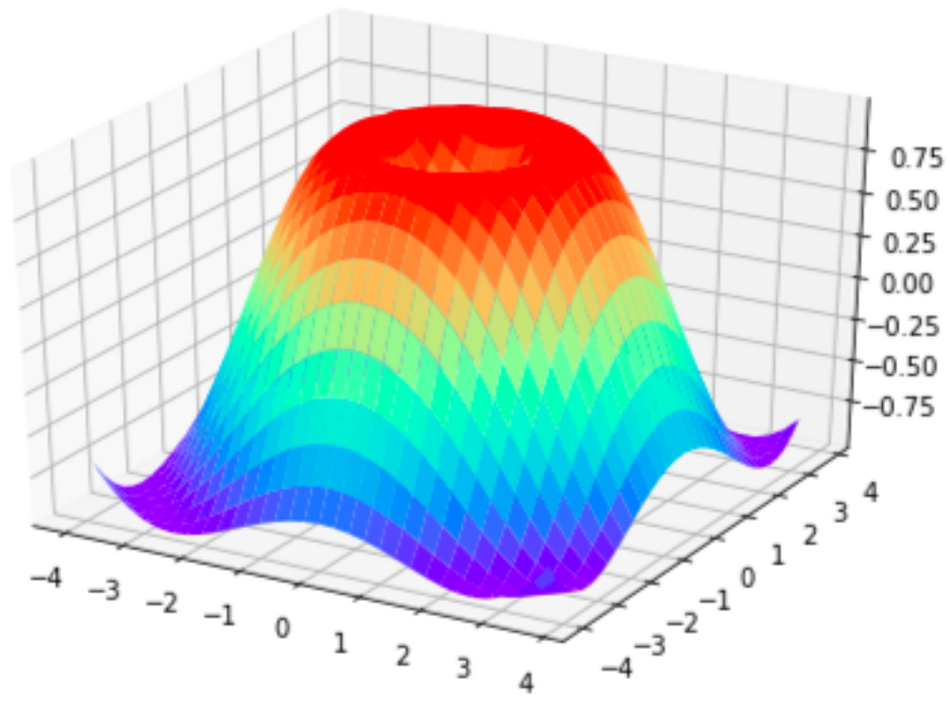
Restart!!!!

Contour in 3D

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d as mmp

fig = plt.figure()
ax = mmp.Axes3D(fig)
```




```
x = np.arange(-4, 4, 0.25)
y = np.arange(-4, 4, 0.25)
X, Y = np.meshgrid(x, y)
R = np.sqrt(X ** 2 + Y ** 2)
Z = np.sin(R)
ax.plot_surface(X, Y, Z,
                rstride=1,
                cstride=1,
                cmap='rainbow')
fig
```

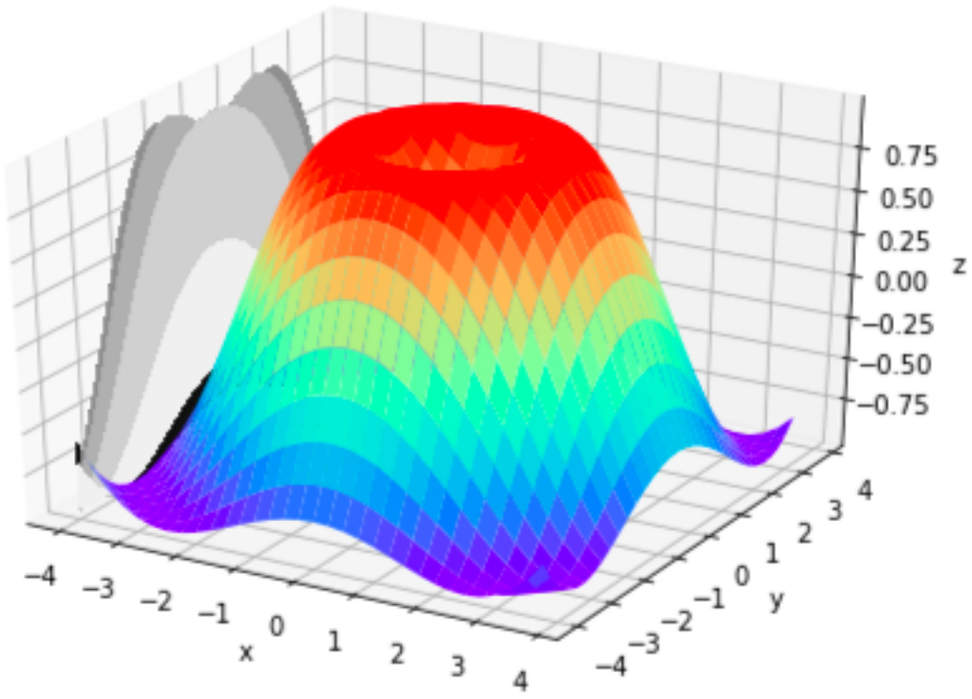


Contour in 3D

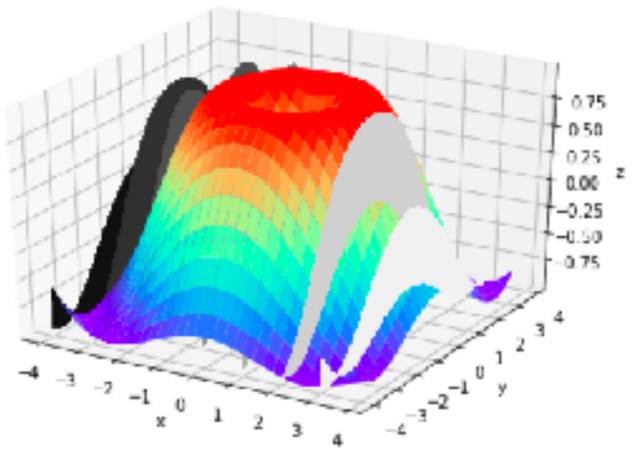
```
ax.contourf(X, Y, Z,
            zdir='x',
            offset=-4,
            cmap='gray')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
fig
```

Projects on the y-z plane

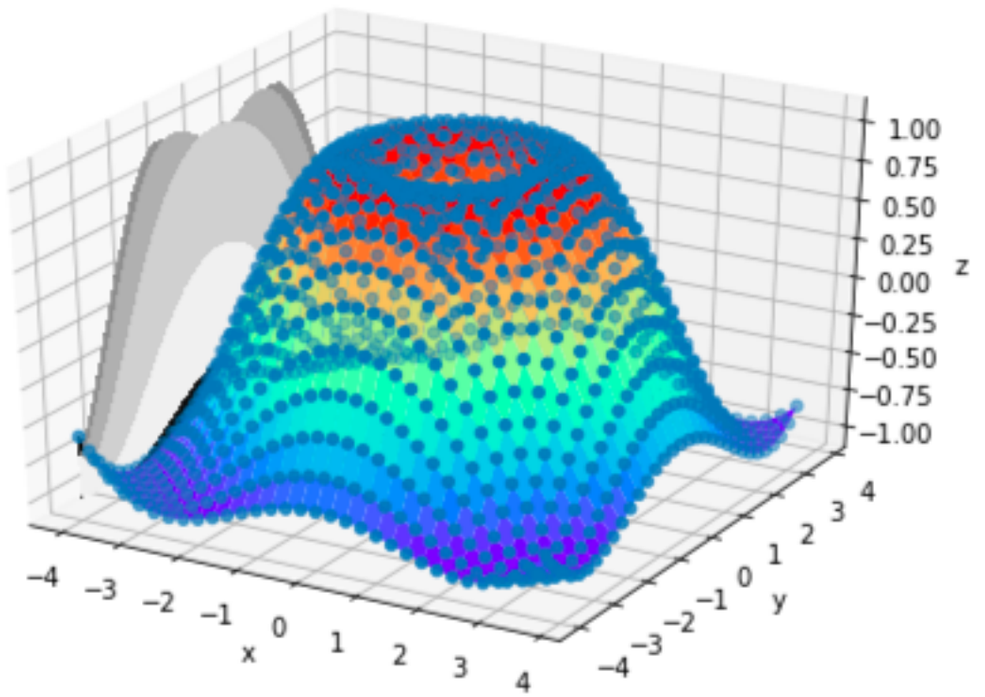
Projects on x = -4



No offset: projects on the ticks with integers

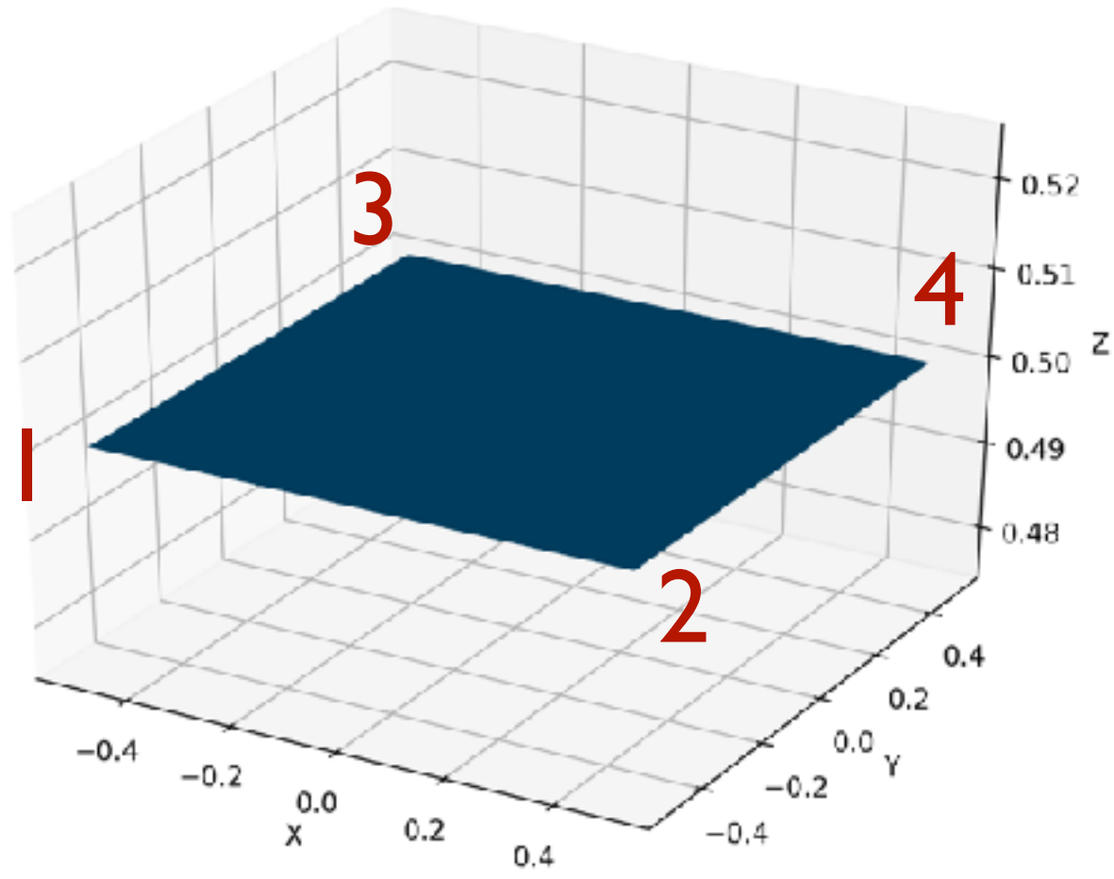



