Quiz

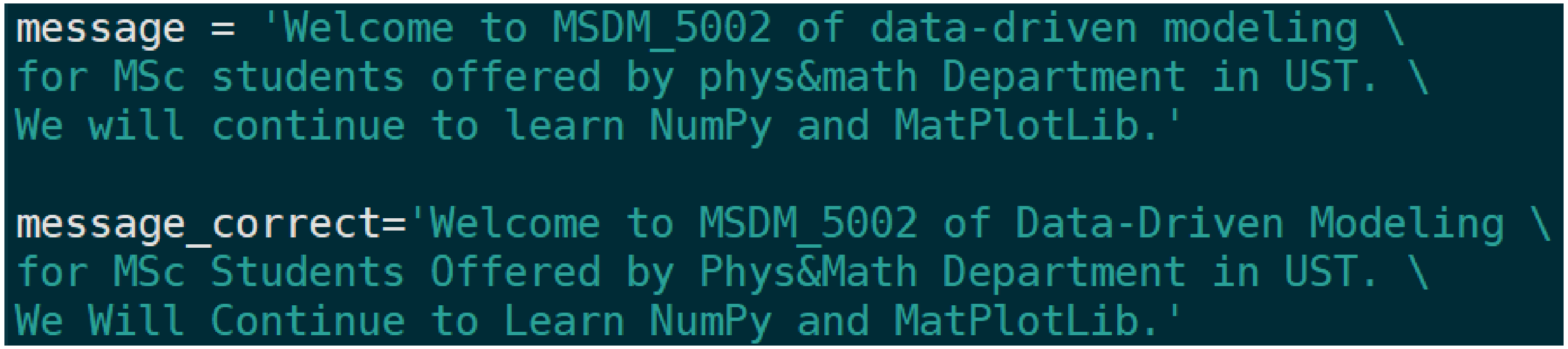
Question: Capitalize all words in a title entered, except a, an, the, at, by, for, in, of, on, to, up, and, as, but, or, and nor. Your output should look like as following.

Enter a title: Welcome to MSDM_5002 of data-driven modeling for MSc students offered by phys&math department in UST. We will continue to learn NumPy.

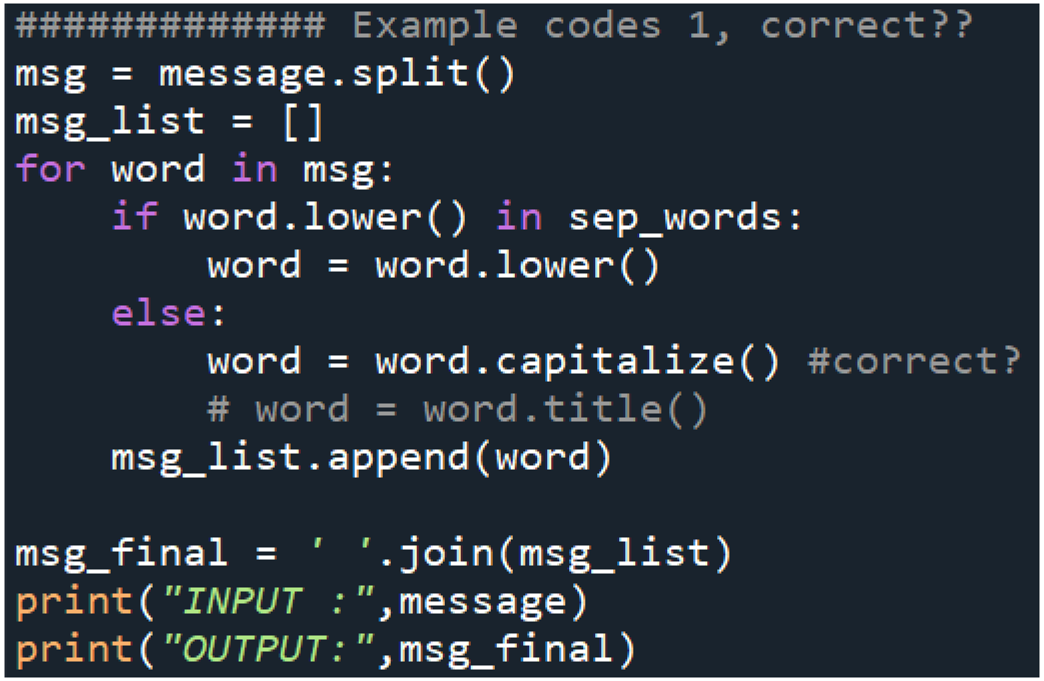
Capitalized: Welcome to MSDM_5002 of Data-Driven Modeling for MSc Students Offered by Phys&Math Department in UST. We Will Continue to Learn NumPy.

- Find some build-in functions or library online??
- Write as many functions as possible by yourself??

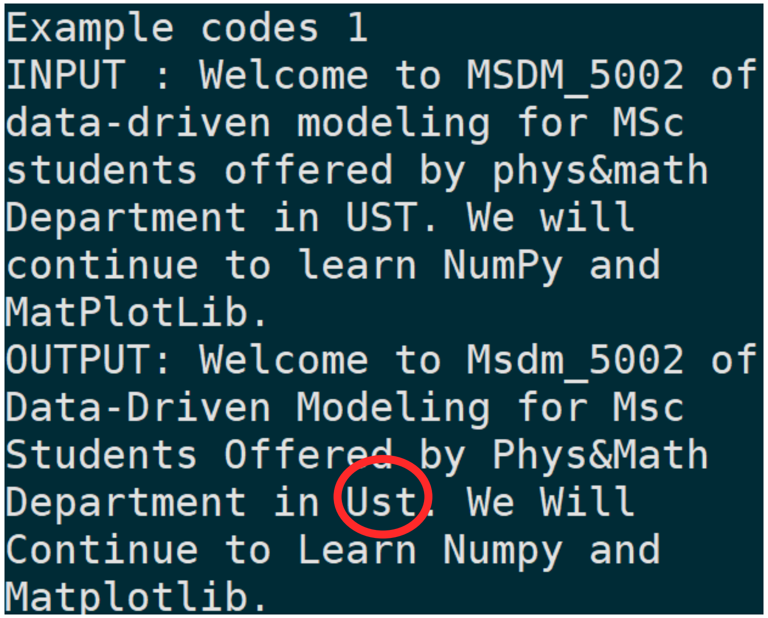
Example codes 1



```python
message = 'Welcome to MSDM_5002 of data-driven modeling \
for MSc students offered by phys&math Department in UST. \
We will continue to learn NumPy and MatPlotLib.'

message_correct='Welcome to MSDM_5002 of Data-Driven Modeling \
for MSc Students Offered by Phys&Math Department in UST. \
We Will Continue to Learn NumPy and MatPlotLib.'
```



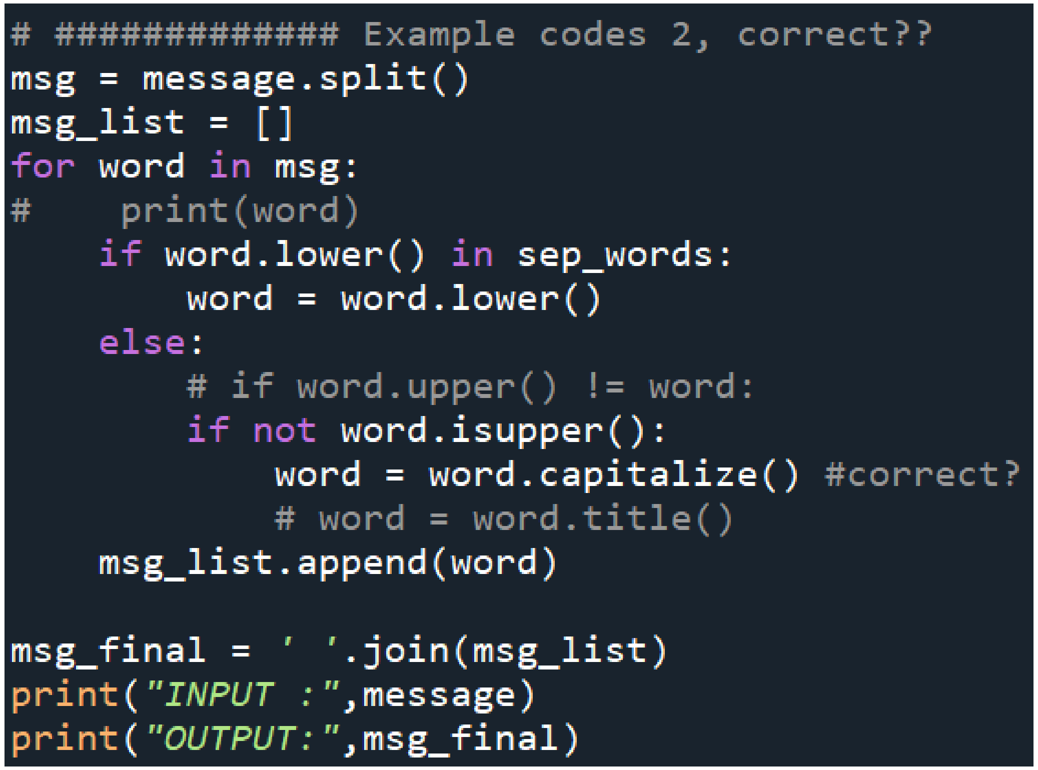
```python
############### Example codes 1, correct??
msg = message.split()
msg_list = []
for word in msg:
    if word.lower() in sep_words:
        word = word.lower()
    else:
        word = word.capitalize() #correct?
        # word = word.title()
    msg_list.append(word)

msg_final = ' '.join(msg_list)
print("INPUT :",message)
print("OUTPUT:",msg_final)
```



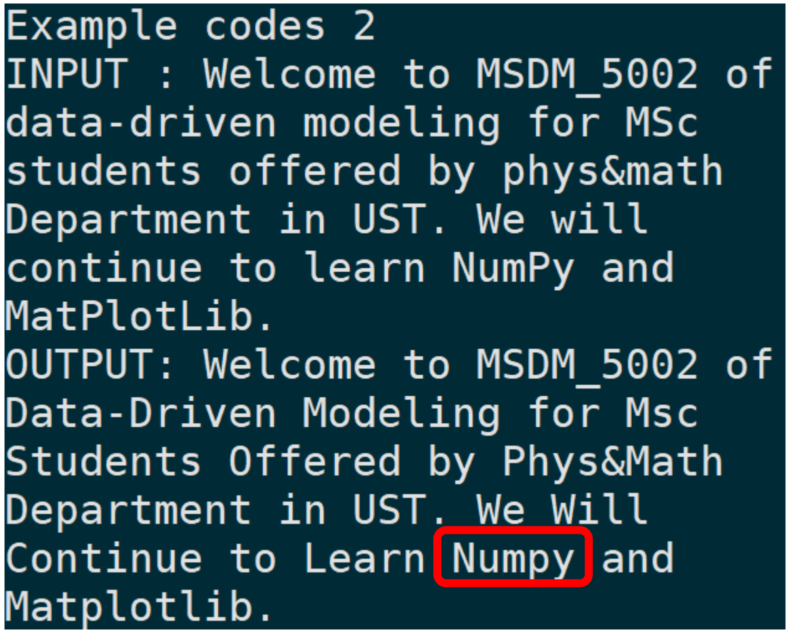
```
Example codes 1
INPUT : Welcome to MSDM_5002 of
data-driven modeling for MSc
students offered by phys&math
Department in UST. We will
continue to learn NumPy and
MatPlotLib.
OUTPUT: Welcome to Msdm_5002 of
Data-Driven Modeling for Msc
Students Offered by Phys&Math
Department in Ust. We Will
Continue to Learn Numpy and
Matplotlib.
```

- It does not work for words of all capitals with special meaning like UST.

Example codes 2



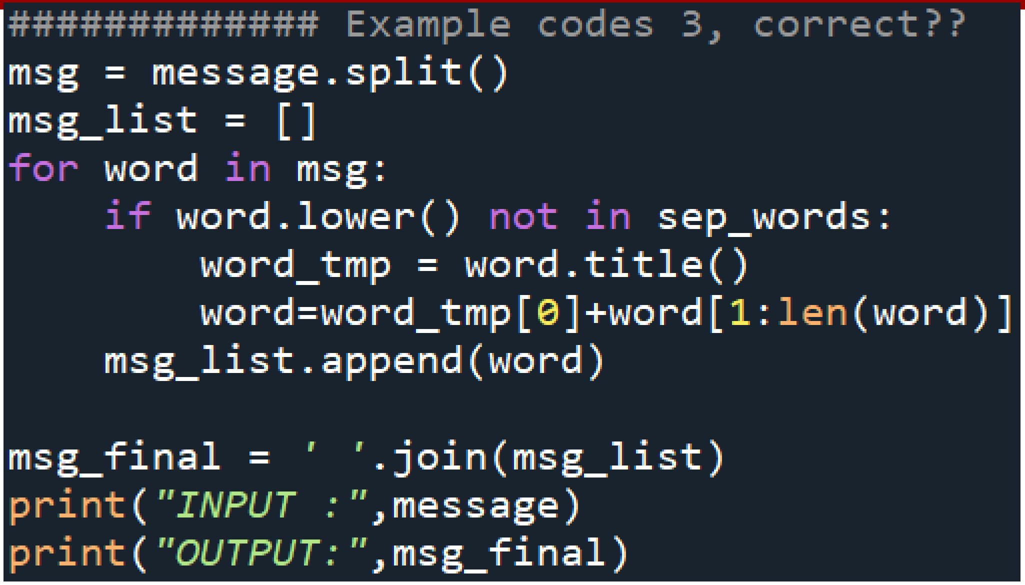
```
# ############# Example codes 2, correct??
msg = message.split()
msg_list = []
for word in msg:
#    print(word)
    if word.lower() in sep_words:
        word = word.lower()
    else:
        # if word.upper() != word:
        if not word.isupper():
            word = word.capitalize() #correct?
            # word = word.title()
    msg_list.append(word)

msg_final = ' '.join(msg_list)
print("INPUT :",message)
print("OUTPUT:",msg_final)
```



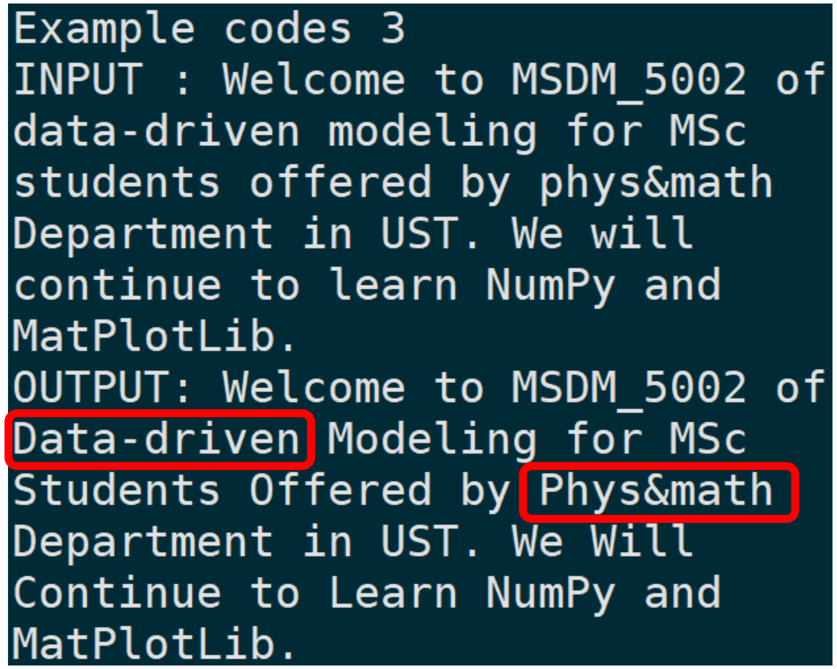
```
Example codes 2
INPUT : Welcome to MSDM_5002 of
data-driven modeling for MSc
students offered by phys&math
Department in UST. We will
continue to learn NumPy and
MatPlotLib.
OUTPUT: Welcome to MSDM_5002 of
Data-Driven Modeling for Msc
Students Offered by Phys&Math
Department in UST. We Will
Continue to Learn Numpy and
Matplotlib.
```

- It does not work for alternating capitals like NumPy.

Example codes 3



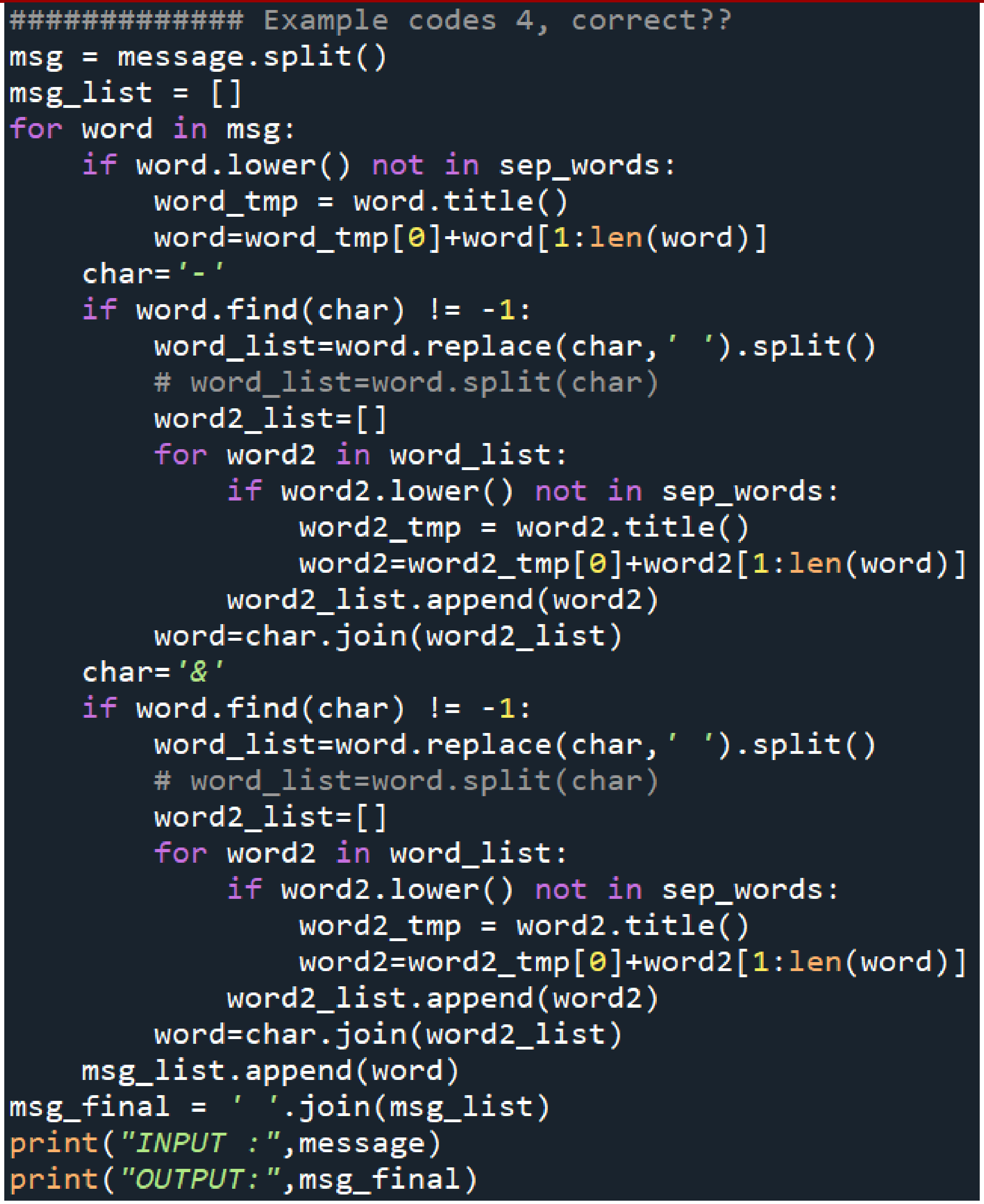
```
############### Example codes 3, correct??
msg = message.split()
msg_list = []
for word in msg:
    if word.lower() not in sep_words:
        word_tmp = word.title()
        word=word_tmp[0]+word[1:len(word)]
    msg_list.append(word)

msg_final = ' '.join(msg_list)
print("INPUT :",message)
print("OUTPUT:",msg_final)
```



```
Example codes 3
INPUT : Welcome to MSDM_5002 of
data-driven modeling for MSc
students offered by phys&math
Department in UST. We will
continue to learn NumPy and
MatPlotLib.
OUTPUT: Welcome to MSDM_5002 of
Data-driven Modeling for MSc
Students Offered by Phys&math
Department in UST. We Will
Continue to Learn NumPy and
MatPlotLib.
```

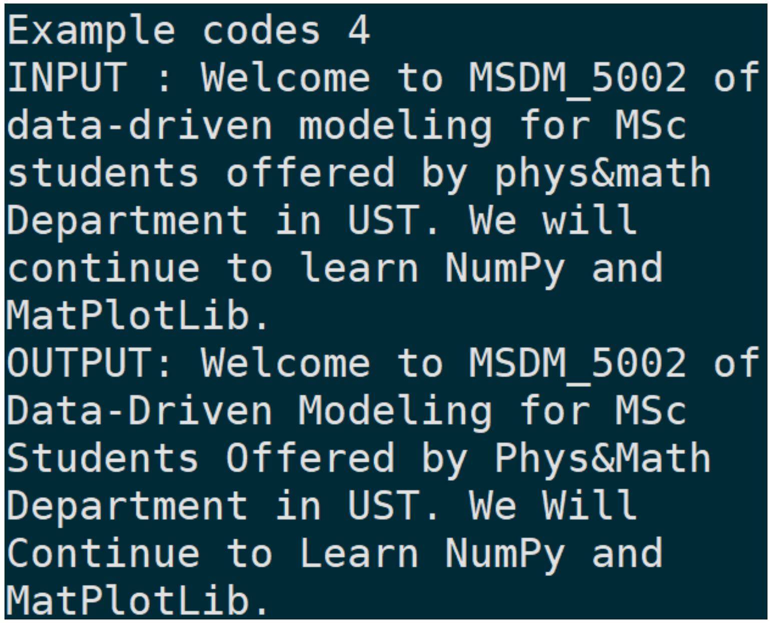
- It does not work for different words connected by special characters like data-driven and math&phys.

Example codes 4: new problems → old ones



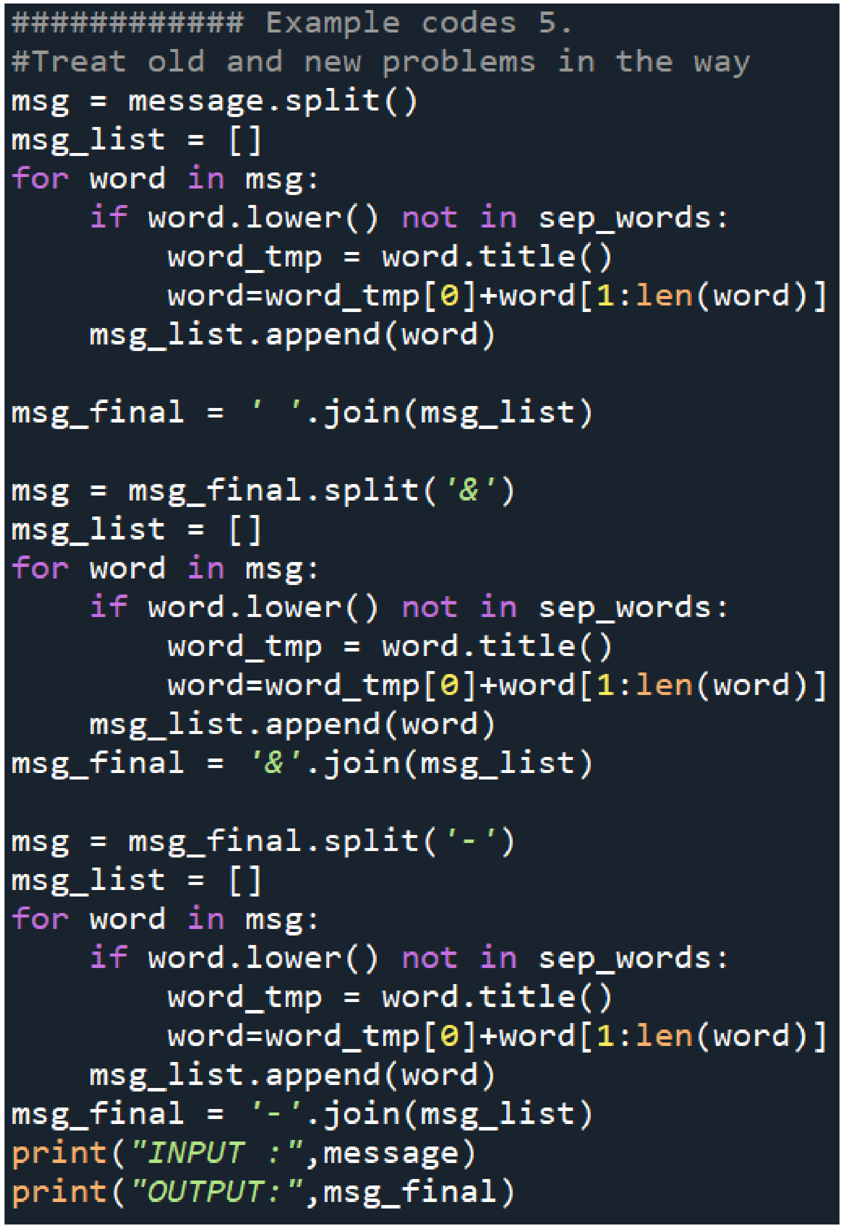
```python
############## Example codes 4, correct??
msg = message.split()
msg_list = []
for word in msg:
    if word.lower() not in sep_words:
        word_tmp = word.title()
        word=word_tmp[0]+word[1:len(word)]
    char='-'
    if word.find(char) != -1:
        word_list=word.replace(char,' ').split()
        # word_list=word.split(char)
        word2_list=[]
        for word2 in word_list:
            if word2.lower() not in sep_words:
                word2_tmp = word2.title()
                word2=word2_tmp[0]+word2[1:len(word)]
            word2_list.append(word2)
        word=char.join(word2_list)
    char='&'
    if word.find(char) != -1:
        word_list=word.replace(char,' ').split()
        # word_list=word.split(char)
        word2_list=[]
        for word2 in word_list:
            if word2.lower() not in sep_words:
                word2_tmp = word2.title()
                word2=word2_tmp[0]+word2[1:len(word)]
            word2_list.append(word2)
        word=char.join(word2_list)
    msg_list.append(word)
msg_final = ' '.join(msg_list)
print("INPUT :",message)
print("OUTPUT:",msg_final)
```

- We take a word as the message and take special characters as splitter. [**Converting new problems to old ones.**]

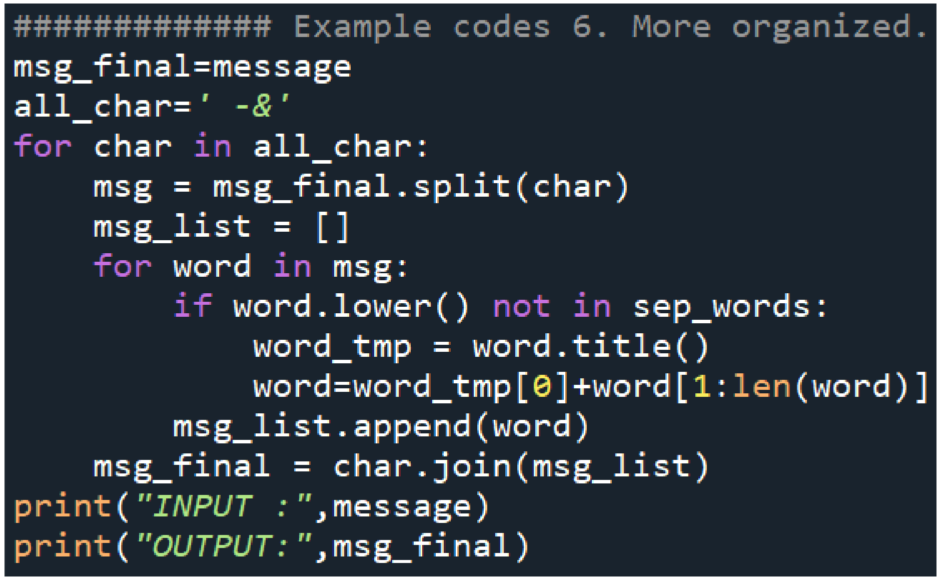


```
Example codes 4
INPUT : Welcome to MSDM_5002 of
data-driven modeling for MSc
students offered by phys&math
Department in UST. We will
continue to learn NumPy and
MatPlotLib.
OUTPUT: Welcome to MSDM_5002 of
Data-Driven Modeling for MSc
Students Offered by Phys&Math
Department in UST. We Will
Continue to Learn NumPy and
MatPlotLib.
```

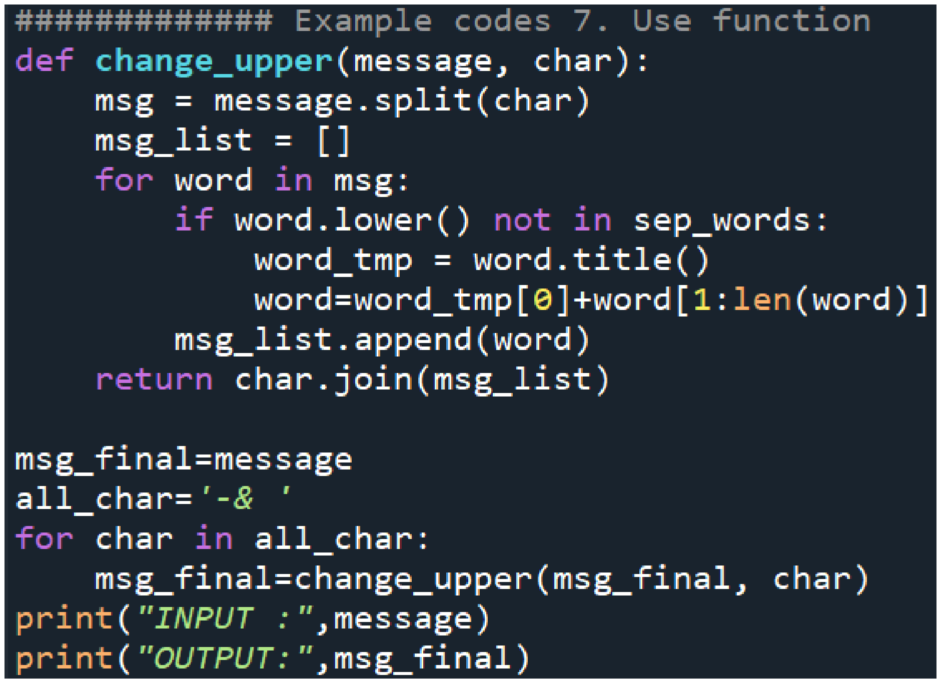
Example codes 5, 6 & 7: organize the code



```python
############# Example codes 5.
#Treat old and new problems in the way
msg = message.split()
msg_list = []
for word in msg:
    if word.lower() not in sep_words:
        word_tmp = word.title()
        word=word_tmp[0]+word[1:len(word)]
    msg_list.append(word)

msg_final = ' '.join(msg_list)

msg = msg_final.split('&')
msg_list = []
for word in msg:
    if word.lower() not in sep_words:
        word_tmp = word.title()
        word=word_tmp[0]+word[1:len(word)]
    msg_list.append(word)
msg_final = '&'.join(msg_list)

msg = msg_final.split('-')
msg_list = []
for word in msg:
    if word.lower() not in sep_words:
        word_tmp = word.title()
        word=word_tmp[0]+word[1:len(word)]
    msg_list.append(word)
msg_final = '-'.join(msg_list)
print("INPUT :",message)
print("OUTPUT:",msg_final)
```