```
ax.scatter3D(X,Y,Z)
fig
```



Draw a patch

```
import numpy as np
import mpl_toolkits.mplot3d as mmp
import matplotlib.pyplot as plt

fig = plt.figure()
ax = mmp.Axes3D(fig)


r = [-0.5,0.5]
X, Y = np.meshgrid(r, r)
one = np.ones([2,2])/2


ax.plot_surface(X,Y,one)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
```




```
In [10]: X
Out[10]:
array([[-0.5,  0.5],
       [-0.5,  0.5]])
```

1 2 3 4

```
In [11]: Y
Out[11]:
array([[-0.5, -0.5],
       [ 0.5,  0.5]])
```

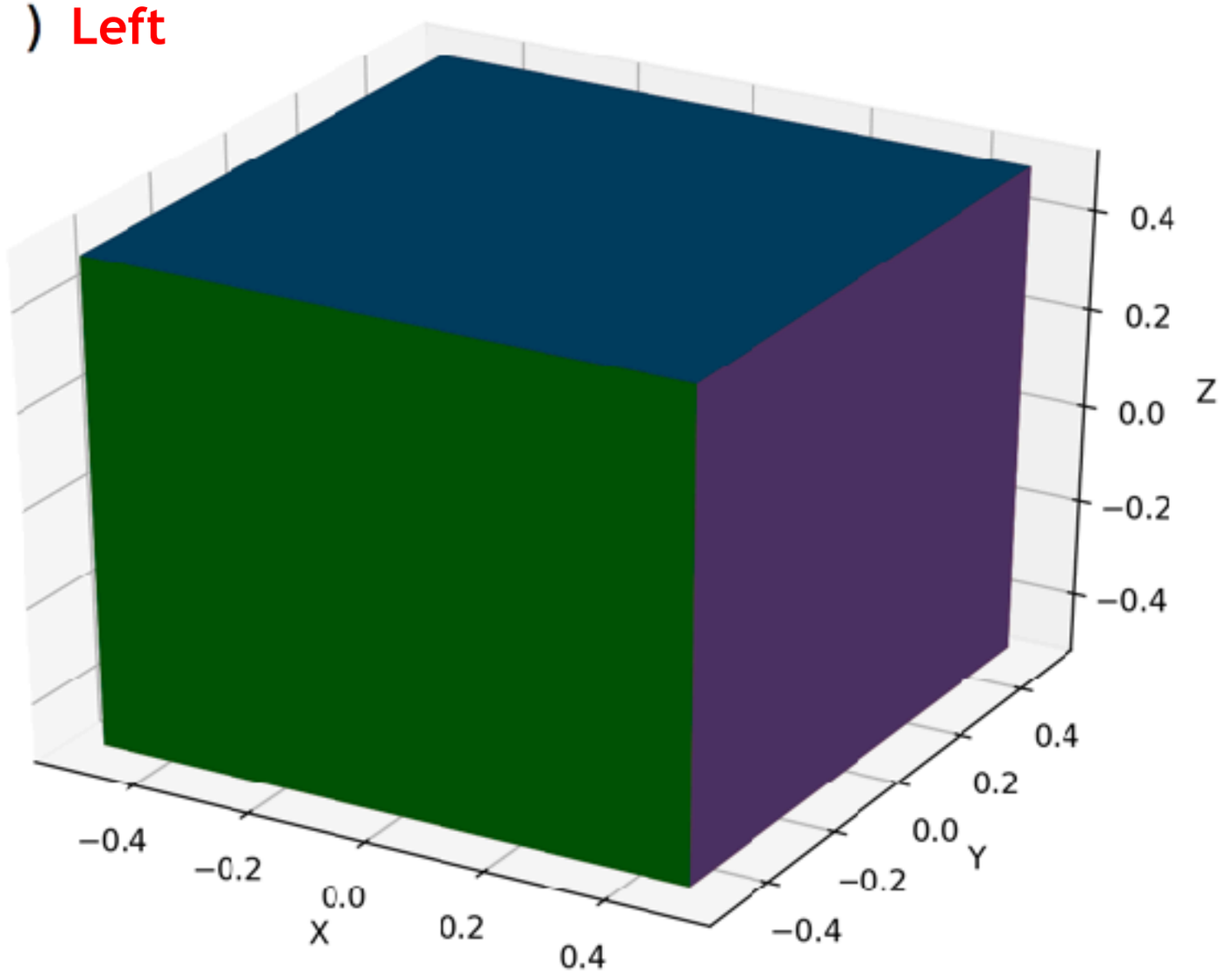
```
In [14]: one
Out[14]:
array([[0.5, 0.5],
       [0.5, 0.5]])
```

Draw a cube

continue…