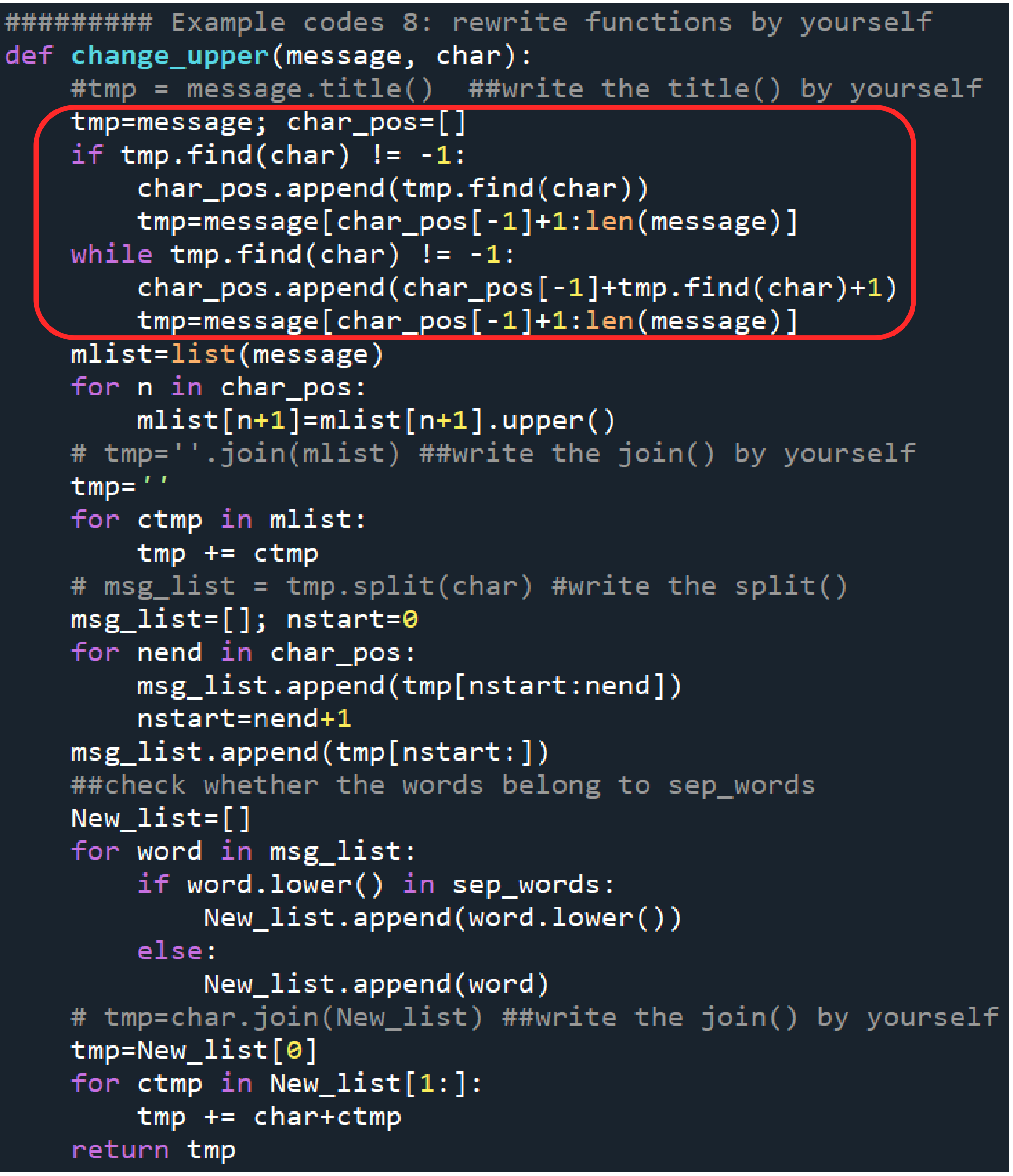


```python
############# Example codes 6. More organized.
msg_final=message
all_char=' -&'
for char in all_char:
    msg = msg_final.split(char)
    msg_list = []
    for word in msg:
        if word.lower() not in sep_words:
            word_tmp = word.title()
            word=word_tmp[0]+word[1:len(word)]
        msg_list.append(word)
    msg_final = char.join(msg_list)
print("INPUT :",message)
print("OUTPUT:",msg_final)
```

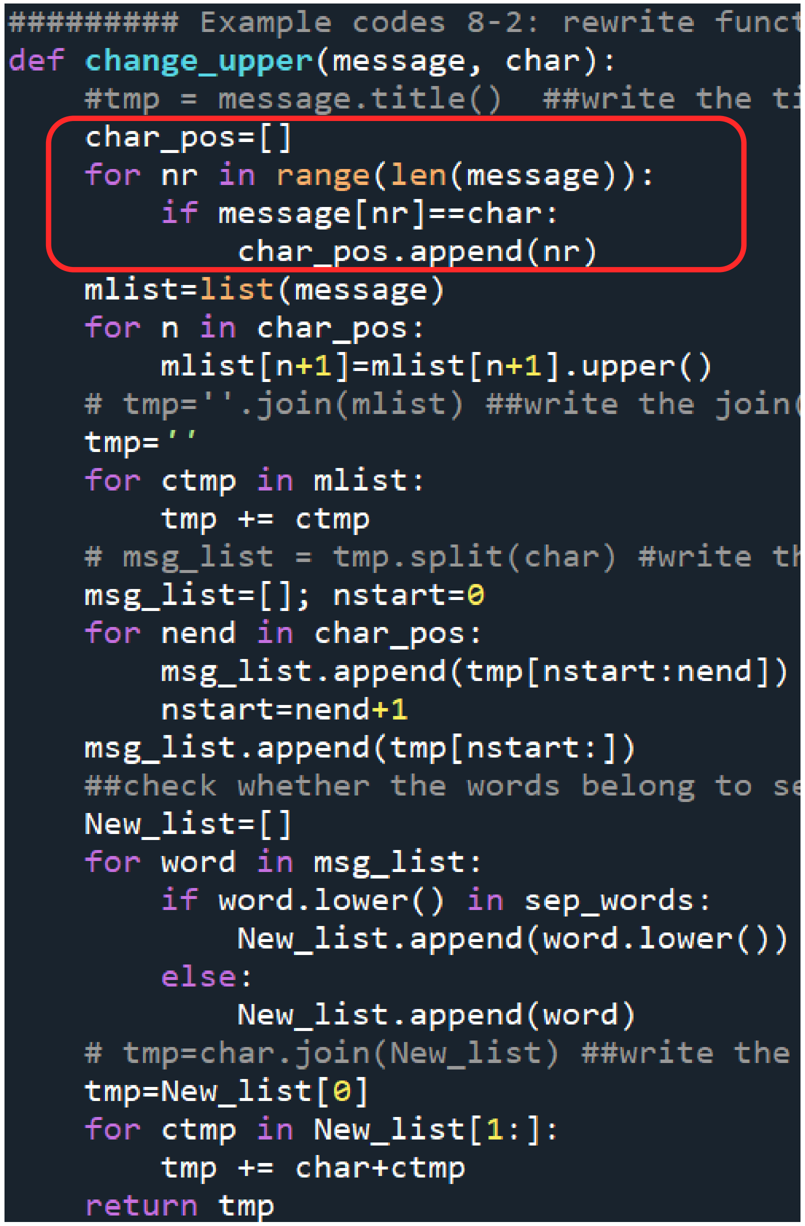


```python
############# Example codes 7. Use function
def change_upper(message, char):
    msg = message.split(char)
    msg_list = []
    for word in msg:
        if word.lower() not in sep_words:
            word_tmp = word.title()
            word=word_tmp[0]+word[1:len(word)]
        msg_list.append(word)
    return char.join(msg_list)

msg_final=message
all_char='-& '
for char in all_char:
    msg_final=change_upper(msg_final, char)
print("INPUT :",message)
print("OUTPUT:",msg_final)
```

Example codes 8: home-made codes

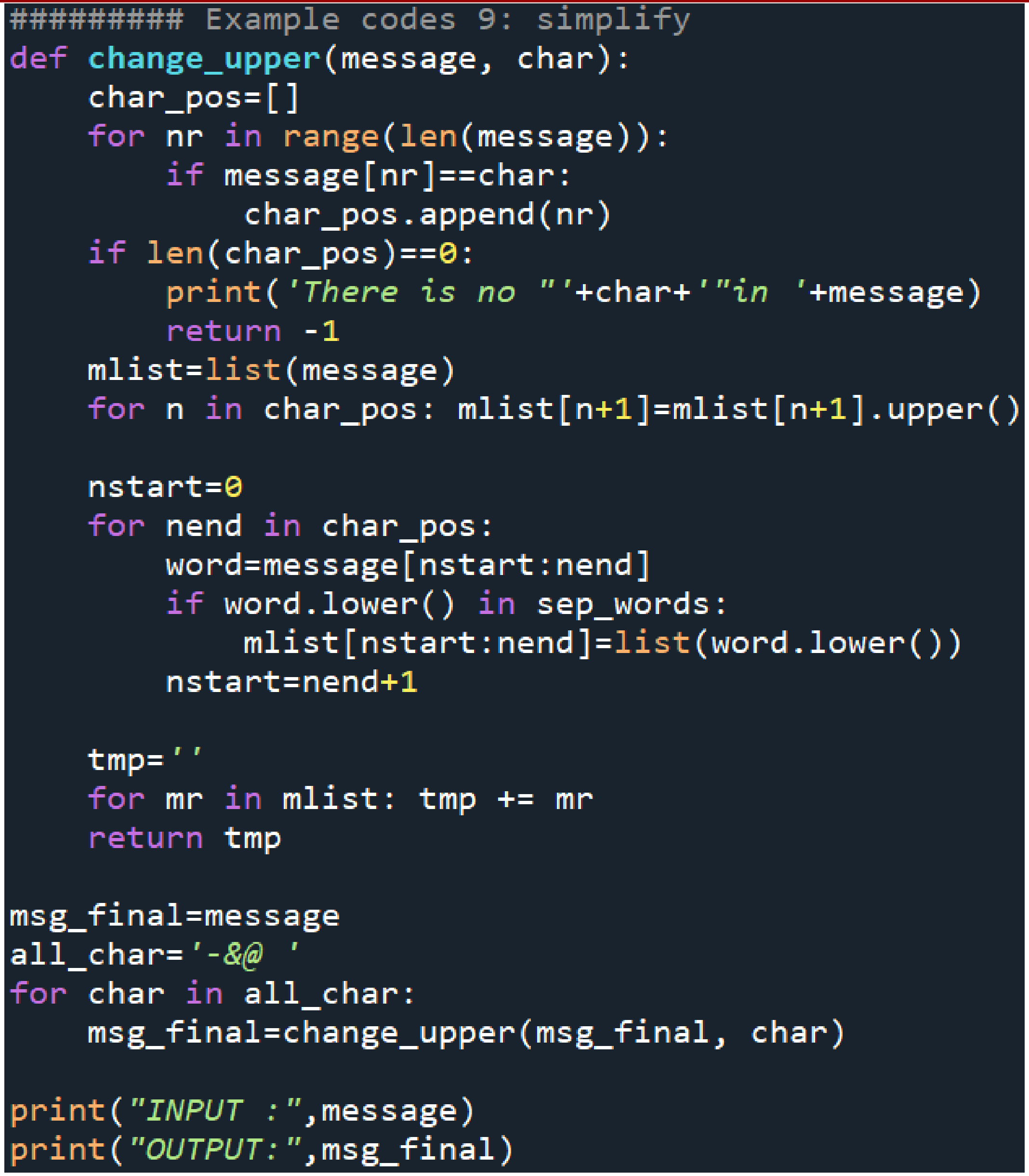


```
######### Example codes 8: rewrite functions by yourself
def change_upper(message, char):
    #tmp = message.title()  ##write the title() by yourself
    tmp=message; char_pos=[]
    if tmp.find(char) != -1:
        char_pos.append(tmp.find(char))
        tmp=message[char_pos[-1]+1:len(message)]
    while tmp.find(char) != -1:
        char_pos.append(char_pos[-1]+tmp.find(char)+1)
        tmp=message[char_pos[-1]+1:len(message)]
    mlist=list(message)
    for n in char_pos:
        mlist[n+1]=mlist[n+1].upper()
    # tmp=''.join(mlist) ##write the join() by yourself
    tmp=''
    for ctmp in mlist:
        tmp += ctmp
    # msg_list = tmp.split(char) #write the split()
    msg_list=[]; nstart=0
    for nend in char_pos:
        msg_list.append(tmp[nstart:nend])
        nstart=nend+1
    msg_list.append(tmp[nstart:])
    ##check whether the words belong to sep_words
    New_list=[]
    for word in msg_list:
        if word.lower() in sep_words:
            New_list.append(word.lower())
        else:
            New_list.append(word)
    # tmp=char.join(New_list) ##write the join() by yourself
    tmp=New_list[0]
    for ctmp in New_list[1:]:
        tmp += char+ctmp
    return tmp
```



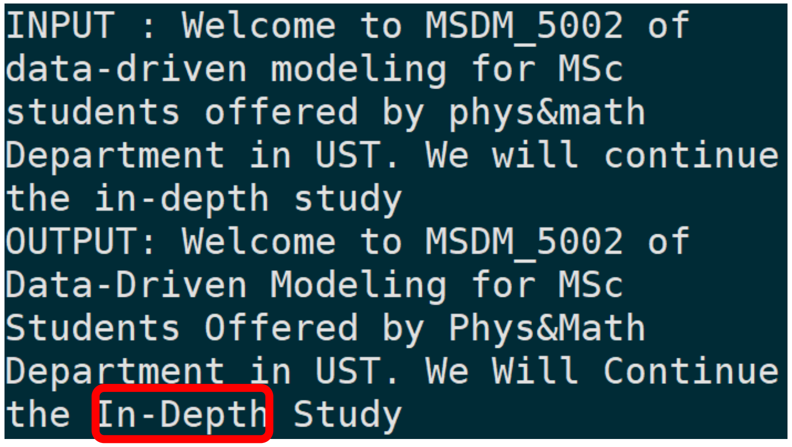
```
######### Example codes 8-2: rewrite funct
def change_upper(message, char):
    #tmp = message.title()  ##write the ti
    char_pos=[]
    for nr in range(len(message)):
        if message[nr]==char:
            char_pos.append(nr)
    mlist=list(message)
    for n in char_pos:
        mlist[n+1]=mlist[n+1].upper()
    # tmp=''.join(mlist) ##write the join(
    tmp=''
    for ctmp in mlist:
        tmp += ctmp
    # msg_list = tmp.split(char) #write th
    msg_list=[]; nstart=0
    for nend in char_pos:
        msg_list.append(tmp[nstart:nend])
        nstart=nend+1
    msg_list.append(tmp[nstart:])
    ##check whether the words belong to se
    New_list=[]
    for word in msg_list:
        if word.lower() in sep_words:
            New_list.append(word.lower())
        else:
            New_list.append(word)
    # tmp=char.join(New_list) ##write the
    tmp=New_list[0]
    for ctmp in New_list[1:]:
        tmp += char+ctmp
    return tmp
```

Example codes 9: simplification



```python
########## Example codes 9: simplify
def change_upper(message, char):
    char_pos=[]
    for nr in range(len(message)):
        if message[nr]==char:
            char_pos.append(nr)
    if len(char_pos)==0:
        print('There is no "'+char+'"in '+message)
        return -1
    mlist=list(message)
    for n in char_pos: mlist[n+1]=mlist[n+1].upper()

    nstart=0
    for nend in char_pos:
        word=message[nstart:nend]
        if word.lower() in sep_words:
            mlist[nstart:nend]=list(word.lower())
        nstart=nend+1

    tmp=''
    for mr in mlist: tmp += mr
    return tmp

msg_final=message
all_char='-&@ '
for char in all_char:
    msg_final=change_upper(msg_final, char)

print("INPUT :",message)
print("OUTPUT:",msg_final)
```

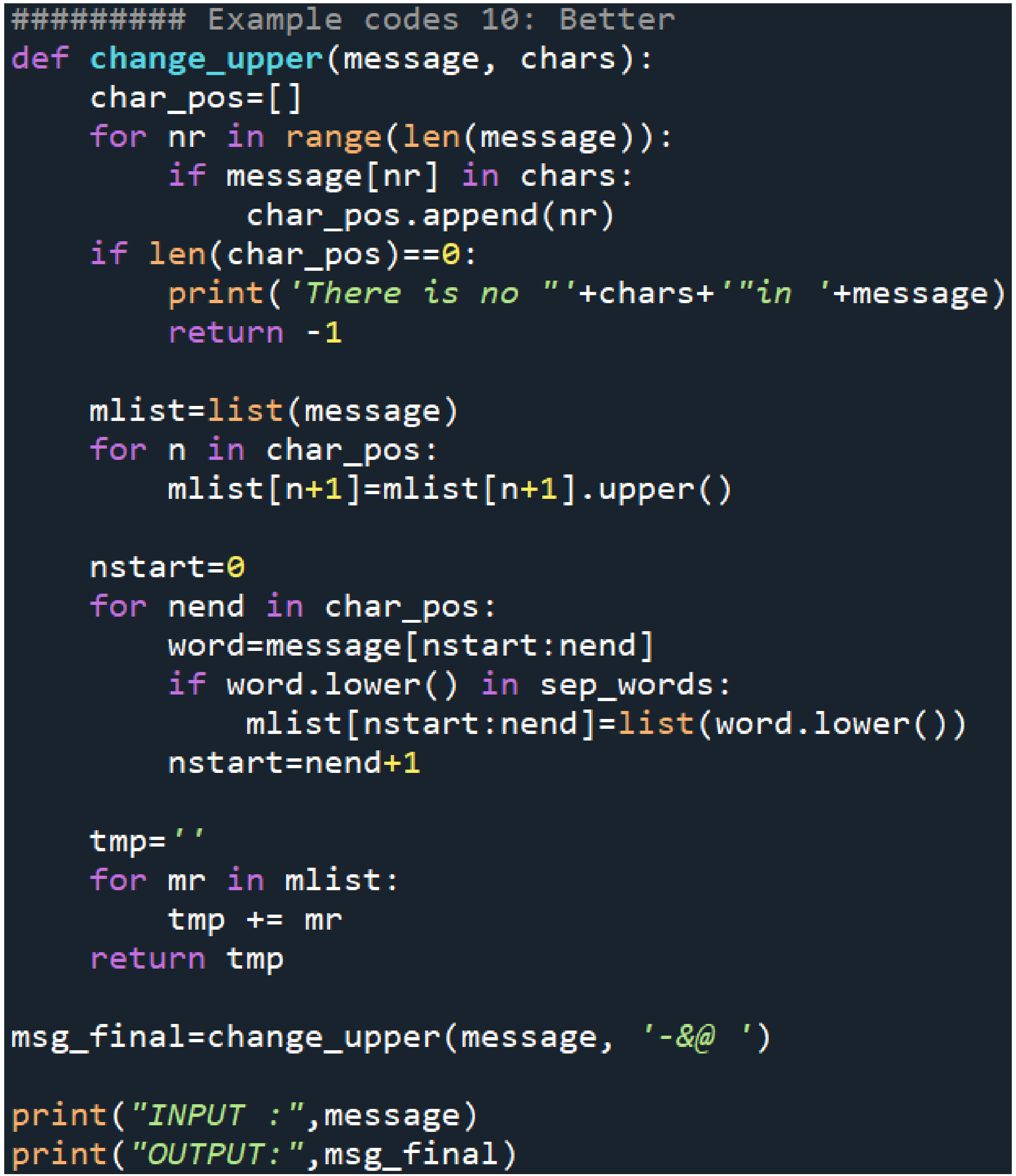
- Is it good enough?