```
ax.plot_surface(X,Y,-one)
ax.plot_surface(X,-one,Y)
ax.plot_surface(X,one,Y)
ax.plot_surface(one,X,Y)
ax.plot_surface(-one,X,Y )

plt.show()
```

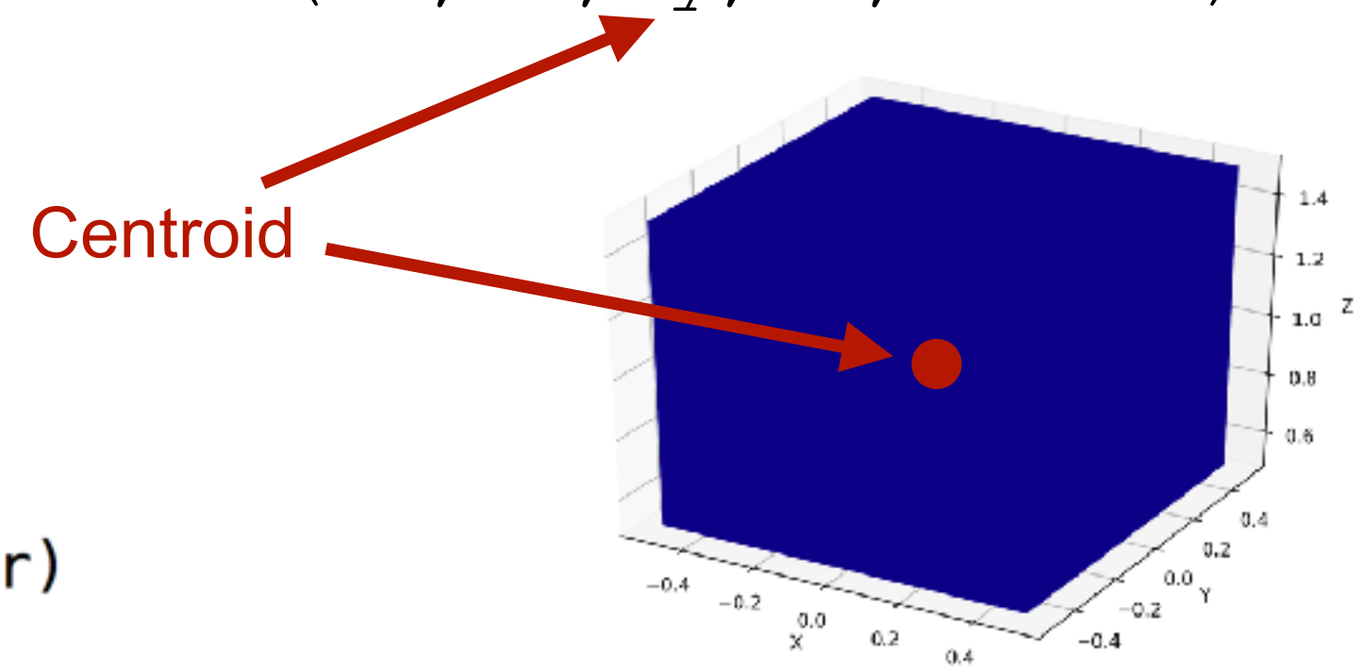
Lower
Front
Rear
Right
Left



Draw a cube in a easier way

- Define a function named `cube(ax, x, y, z, color)`

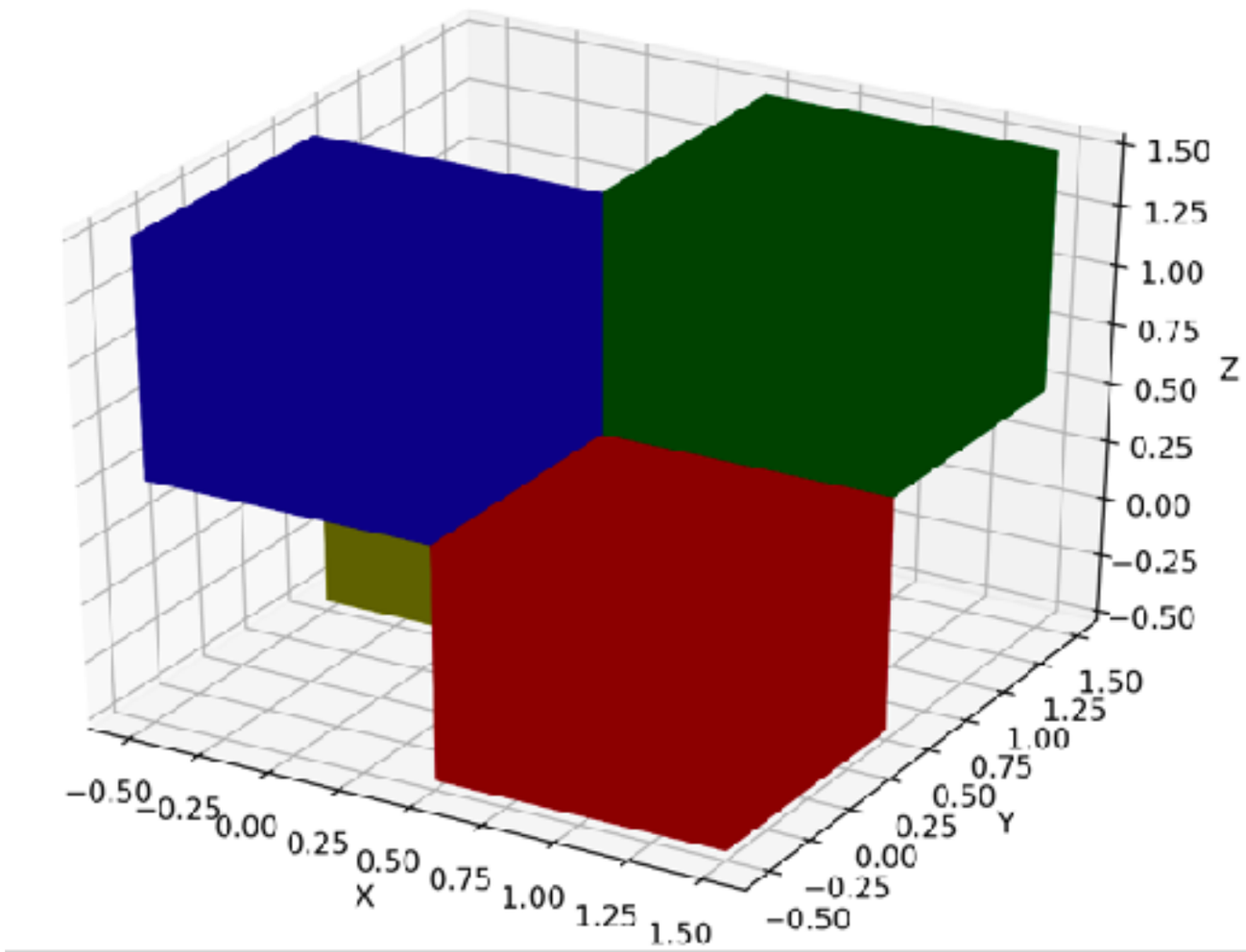
```
fig = plt.figure()
ax = mmp.Axes3D(fig)

def cube(ax,x,y,z,color):
    r = [-0.5,0.5]
    X, Y = np.meshgrid(r, r)
    one = np.ones([2,2])/2
    ax.plot_surface(X+x,Y+y,one+z, color=color)
    ax.plot_surface(X+x,Y+y,-one+z, color=color)
    ax.plot_surface(X+x,-one+y,Y+z, color=color)
    ax.plot_surface(X+x,one+y,Y+z, color=color)
    ax.plot_surface(one+x,X+y,Y+z, color=color)
    ax.plot_surface(-one+x,X+y,Y+z, color=color)

cube(ax,0,0,1,'b')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
```



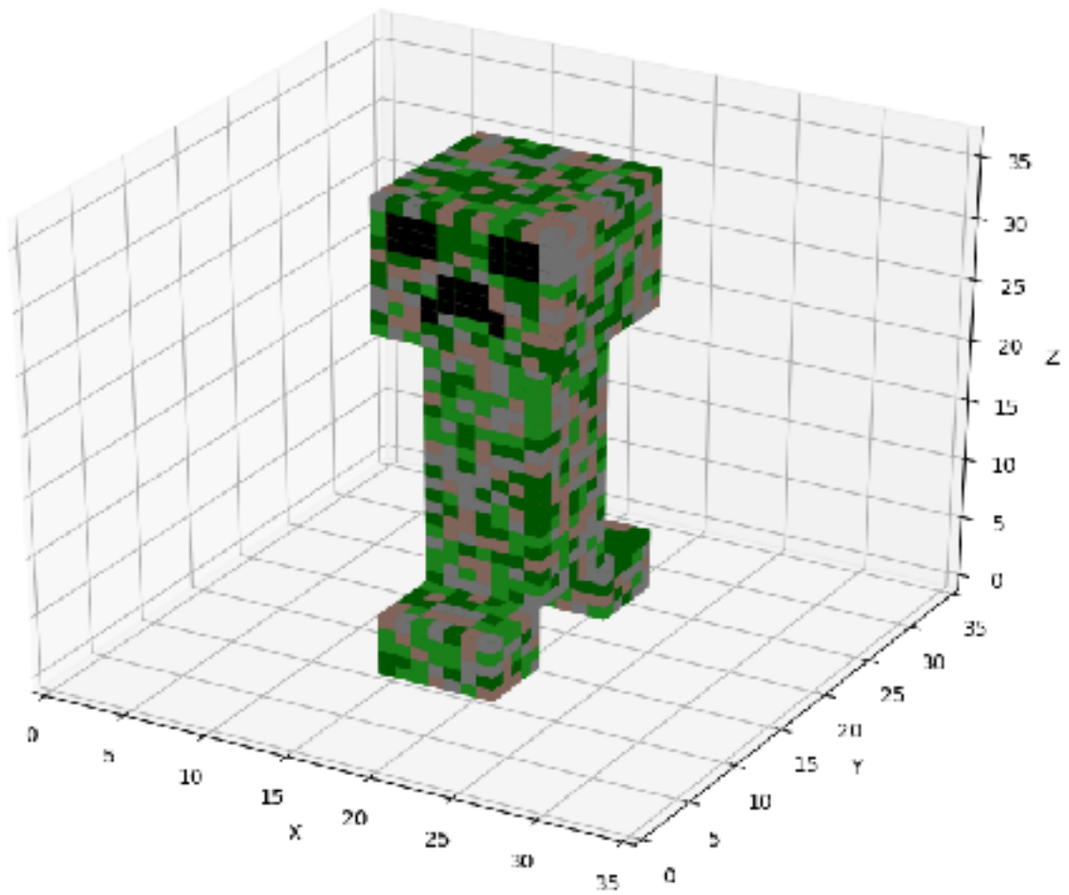
Draw multiple cubes

continue…