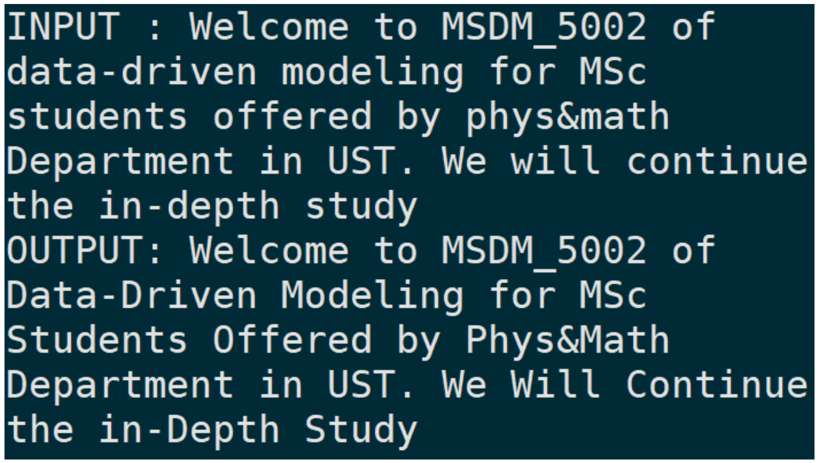
```
INPUT : Welcome to MSDM_5002 of
data-driven modeling for MSc
students offered by phys&math
Department in UST. We will continue
the in-depth study
OUTPUT: Welcome to MSDM_5002 of
Data-Driven Modeling for MSc
Students Offered by Phys&Math
Department in UST. We Will Continue
the In-Depth Study
```

- It does not work for very special cases.

Example codes 10: tailor-made function



```python
########## Example codes 10: Better
def change_upper(message, chars):
    char_pos=[]
    for nr in range(len(message)):
        if message[nr] in chars:
            char_pos.append(nr)
    if len(char_pos)==0:
        print('There is no "'+chars+'"in '+message)
        return -1

    mlist=list(message)
    for n in char_pos:
        mlist[n+1]=mlist[n+1].upper()

    nstart=0
    for nend in char_pos:
        word=message[nstart:nend]
        if word.lower() in sep_words:
            mlist[nstart:nend]=list(word.lower())
        nstart=nend+1

    tmp=''
    for mr in mlist:
        tmp += mr
    return tmp

msg_final=change_upper(message, '-&@ ')

print("INPUT :",message)
print("OUTPUT:",msg_final)
```



```
INPUT  : Welcome to MSDM_5002 of
data-driven modeling for MSc
students offered by phys&math
Department in UST. We will continue
the in-depth study
OUTPUT: Welcome to MSDM_5002 of
Data-Driven Modeling for MSc
Students Offered by Phys&Math
Department in UST. We Will Continue
the in-Depth Study
```

- To learn coding, you will have to keep trying!

3D plot in Python

- Matplotlib was initially designed with only 2D plotting in mind. Now, some 3D plotting utilities were built on top of Matplotlib's 2D display, and the result is a convenient set of tools for three-dimensional data visualization. 3D plots are enabled by importing the **mplot3d toolkit**, included with the main Matplotlib installation.
- We can import the mplot3d in the following way

from mpl_toolkits import mplot3d

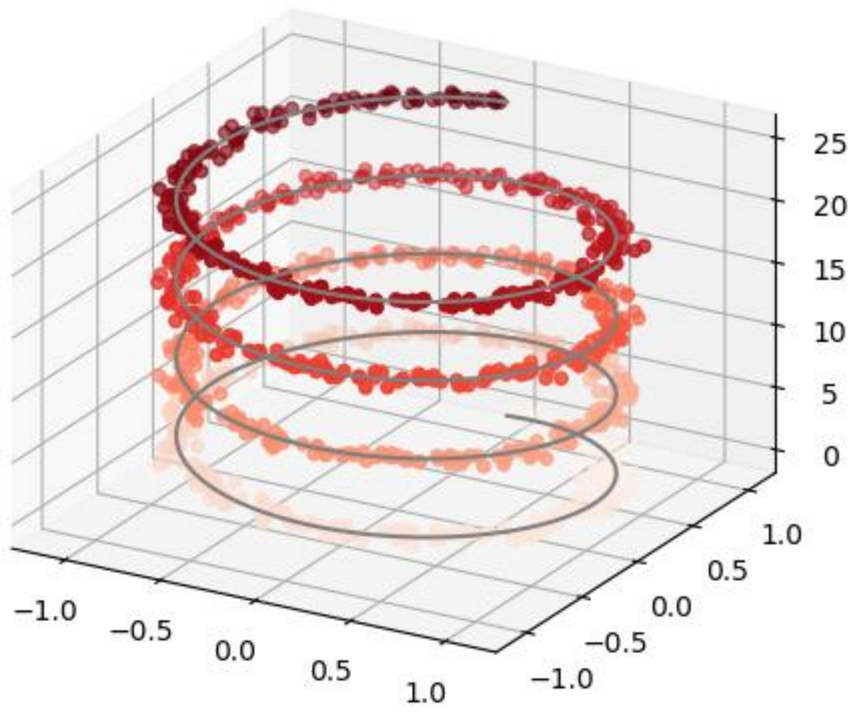
- Once this submodule is imported, a three-dimensional axes can be created by passing the keyword projection='3d' to any of the normal axes creation routines:

fig = plt.figure()
ax = plt.axes(projection='3d')

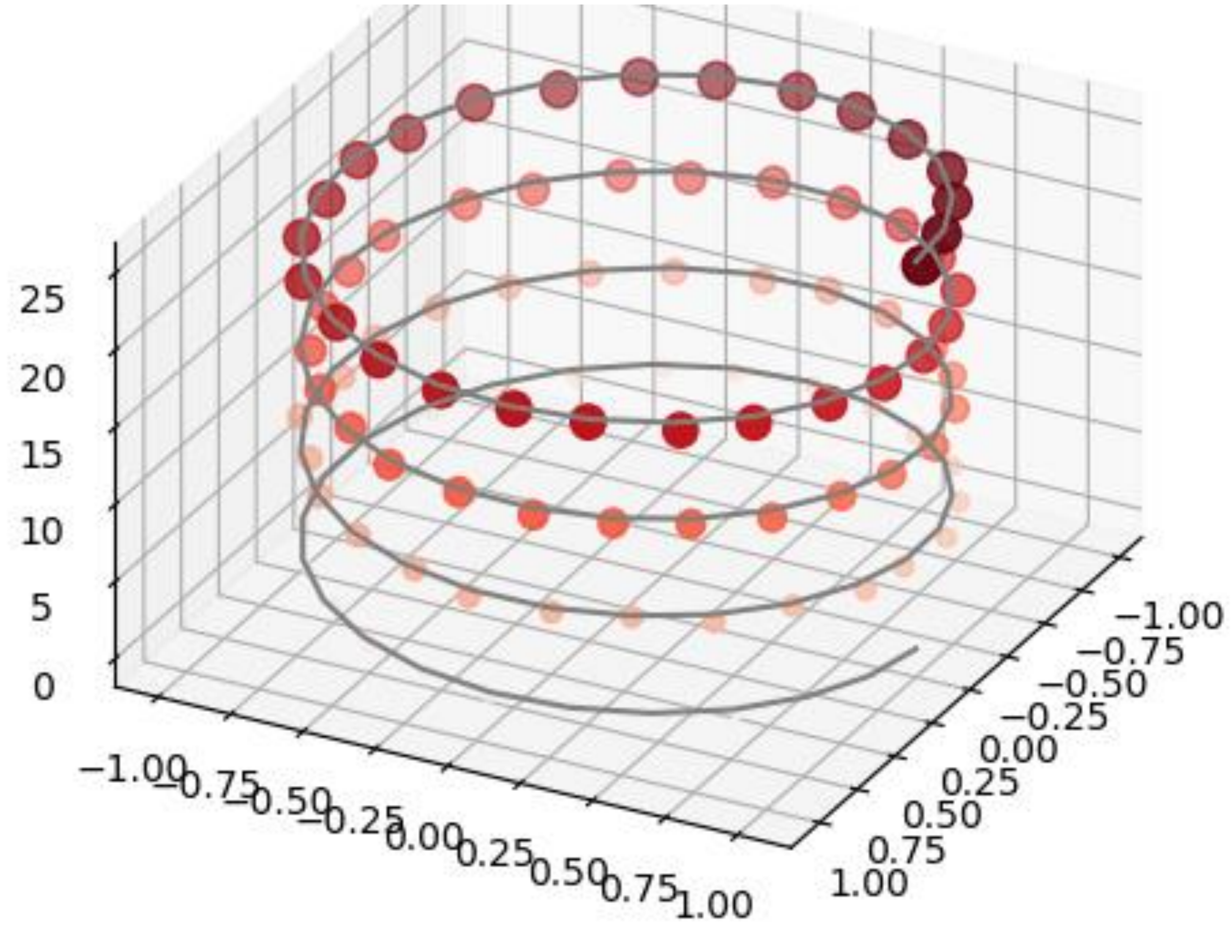
- In 3D, we use **ax.set_box_aspect(x,y,z)** to set the aspect ratio for x, y and z axes.

3D Points and Lines

```
from mpl_toolkits import mplot3d

import numpy as np
import matplotlib.pyplot as plt

ax = plt.axes(projection='3d')

# Data for a three-dimensional line
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'gray')

# Data for three-dimensional scattered points
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter3D(xdata, ydata, zdata, s=2, c=zdata, cmap='Reds')
```



Use keyword “s=xxx” to change the dot size

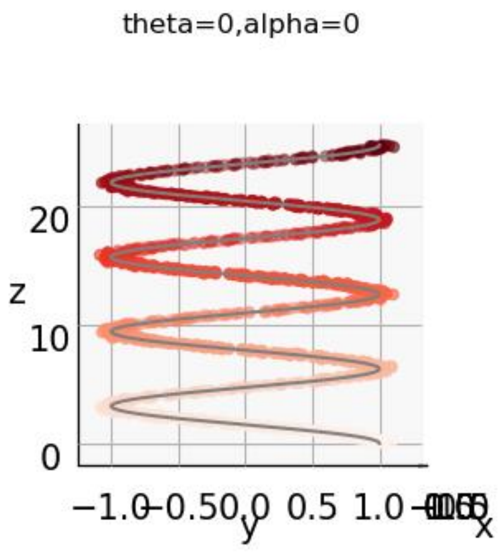


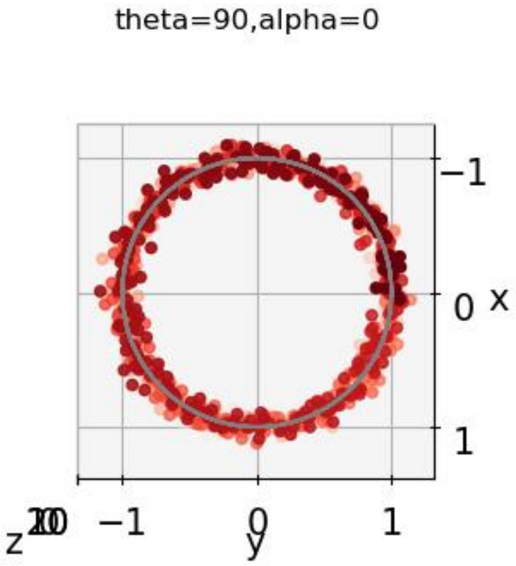
Some keywords

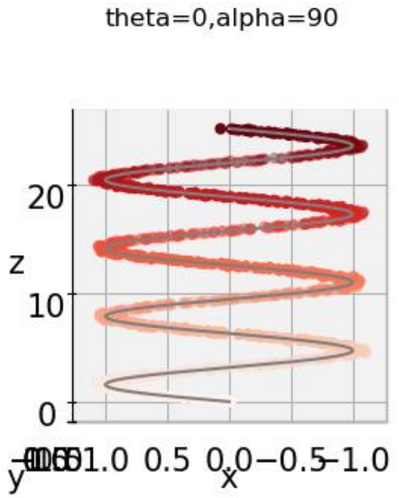
Keywords	Description
xs, ys	Positions of data points.
zs	Either an array of the same length as xs and ys or a single value to place all points in the same plane. Default is 0.
zdir	Which direction to use as z (‘x’, ‘y’ or ‘z’) when plotting a 2D set.
s	Size in points^2. It is a scalar or an array of the same length as x and y.
c	A color. c can be a single color format string, or a sequence of color specifications of length N, or a sequence of N numbers to be mapped to colors using the cmap and norm specified via kwargs (see below). Note that c should not be a single numeric RGB or RGBA sequence because that is indistinguishable from an array of values to be colormapped. c can be a 2-D array in which the rows are RGB or RGBA, however, including the case of a single row to specify the same color for all points.
depthshade	Whether or not to shade the scatter markers to give the appearance of depth. Default is True.