```
cube(ax,1,0,0,'r')
cube(ax,1,1,1,'g')
cube(ax,0,1,0,'y')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```



Creeper by 楊仲齊



Some prework suggestions:

- You can design the position and the size before using python to draw your model.

Draw a creeper

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d as mmp

fig = plt.figure()
ax = mmp.Axes3D(fig)


def cube(ax,x,y,z,color,w):
    r = [-0.5,0.5]
X, Y = np.meshgrid(r, r)
one = np.ones([2,2])/2
ax.plot_surface(X+x,Y+y,one+z,color=color,shade=w)
ax.plot_surface(X+x,Y+y,-one+z,color=color,shade=w)
ax.plot_surface(X+x,-one+y,Y+z,color=color,shade=w)
ax.plot_surface(X+x,one+y,Y+z,color=color,shade=w)
ax.plot_surface(one+x,X+y,Y+z,color=color,shade=w)
ax.plot_surface(-one+x,X+y,Y+z,color=color,shade=w)
```

Set the amount of the color.

Whether we want shade or not.

Construct the body

```
#body
color=[   '#00AA00','#33FF33','#DDDDDD','#FFC8B4']
```

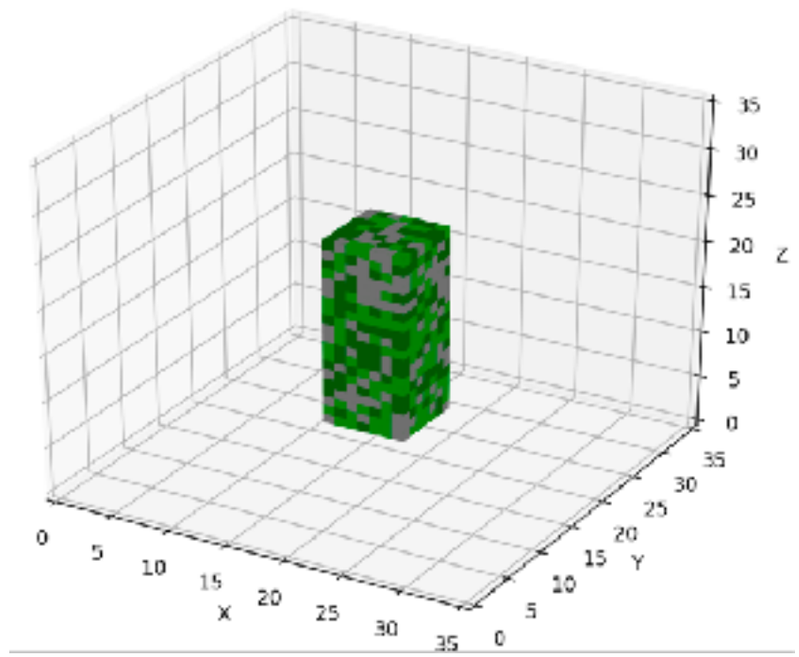
#00AA00 #33FF33 #DDDDDD #FFC8B4

```
for i in range(15,22):
    for j in range(15,22):
        for k in range(5,25):
            cube(ax,i,j,k,color[np.random.randint(0,3)],True)
```

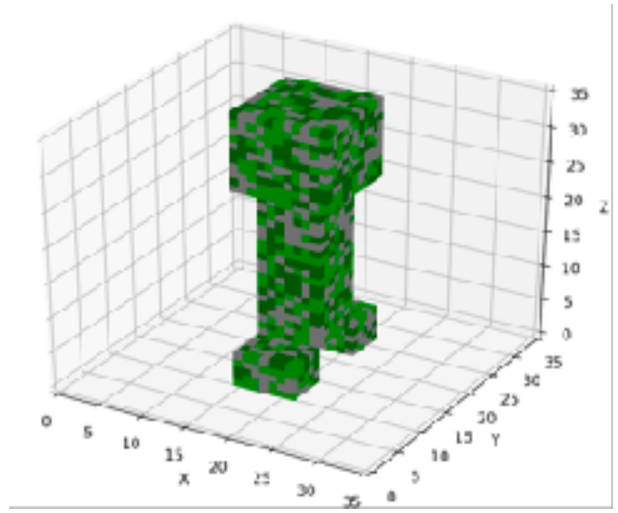
Return a random integer N, such that 0 <= N <= 3.

Here we want to generate a random color for each cube.

```
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_xlim3d(0,35)
ax.set_ylim3d(0,35)
ax.set_zlim3d(0,35)
```



Construct the head and the legs



```
#head
for a in range(13,24):
    for b in range(13,24):
        for c in range(26,37):
            cube(ax,a,b,c,color[np.random.randint(0,3)],True)
```

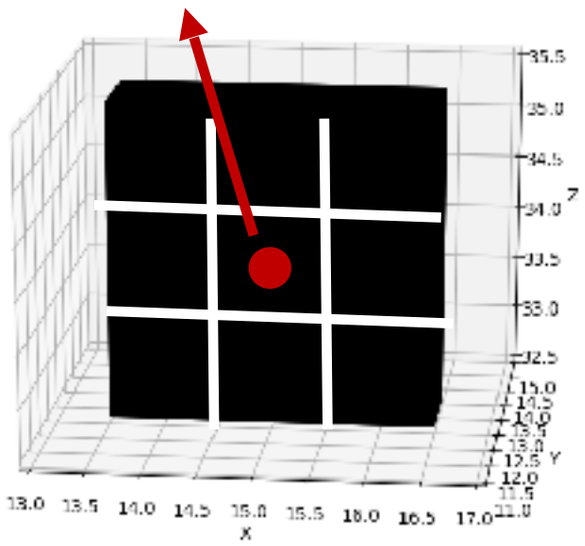
```
#leg
#front
for d in range(15,22):
    for e in range(10,15):
        for f in range(0,5):
            cube(ax,d,e,f,color[np.random.randint(0,3)],True)
#back
for l in range(15,22):
    for m in range(22,27):
        for n in range(0,5):
            cube(ax,l,m,n,color[np.random.randint(0,3)],True)
```

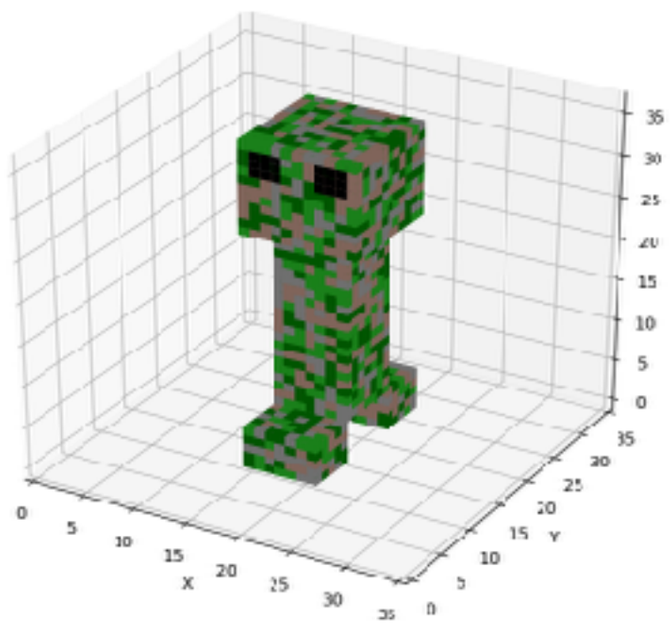
Construct the eyes

- Define a function named `eye (ax, x, y, z).`

```
#eyes
def eye(ax,x,y,z):
    cube(ax,x-1,y,z-1,'#000000',True)
    cube(ax,x,y,z-1,'#000000',True)
    cube(ax,x+1,y,z-1,'#000000',True)
    cube(ax,x-1,y,z,'#000000',True)
    cube(ax,x+1,y,z,'#000000',True)
    cube(ax,x-1,y,z+1,'#000000',True)
    cube(ax,x,y,z+1,'#000000',True)
    cube(ax,x+1,y,z+1,'#000000',True)
    cube(ax,x,y,z,'#000000',True)
eye(ax,15,13,34)
eye(ax,21,13,34)
```

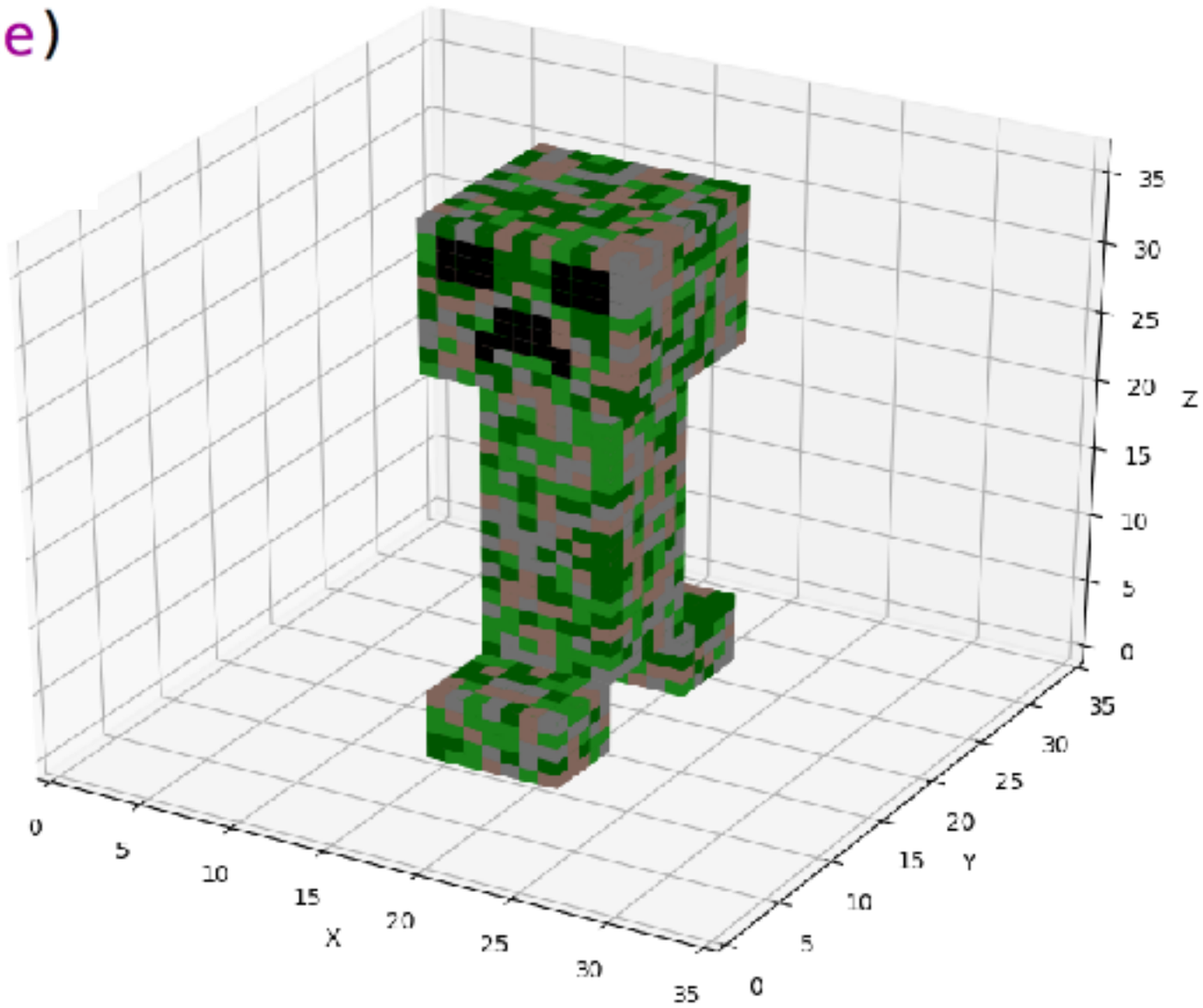
#000000



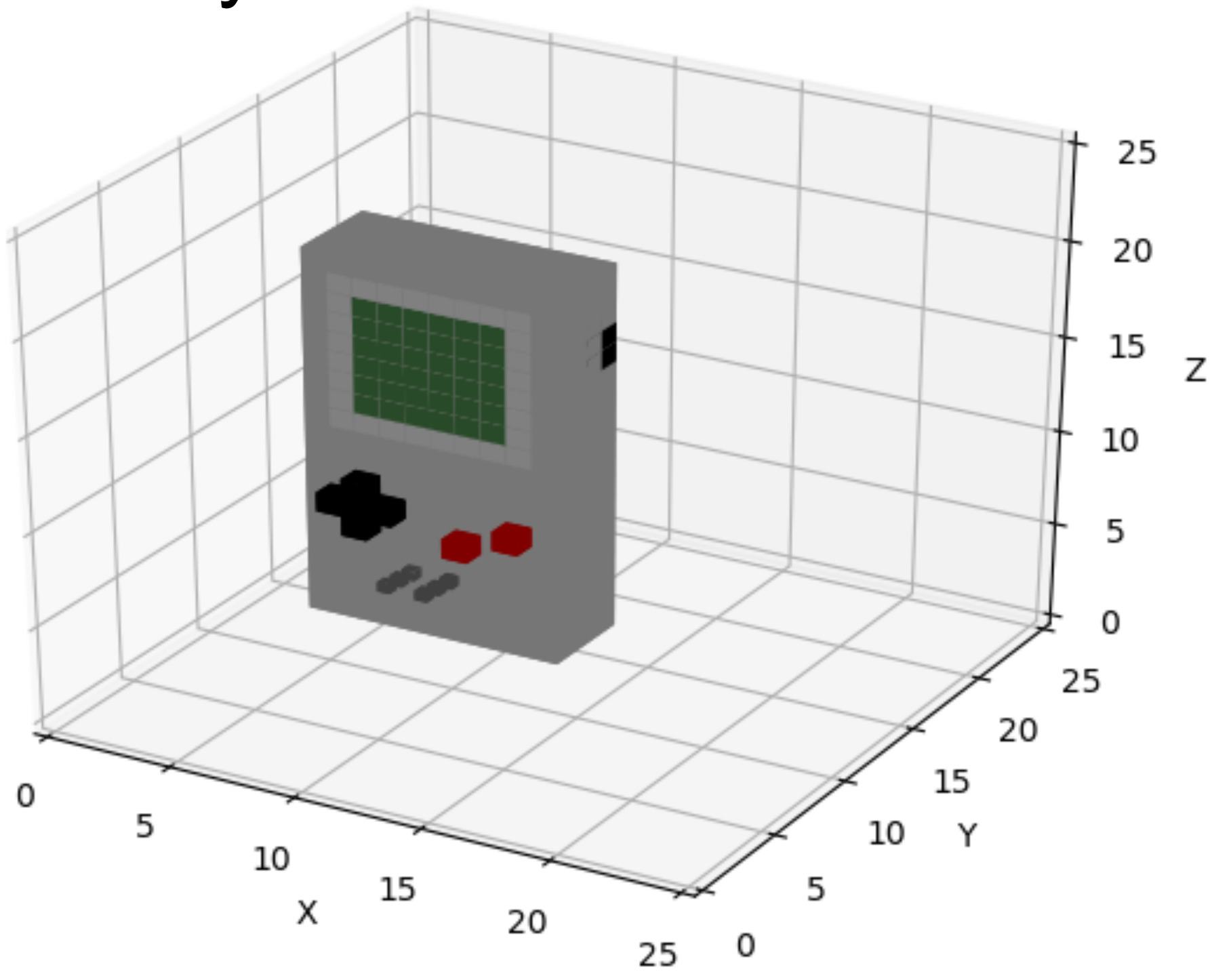


Construct the mouth

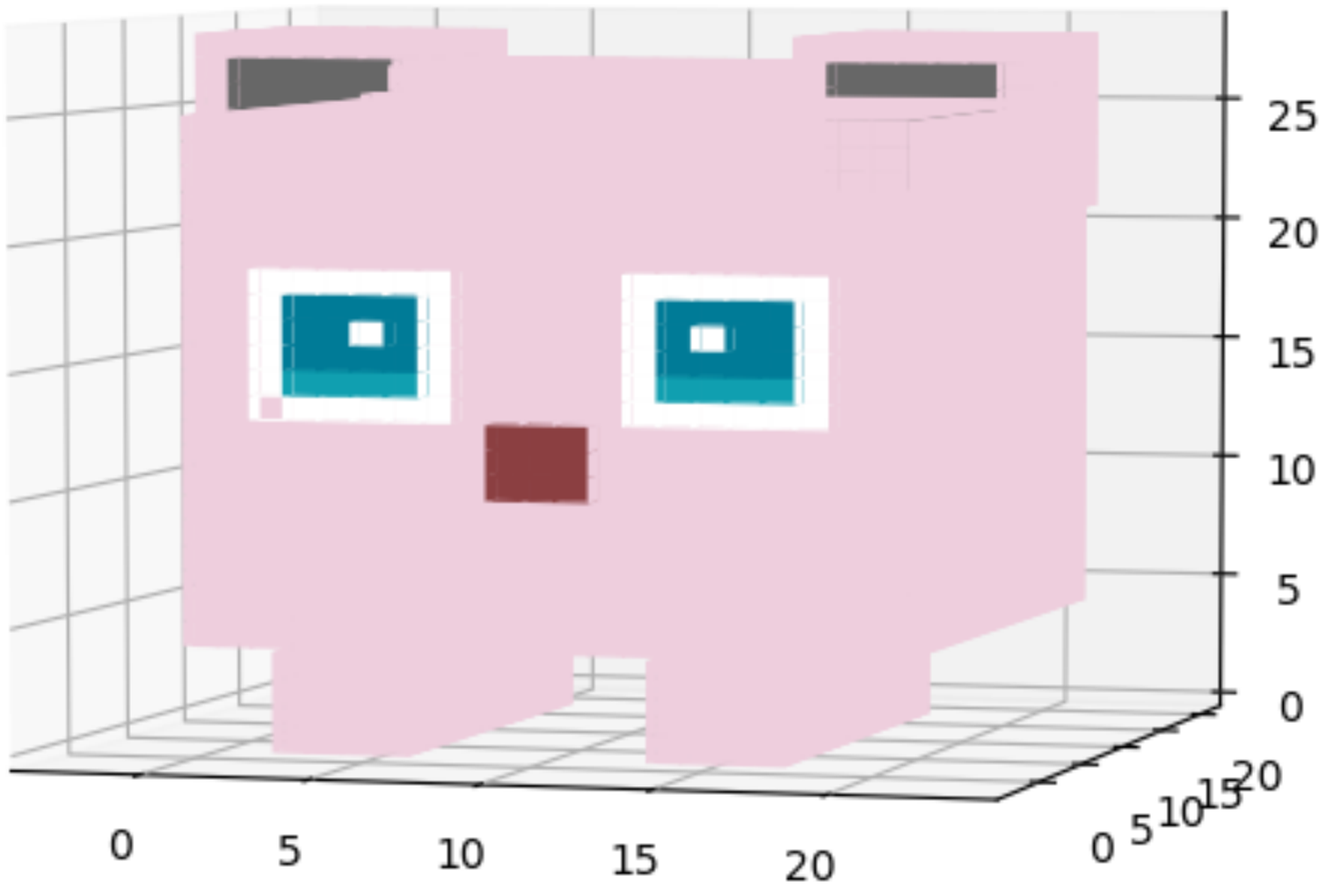
```
#mouth
eye(ax,18,13,30)
cube(ax,16,13,29,'#000000',True)
cube(ax,16,13,28,'#000000',True)
cube(ax,20,13,29,'#000000',True)
cube(ax,20,13,28,'#000000',True)