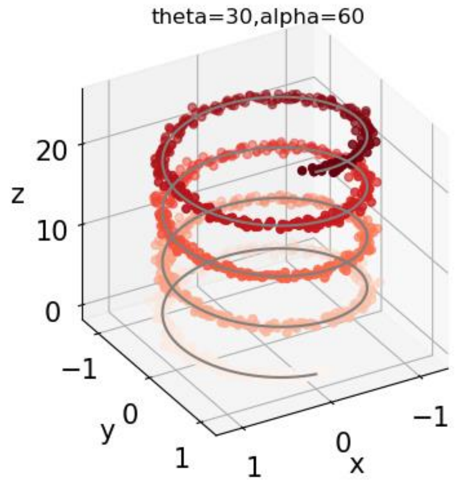
Change the viewpoint

- You can change the viewpoint by using function ***view_init(theta, alpha)***





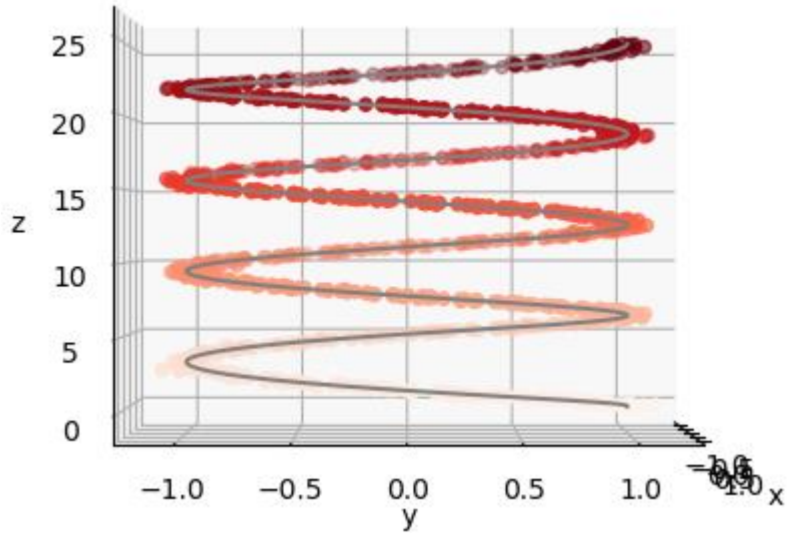




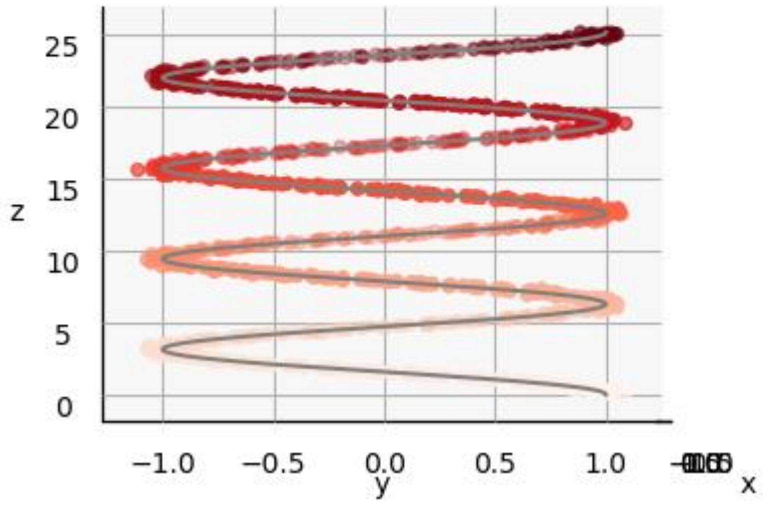
Perspective or Orthogonal projection

- You can change the projection to be Perspective or Orthogonal type using the function **set_proj_type**('ortho','persp')
- You need to use **draw**() to replot the figure after you reset your projection type.

Perspective



Orthogonal



Set the ticks

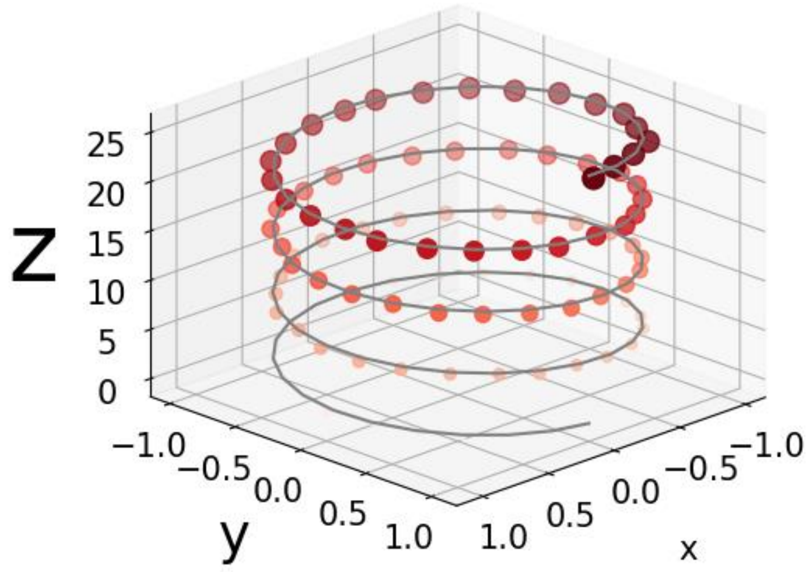
- You can also set the ticks for all x, y and z axis like 2D plot

xaxis.set_tick_params()
yaxis.set_tick_params()
zaxis.set_tick_params()

- The allowed parameters can be found in https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.axes.Axes.tick_params.html#matplotlib.axes.Axes.tick_params

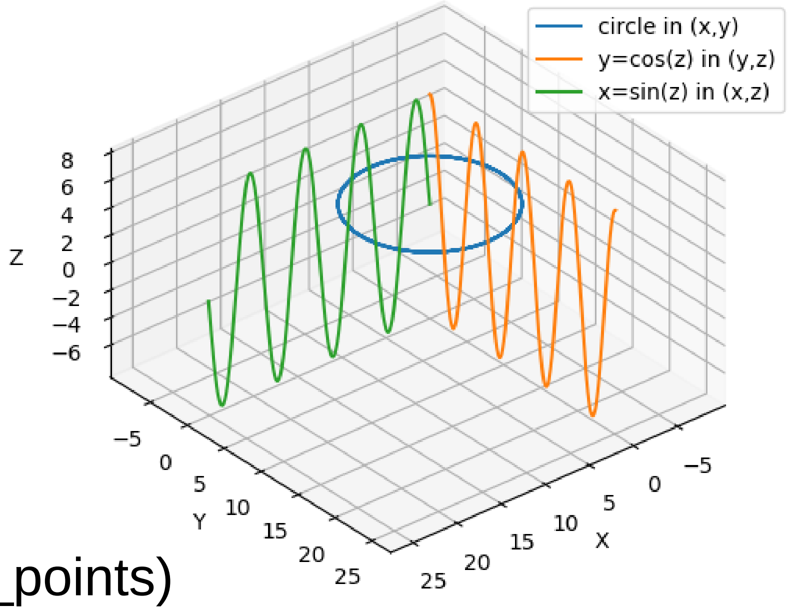
- Use keyword ***labelpad*** to control the distance between the label and axis.

```
ax.set_xlabel('x',fontsize=15,labelpad=20)
ax.set_ylabel('y',fontsize=25,labelpad=15)
ax.set_zlabel('z',fontsize=45,labelpad=10)
ax.xaxis.set_tick_params(labelsize=15)
ax.yaxis.set_tick_params(labelsize=15)
ax.zaxis.set_tick_params(labelsize=15)
```

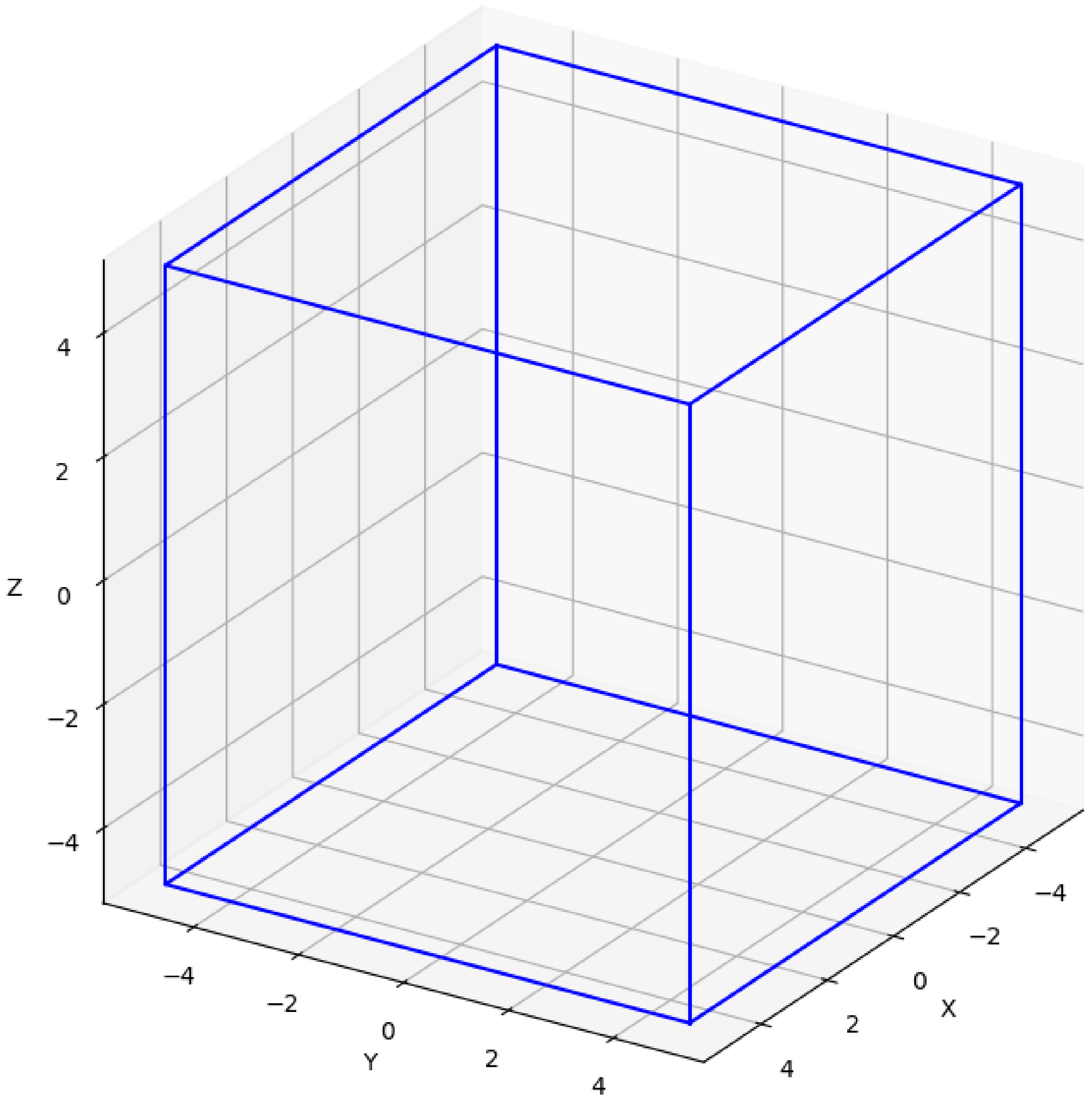


plot 2D data in 3D

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

ax = plt.axes(projection='3d',proj_type='ortho')

Num_period=8; Num_points=1000;

zline = np.linspace(0, np.pi*Num_period, Num_points)
xline = np.sin(zline)*Num_period
yline = np.cos(zline)*Num_period
ax.plot(xline, yline, zdir='z', label='circle in (x,y)')
ax.plot(zline, yline, zdir='x', label='y=cos(z) in (y,z)')
ax.plot(zline, xline, zdir='y', label='x=sin(z) in (x,z)')

ax.legend()
ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Z')