plt.show()
```



子涵-Gameboy



威鈞-Jigglypuff



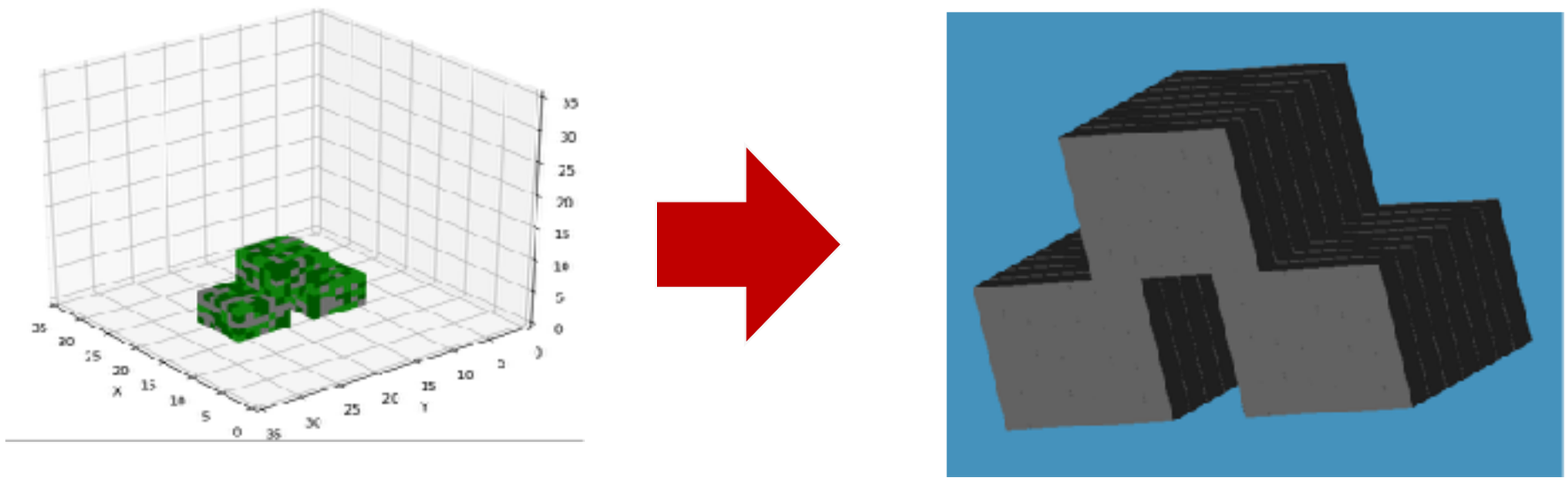
Introduction to Computer Science

3D Printing and STL Files

鄒年棣 (Nien-Ti Tsou)
許家維
楊仲齊

STL

- STL (STero Lithography) is commonly used in preparing solid figure for 3D printing.
- The 3D geometry data is typically written in the form of binary or ASCII variants.



Append the points

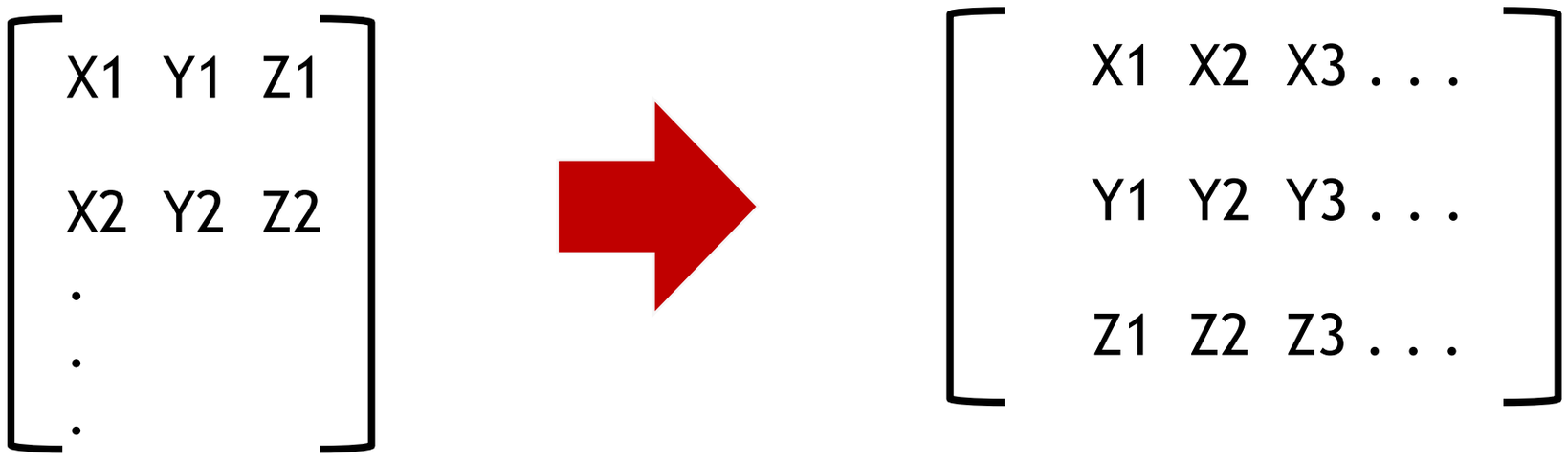
```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d as mmp

fig = plt.figure()
ax = mmp.Axes3D(fig)

lx=[]
ly=[]
lz=[]
def cube_RecordPoint(ax,x,y,z,color,w):
    r = [-0.5,0.5]
    X, Y = np.meshgrid(r, r)
    one = np.ones([2,2])/2
    ax.plot_surface(X+x,Y+y,one+z,color=color,shade=w)
    ax.plot_surface(X+x,Y+y,-one+z,color=color,shade=w)
    ax.plot_surface(X+x,-one+y,Y+z,color=color,shade=w)
    ax.plot_surface(X+x,one+y,Y+z,color=color,shade=w)
    ax.plot_surface(one+x,X+y,Y+z,color=color,shade=w)
    ax.plot_surface(-one+x,X+y,Y+z,color=color,shade=w)
    lx.append(x)
    ly.append(y)
    lz.append(z)
```

Make a list of sequence of every point.

It will automatically save the x, y, z position everytime when `cube_RecordPoint()` is used.

Define `save()` function

- Define a `save()` function to obtain a `3*i` matrix, then transform into a `i*3` matrix.
- Then save the text file named '`loc`' in order for file transforming process.



```python
def save():
    faces=np.vstack((np.array(lx),
                     np.array(ly),
                     np.array(lz))).T
    ### Quiz ##################################
    print(np.vstack((np.array(lx),
                     np.array(ly),
                     np.array(lz))))
    print(np.vstack((np.array(lx),
                     np.array(ly),
                     np.array(lz))).T)
    ##########################################
    np.savetxt('loc',faces)


### Draw your 3D model here ###################
cube_RecordPoint(ax,1,2,3,'blue',True)
cube_RecordPoint(ax,4,5,6,'blue',True)
### Quiz ######################################
print(lx)
print(ly)
print(lz)
##############################################
save()
```

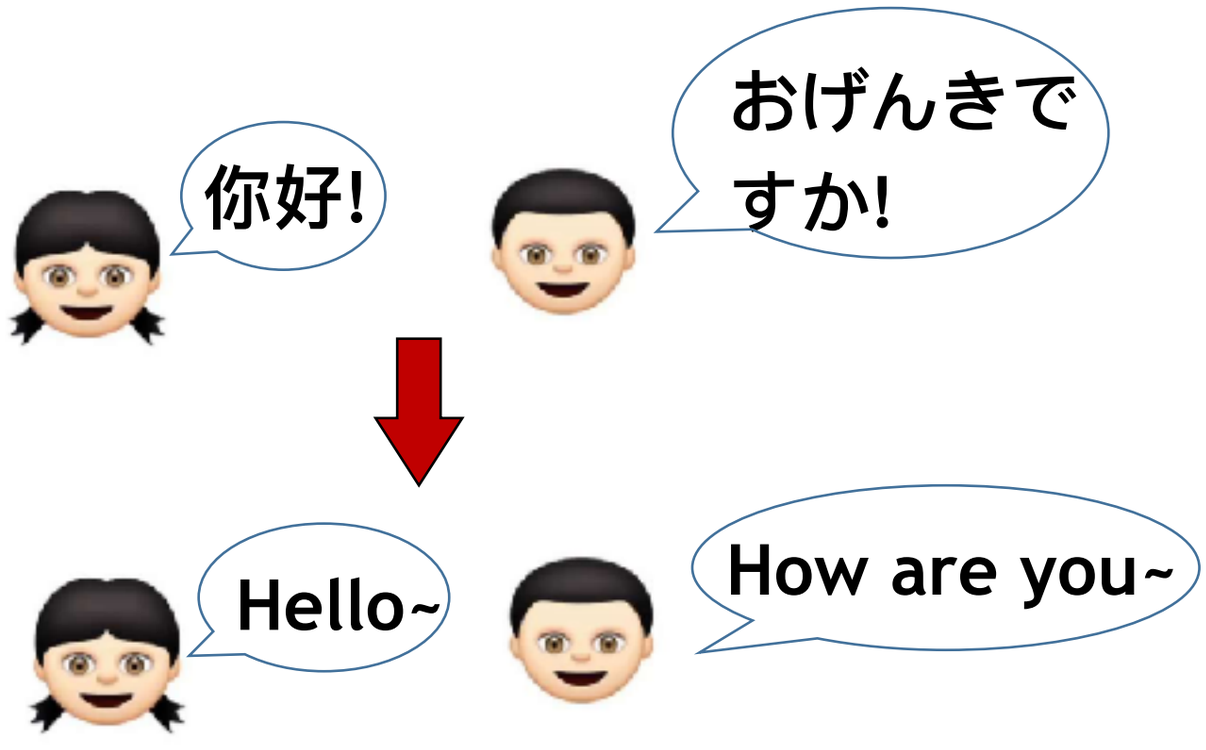
loc

```
1.000000000000000000e+00 2.000000000000000000e+00 3.000000000000000000e+00
4.000000000000000000e+00 5.000000000000000000e+00 6.000000000000000000e+00
```

`to_stl.py` by 許家維

- We use the `struct` module, when Python communicates with binary datum.
- Then assure every data in loc file is separated by delimiter ‘ ’.
- Determine binary data parameters.

```
import struct
import numpy as np
loc=np.loadtxt('loc',delimiter=' ')

BINARY_HEADER ="80sI"
BINARY_FACET = "12fH"
```



ASCII_STL_Writer

- We create a class named `ASCII_STL_Writer` to convert the loc file to binary datum.

```
class ASCII_STL_Writer:
    def __init__(self, stream):
        self.fp = stream
        self._write_header()
    def _write_header(self):
        self.fp.write("solid python\n")
    def close(self):
        self.fp.write("endsolid python\n")
def _write(self, face):
    self.fp.write(ASCII_FACET.format(face=face))
def _split(self, face):
    p1, p2, p3, p4 = face
    return (p1, p2, p3), (p3, p4, p1)
def add_face(self, face):
    if len(face) == 4:
        face1, face2 = self._split(face)
        self._write(face1)
        self._write(face2)
    elif len(face) == 3:
        self._write(face)
    else:
        raise ValueError('only 3 or 4 vertices for each face')