theta=45; alpha=50; ax.view_init(theta, alpha); plt.draw()
```



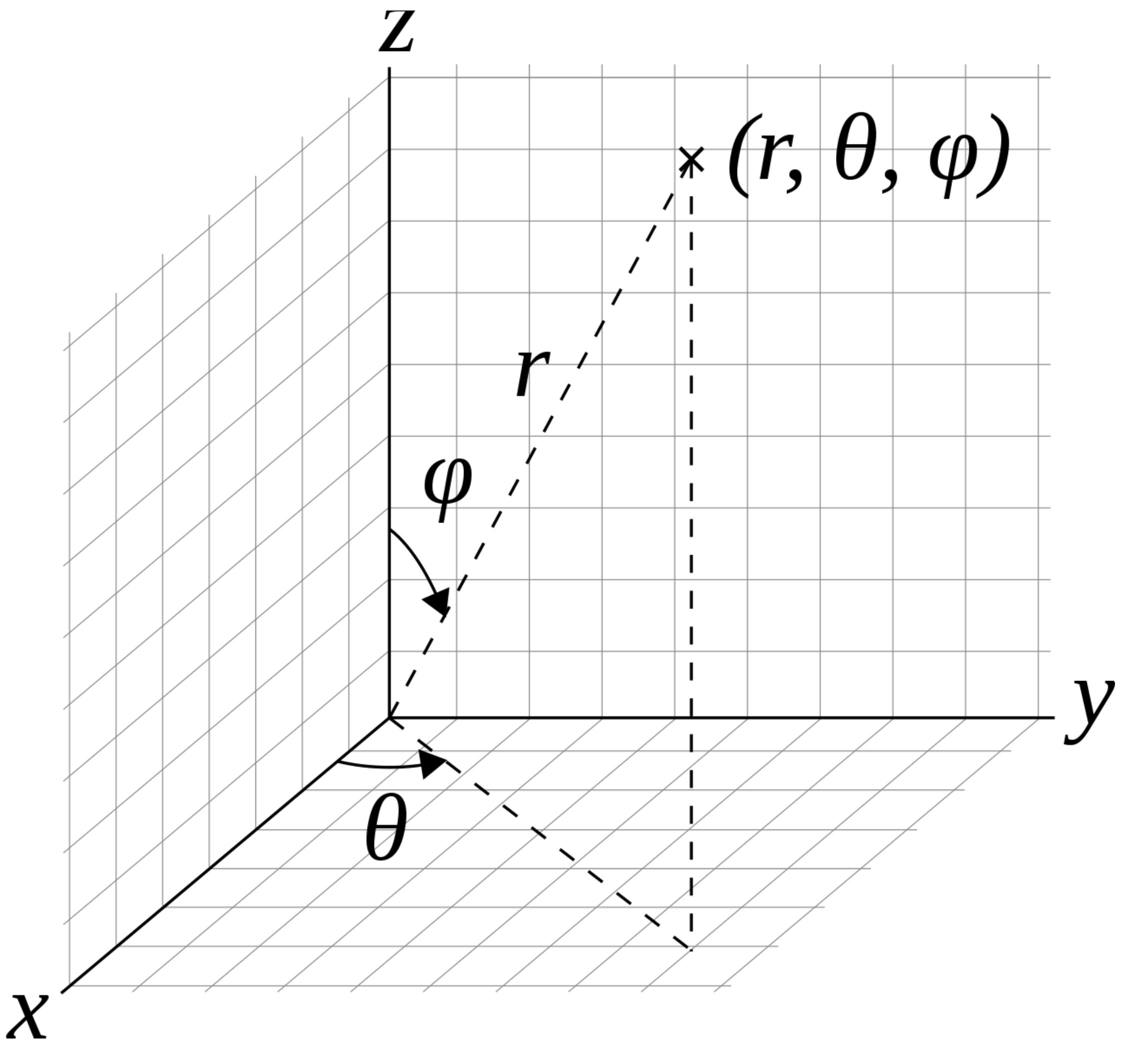
Practice: plot a cube



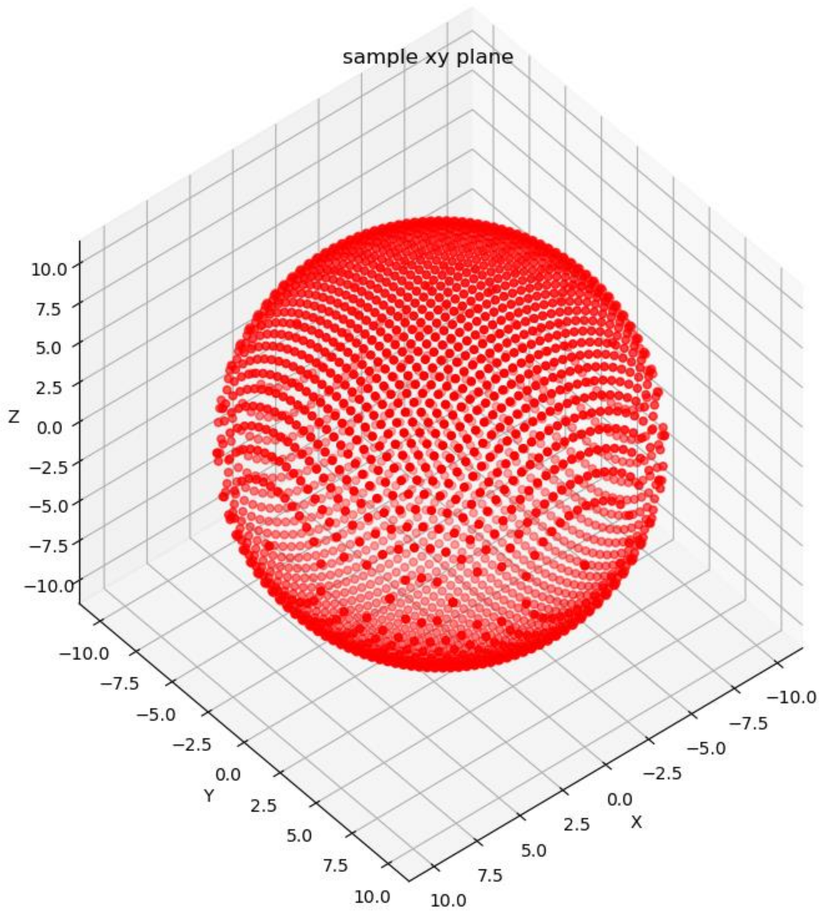
Spherical coordinate system

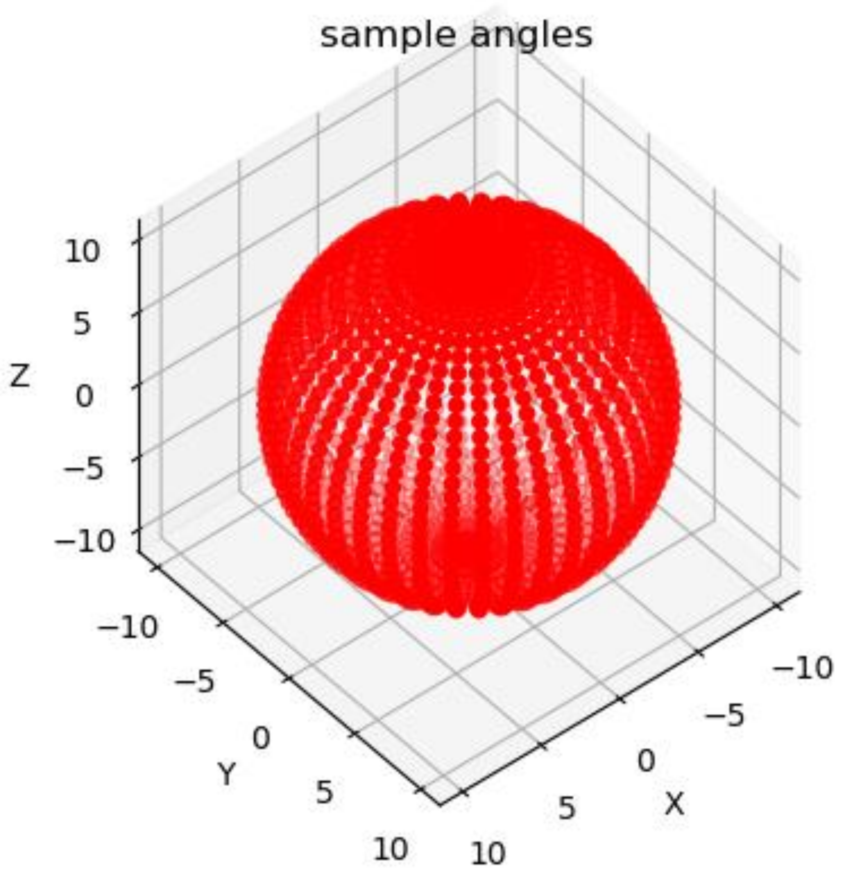
Relation between **Cartesian coordinates** and **Spherical coordinates**

- $z = r\cos(\phi)$
- $x = r\sin(\phi)\cos(\theta)$
- $y = r\sin(\phi)\sin(\theta)$



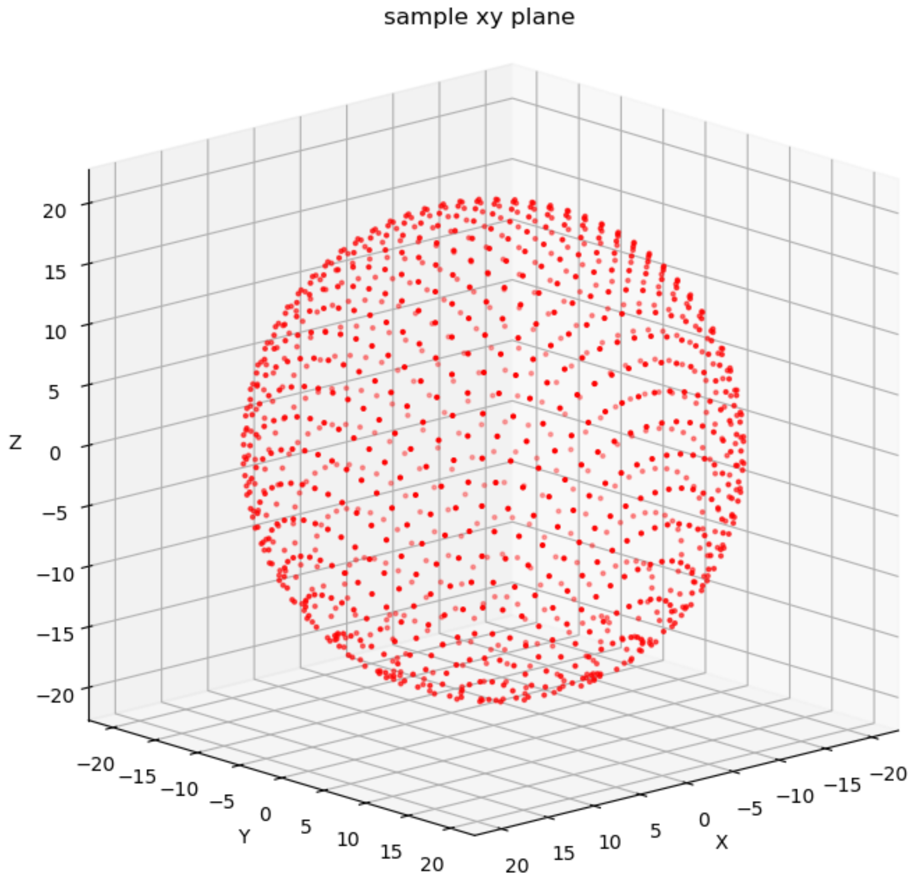
Plot a sphere

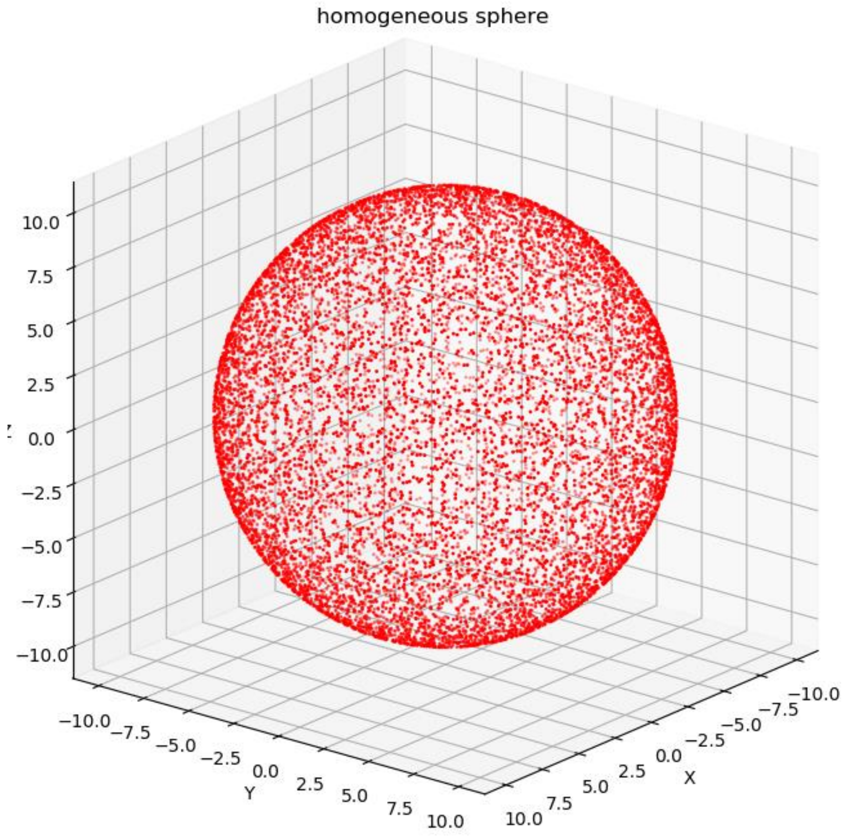




- How to plot a homogeneous sphere?

Homogeneous sphere

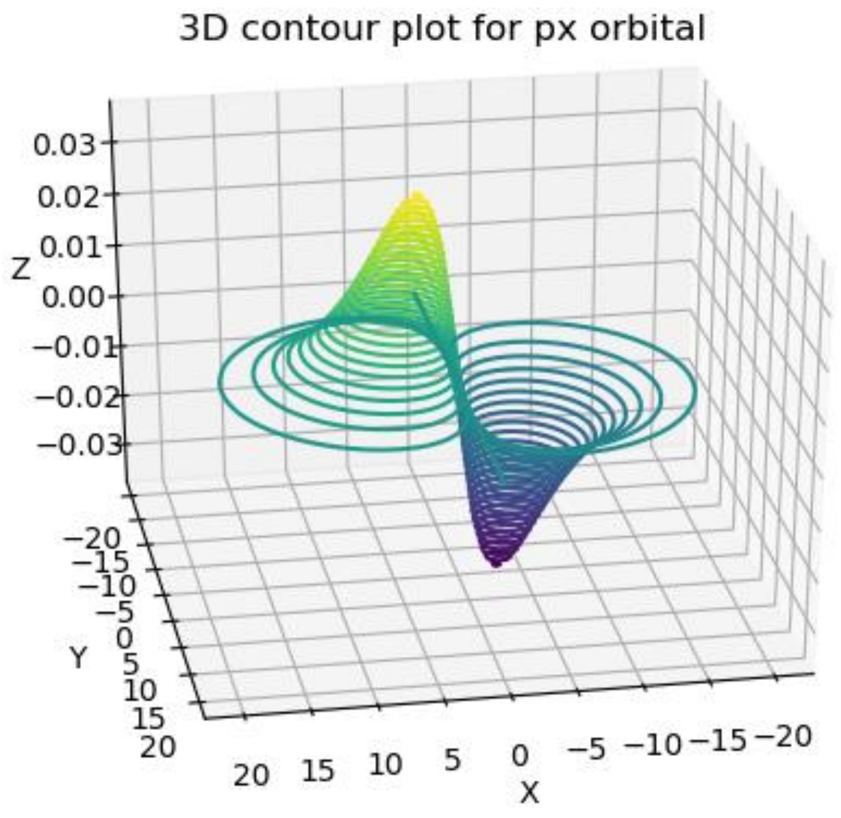




3D contour plots

- Analogous to the contour plots, mplot3d contains tools to create 3D relief plots using the same inputs. Like two-dimensional ax.contour plots, ax.contour3D requires all the input data to be in the form of 3D regular grids, with the Z data evaluated at each point.

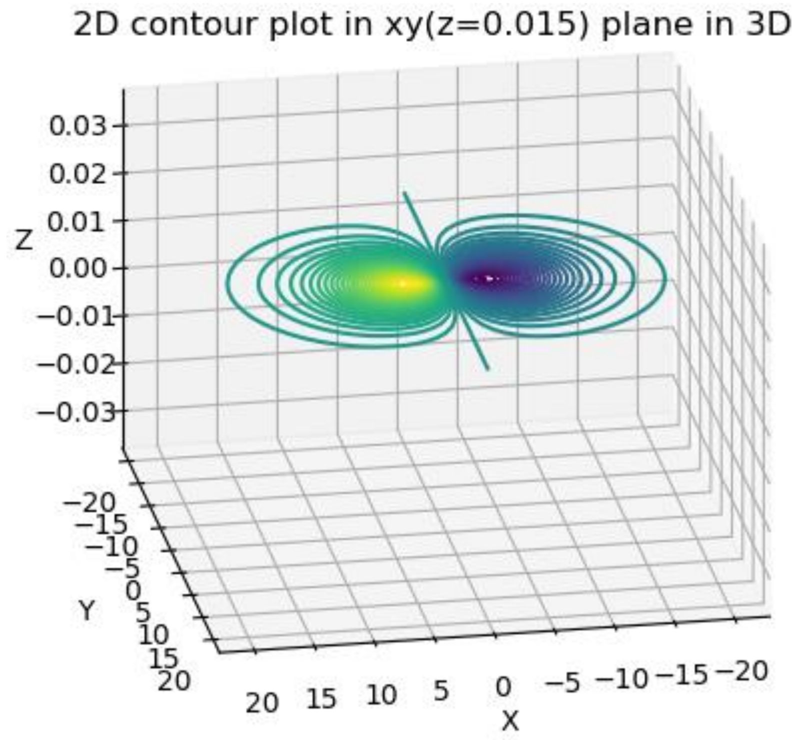
ax=plt.axes(projection='3d')
plt.title('3D controu plot')
ax.contour3D(xx,yy,px,50)

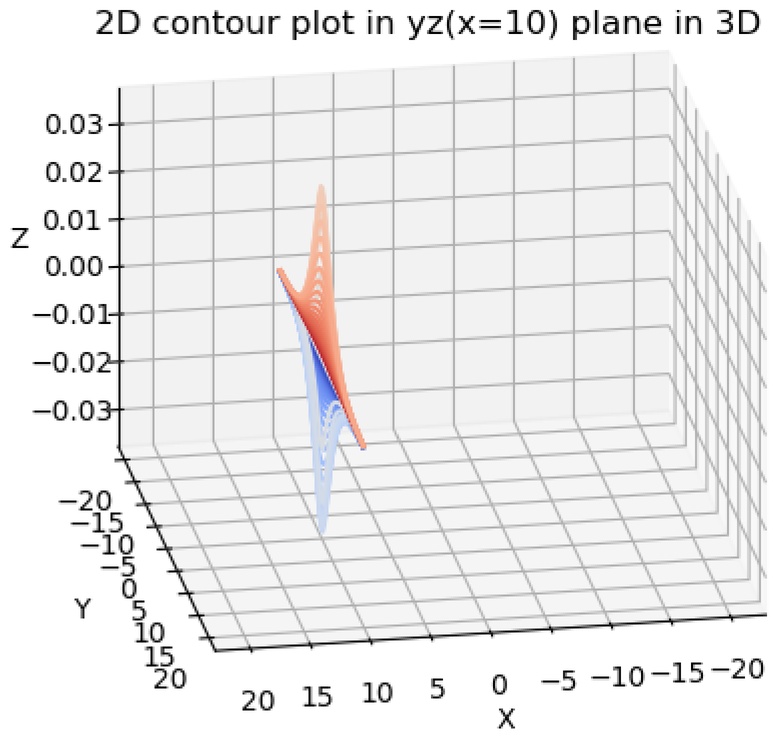


2D contour plot in 3D axe

- We can also realize 2D contour plot in 3D axe as we plot the 2D lines in 3D axe. We can use keywords **zdir='x','y' or 'z'** and **offset** to control the plot