def add_faces(self, faces):
    for face in faces:
        self.add_face(face)
```

Binary_STL_Writer

- Then create a class named `Binary_STL_Writer` which include all the function in the previous class.
- It is for converting the binary datum into STL file from.

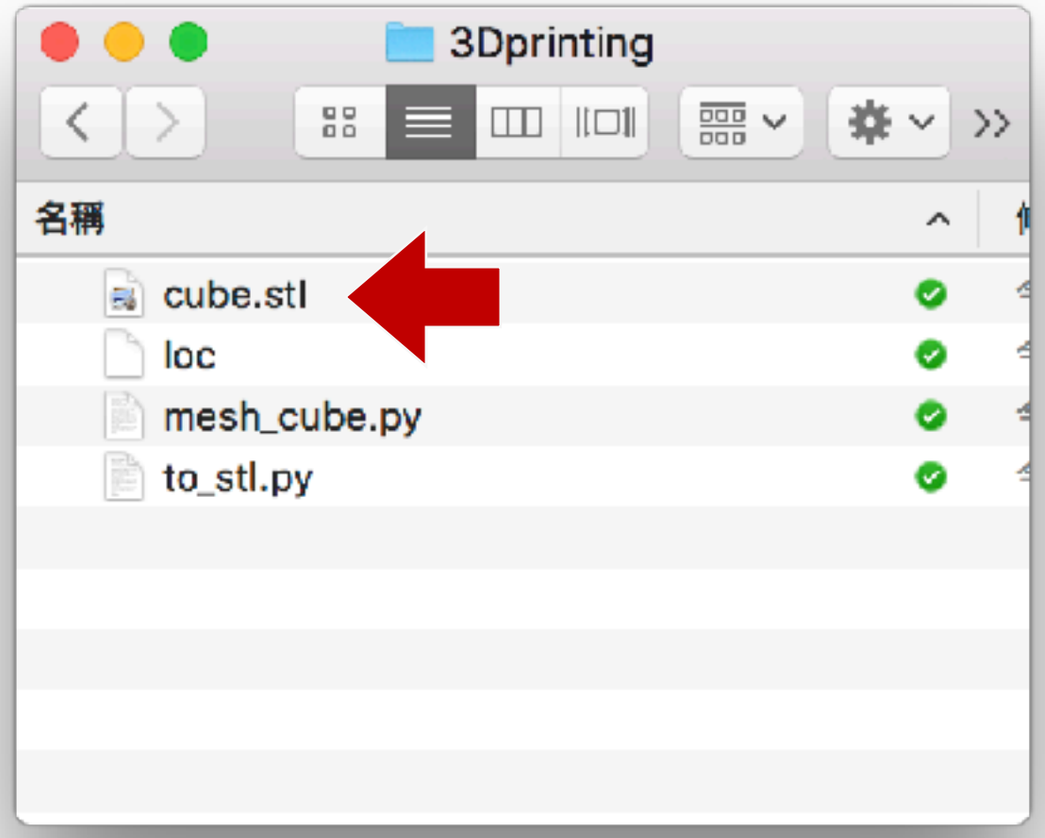
```
class Binary_STL_Writer(ASCII_STL_Writer):
    def __init__(self, stream):
        self.counter = 0
        super(Binary_STL_Writer, self).__init__(stream)
    def close(self):
        self._write_header()
def _write_header(self):
    self.fp.seek(0)
    self.fp.write(struct.pack(BINARY_HEADER, b'Python Binary STL Writer', self.counter))
def _write(self, face):
    self.counter += 1
    data = [
        0., 0., 0.,
        face[0][0], face[0][1], face[0][2],
        face[1][0], face[1][1], face[1][2],
        face[2][0], face[2][1], face[2][2],
        0
    ]
    self.fp.write(struct.pack(BINARY_FACET, *data))
```

Define `example()` to convert into STL

- Define a new function named example() to execute the converting process.

```
def example():
    def get_cube(x,y,z):
        s = 1.
        p1 = (x, y, z)
        p2 = (x, y, z+s)
        p3 = (x, y+s, z)
        p4 = (x, y+s, z+s)
        p5 = (x+s, y, z)
        p6 = (x+s, y, z+s)
        p7 = (x+s, y+s, z)
        p8 = (x+s, y+s, z+s)
        return [
            [p1, p5, p7, p3],
            [p1, p5, p6, p2],
            [p5, p7, p8, p6],
            [p7, p8, p4, p3],
            [p1, p3, p4, p2],
            [p2, p6, p8, p4],
        ]
    with open('cube.stl', 'wb') as fp:
        writer = Binary_STL_Writer(fp)
        for i in range(len(loc)):
            writer.add_faces(get_cube(loc[i,0],loc[i,1],loc[i,2]))
        writer.close()
example()
```

View your STL



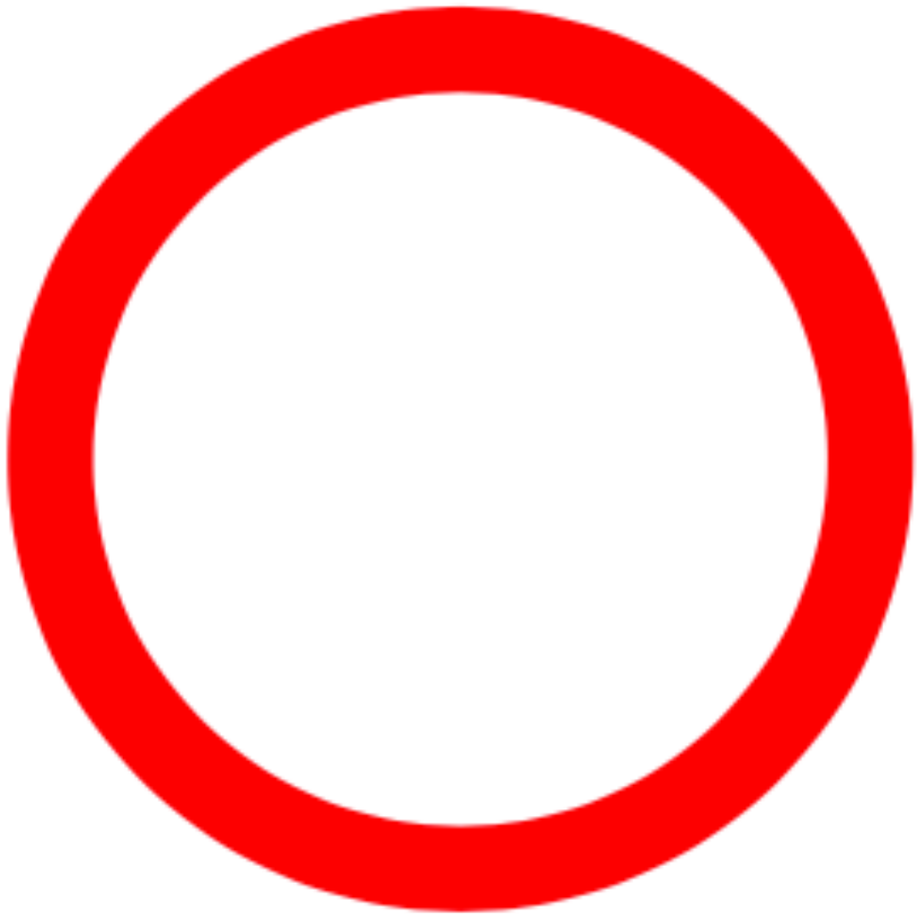
MeshLab

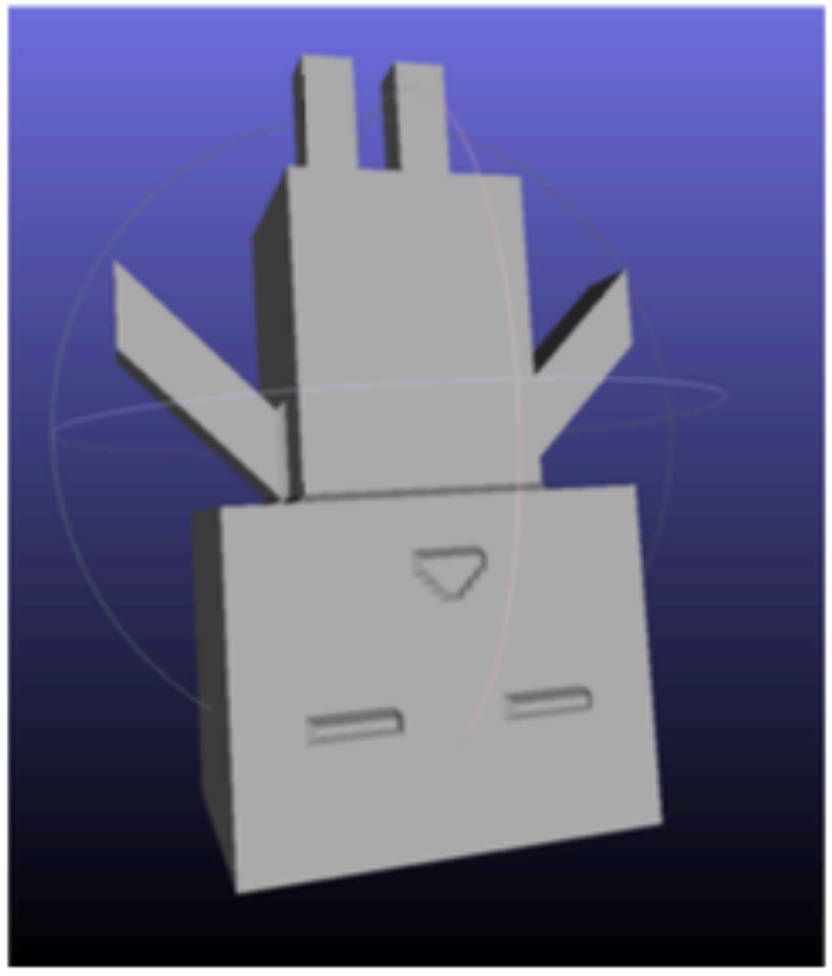
- For a 3D preview

http://www.meshlab.net/

NOTE!

- Be careful about gravity when printing





Homework: Minecraft and 3D printing

- Create your own 3D patch artwork.
- Turn your patch work into 3D printable STL.
- The cooler, the better. The better, the higher score.
- We will vote for the Top 5. Their artwork will be printed as a gift.