ax.contour3D(xx,yy,px,50,zdir='z',offset=0.015)
ax.contour3D(xx,yy,px,50,zdir='x',offset=10,cmap=cm.coolwarm)





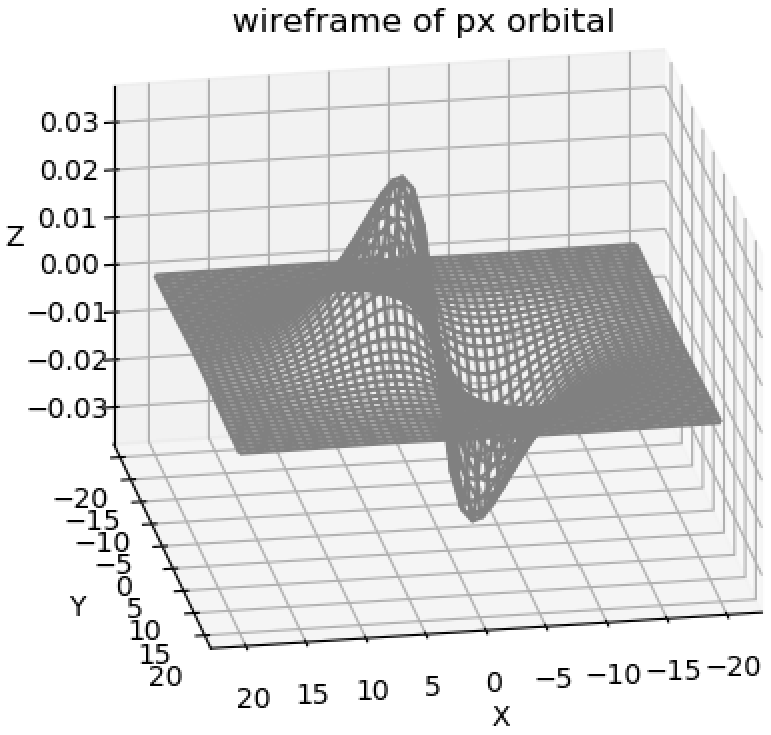
Wireframes and Surface Plots

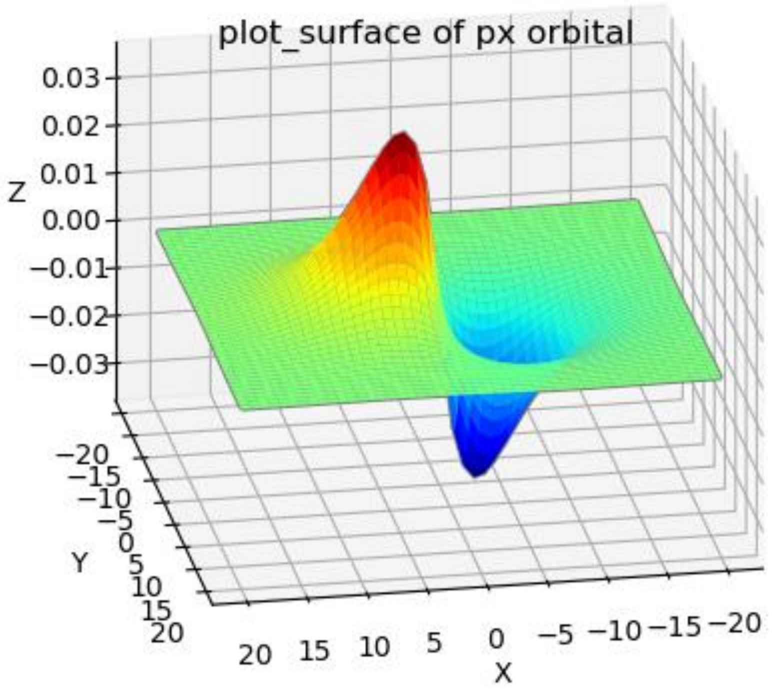
- Two other types of three-dimensional plots that work on gridded data are wireframes and surface plots.

ax.plot_wireframe(xx, yy, px, color='black')

- A surface plot is like a wireframe plot, but each face of the wireframe is a filled polygon.

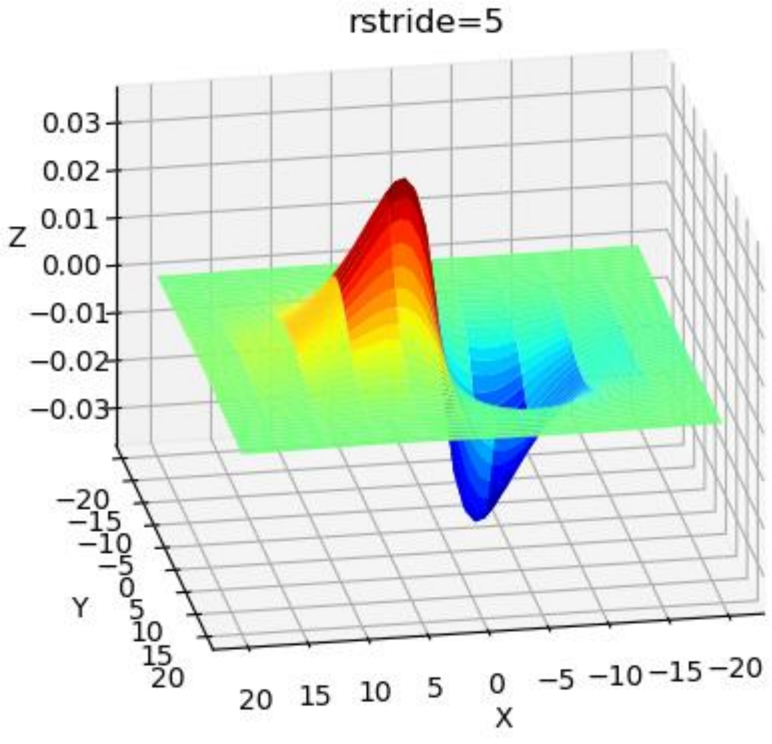
ax.plot_surface(xx, yy, px, cmap='jet', edgecolor='none')

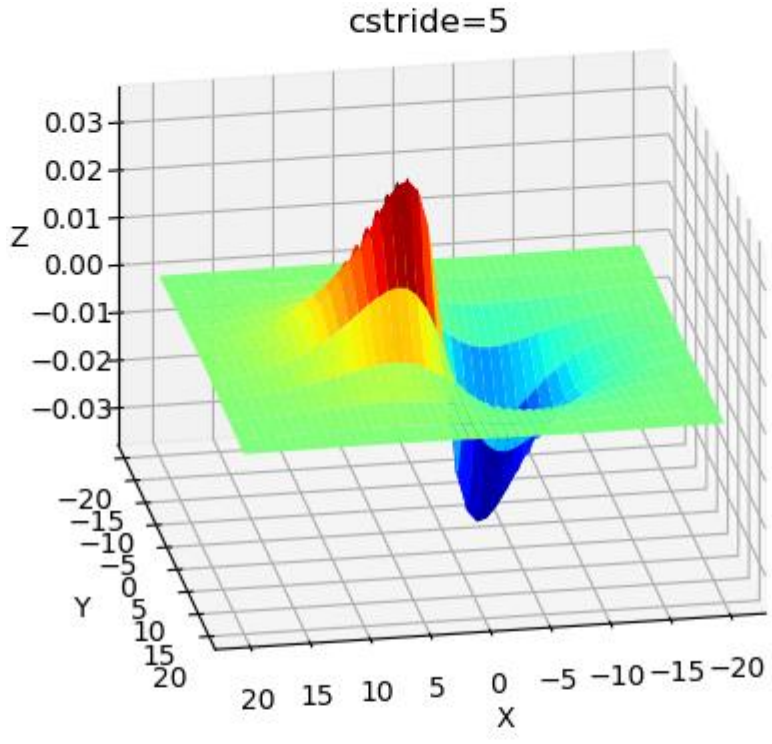




rstride, cstride rcount and ccount

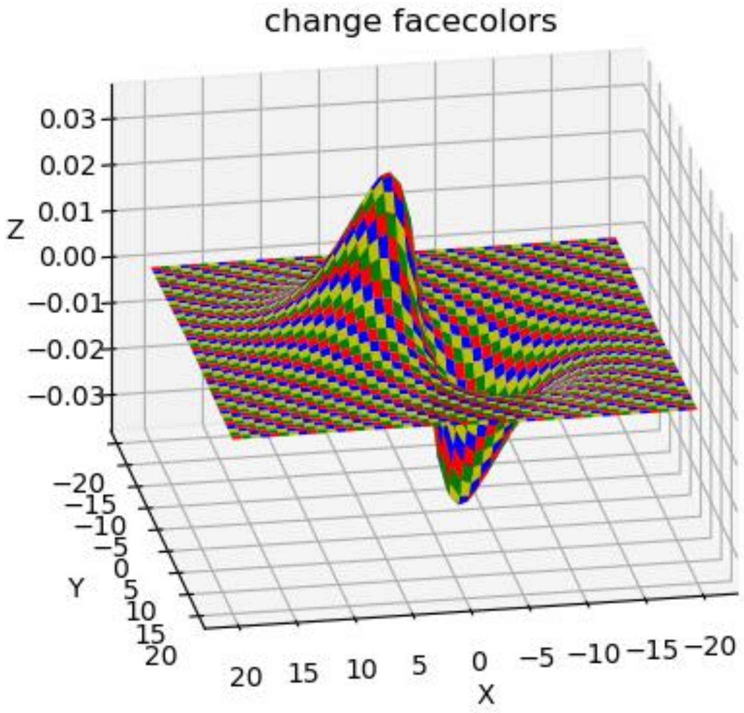
- We can use keywords rstride, cstride rcount and ccount to control the intervals and limits of the rows and columns.
- ***ax.plot_surface(xx, yy, px, rstride=5, cstride=1, cmap='jet', edgecolor='none')***
- ***ax.plot_surface(xx, yy, px, rstride=1, cstride=5, cmap='jet', edgecolor='none')***





facecolors

- We can use keywords facecolors to control the colors for all the individual patches.

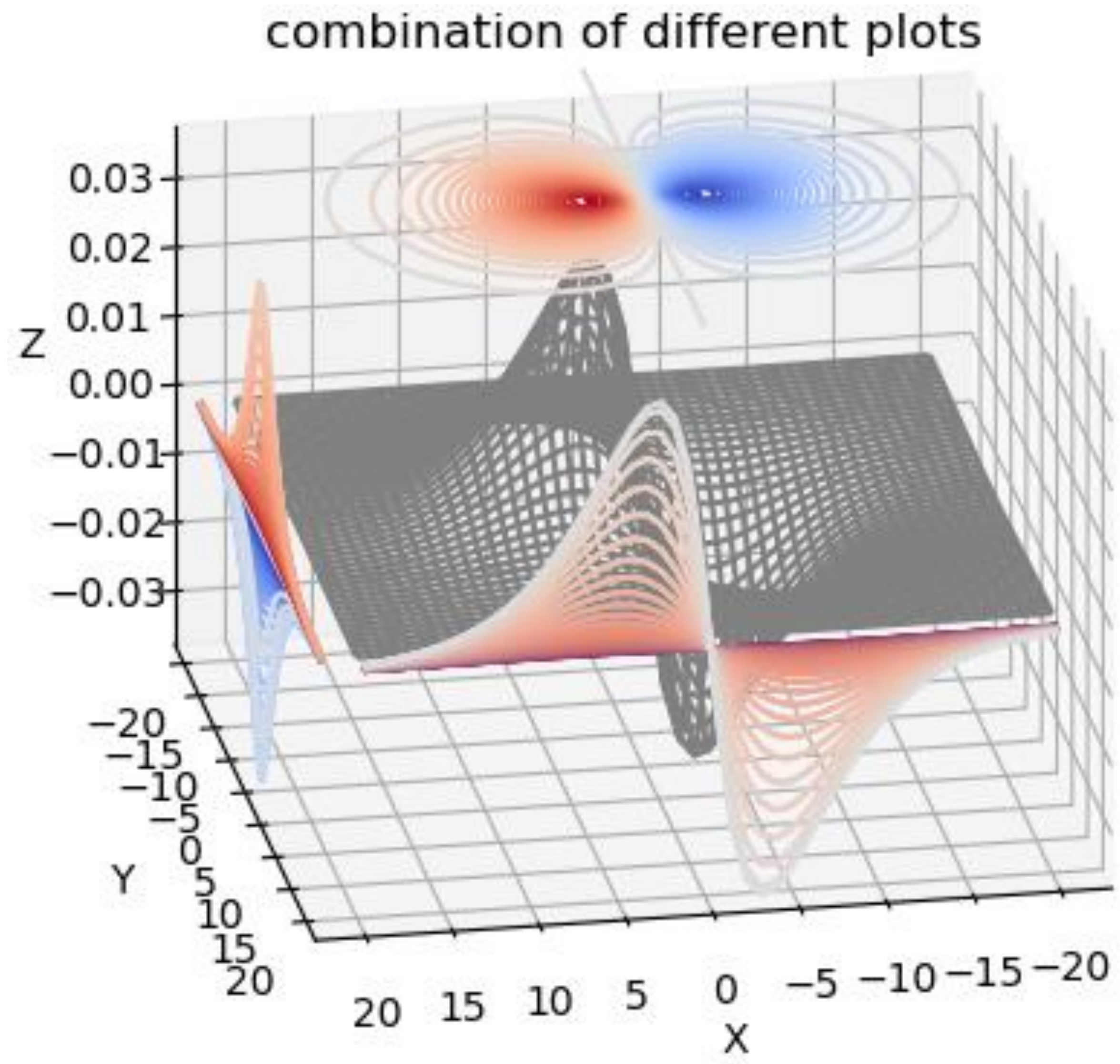


```
colors=np.zeros([num_x,num_y],dtype=str);
colortuple=('r','b','y','g')
for nx in range(num_x):
    for ny in range(num_y):
        colors[nx,ny] = colortuple[(nx + ny)%len(colortuple)]

ax.plot_surface(xx, yy, px, facecolors=colors, linewidth=0,shade=False)
```

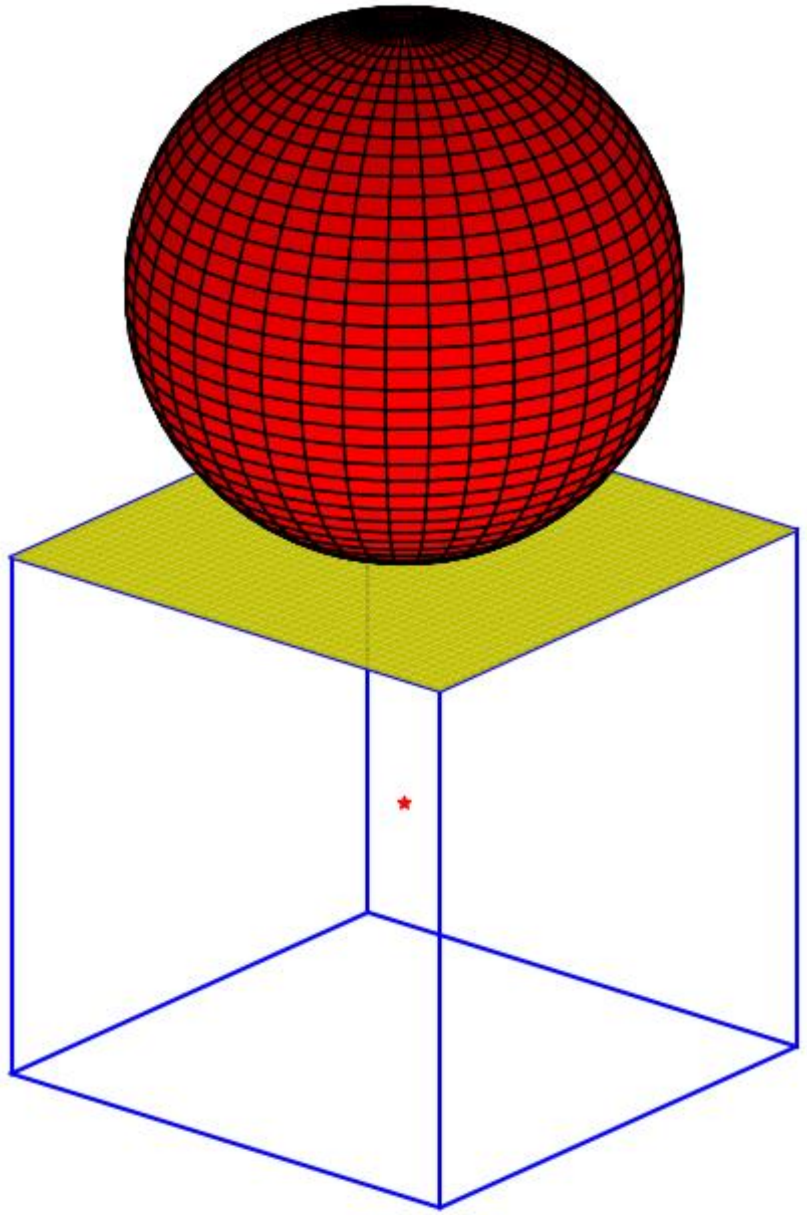
Other keywords

keywords	**Description**
X, *Y*, *Z*	Data values as 2D arrays
rstride	Array row stride (step size)
cstride	Array column stride (step size)
rcount	Use at most this many rows, defaults to 50
ccount	Use at most this many columns, defaults to 50
color	Color of the surface patches
cmap	A colormap for the surface patches.
facecolors	Face colors for the individual patches
norm	An instance of Normalize to map values to colors
vmin	Minimum value to map
vmax	Maximum value to map
shade	Whether to shade the facecolors

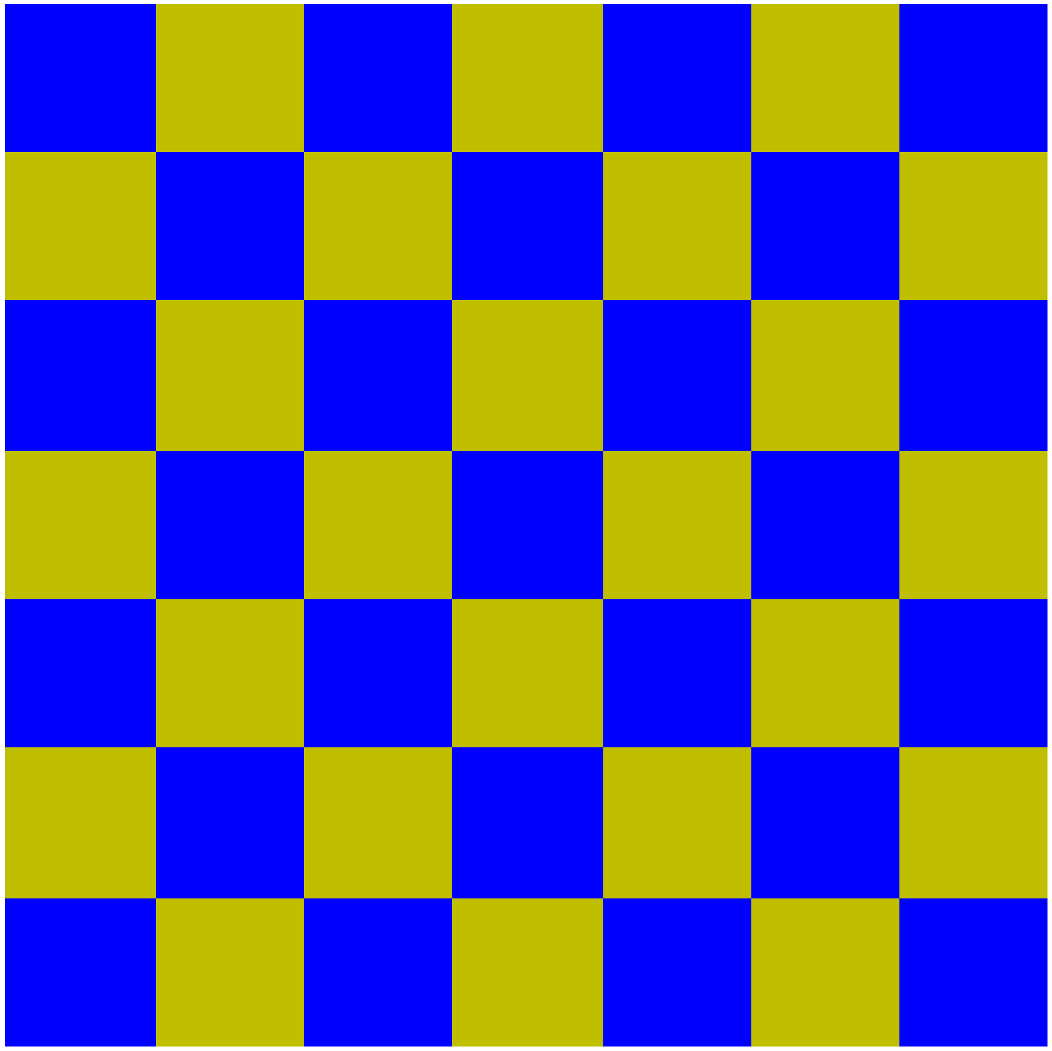
practice


combination of different plots
Z
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
Y
−20
−15
−10
−5
0
5
10
15
20
X
20
15
10
5
0
−5
−10
−15
−20

Plot a sphere one a cube



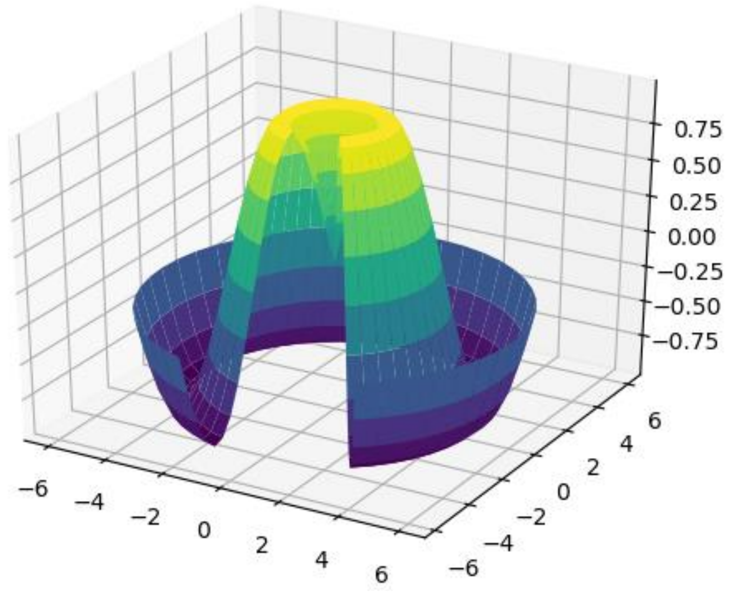
Revisit the checkboard



Non-rectilinear grid

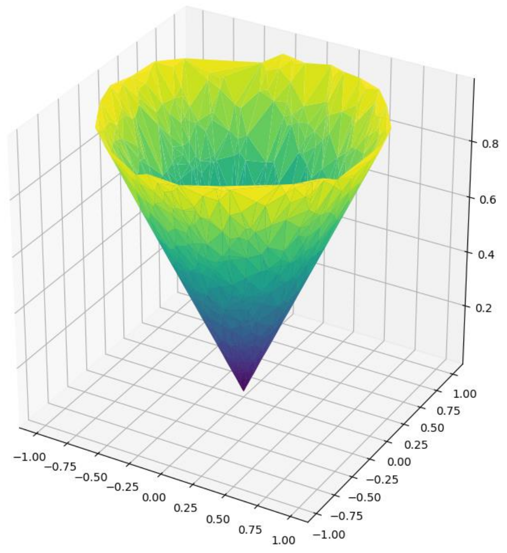
- Note that though the grid of values for a surface plot needs to be two-dimensional, it need not be rectilinear.

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
def f(x, y):
    return np.sin(np.sqrt(x ** 2 + y ** 2))

r = np.linspace(0, 6, 20)
theta = np.linspace(-0.9 * np.pi, 0.8 * np.pi, 40)
r, theta = np.meshgrid(r, theta)

X = r * np.sin(theta)
Y = r * np.cos(theta)
Z = f(X, Y)

ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='viridis', edgecolor='none');
```



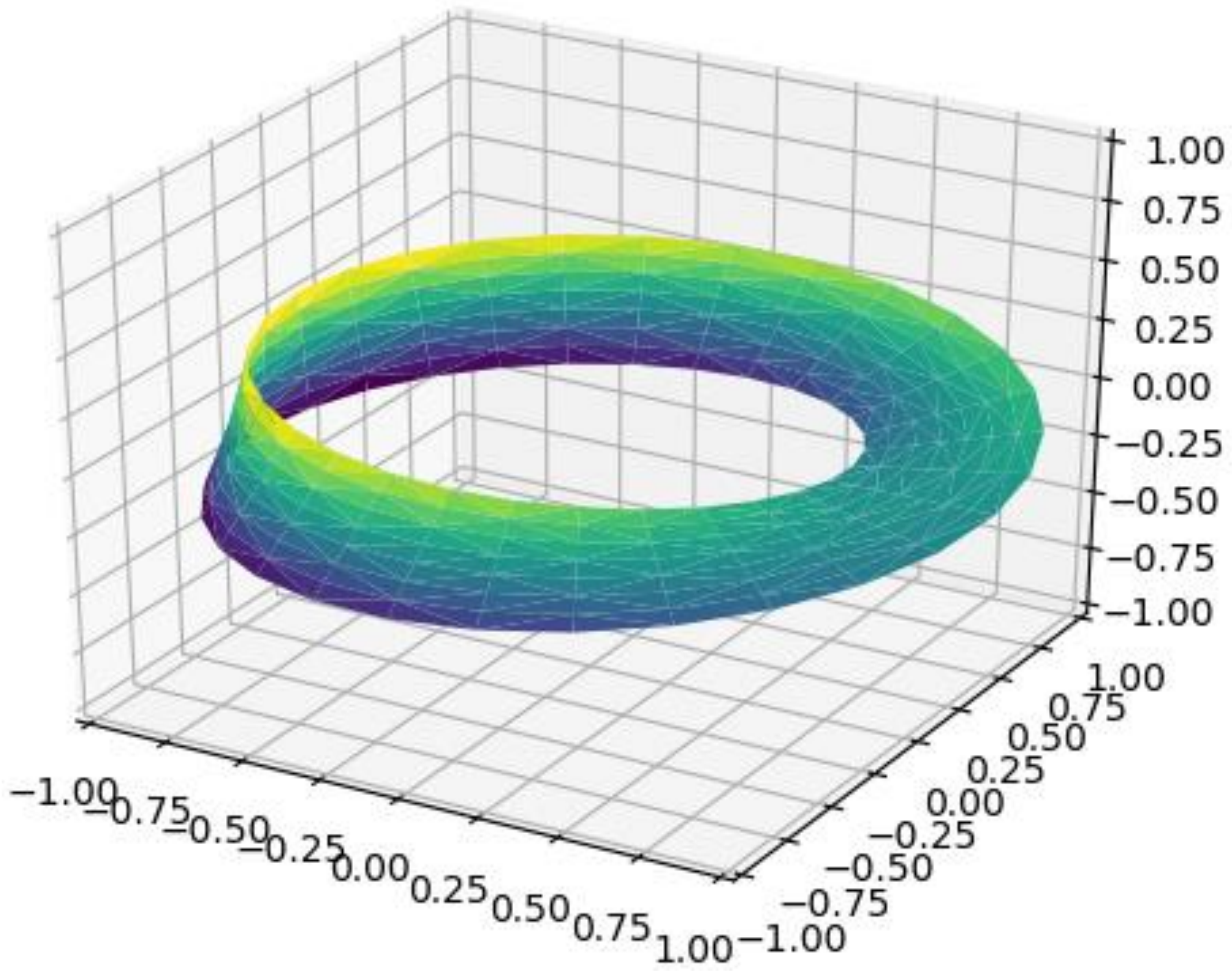
Surface Triangulations

- For some applications, the evenly sampled grids required by the above routines is overly restrictive and inconvenient. In these situations, the triangulation-based plots can be very useful. What if rather than an even draw from a Cartesian or a polar grid, we instead have a set of random draws?



```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
def f(x, y):
    return np.sqrt(x ** 2 + y ** 2)

num_points=1000
theta = 2 * np.pi * np.random.random(num_points)
r = np.random.random(num_points)
X = np.ravel(r * np.sin(theta))*radius; Y = np.ravel(r * np.cos(theta));
Z = f(X, Y)
ax = plt.axes(projection='3d'); plt.axis('square');
ax.plot_trisurf(X, Y, Z, cmap='viridis', edgecolor='none');
```

Practice: Visualizing a Möbius strip



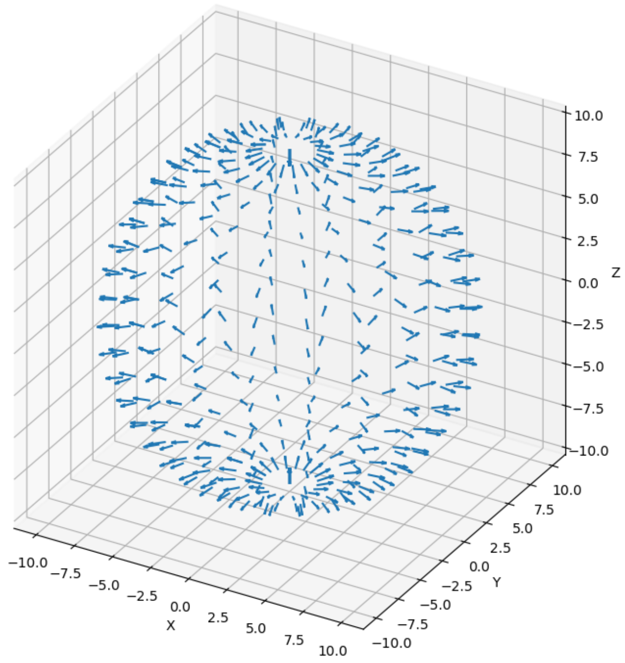
Plot 3D arrows

- We can also plot 3D arrows

quiver3D(x, y, z, u, v, w, length=0.2, normalize=True)

x,y,z is the position of the arrow

u,v,w is the direction of the arrow



Keyword arguments:

length: [1.0 | float]

The length of each quiver, default to 1.0, the unit is the same with the axes

arrow_length_ratio: [0.3 | float]

The ratio of the arrow head with respect to the quiver, default to 0.3

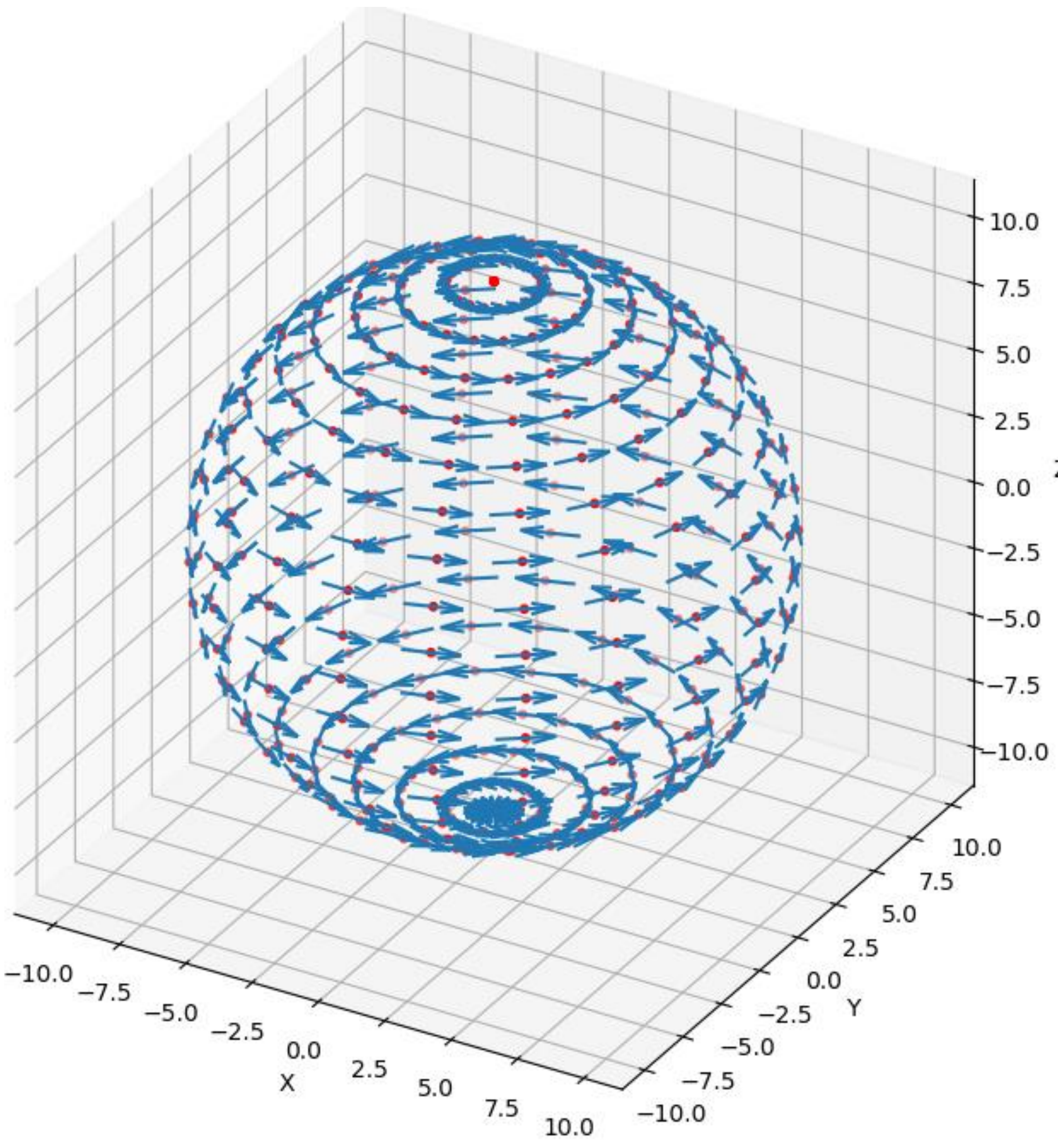
pivot: [‘tail’ | ‘middle’ | ‘tip’]

The part of the arrow that is at the grid point; the arrow rotates about this point, hence the name pivot. Default is ‘tail’

normalize: [False | True]

When True, all of the arrows will be the same length. This defaults to False, where the arrows will be different lengths depending on the values of u,v,w.

Practice: sphere with arrows



10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
Z
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
Y