

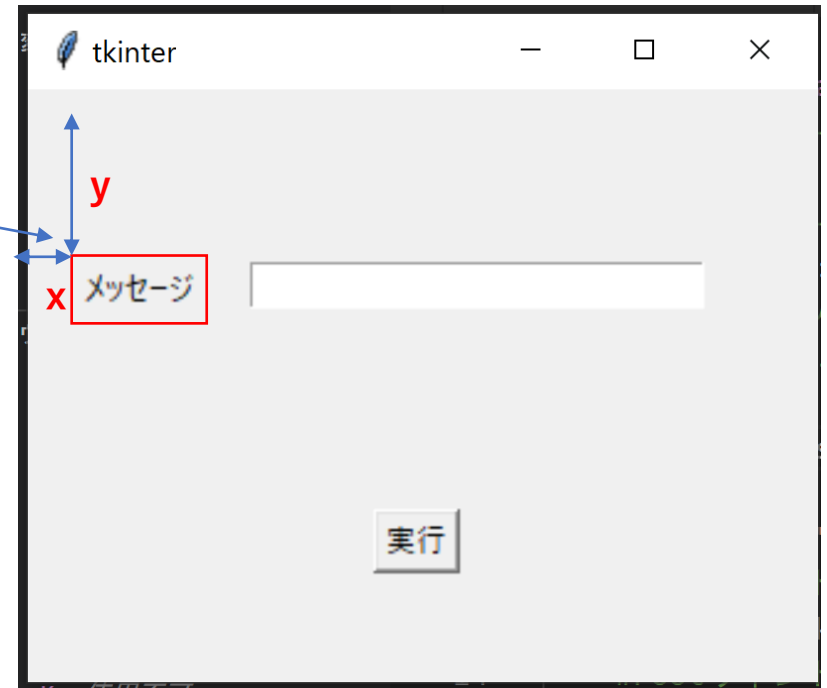
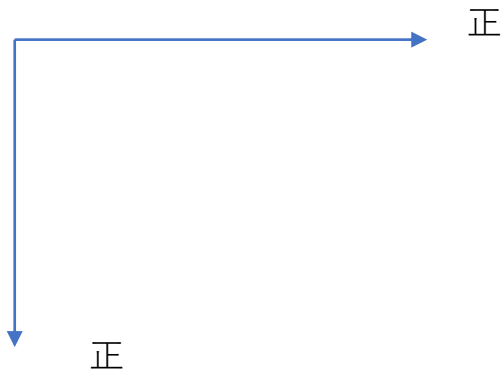
Matlibplot入門

前回の復習

課題 簡易マスターメンテ

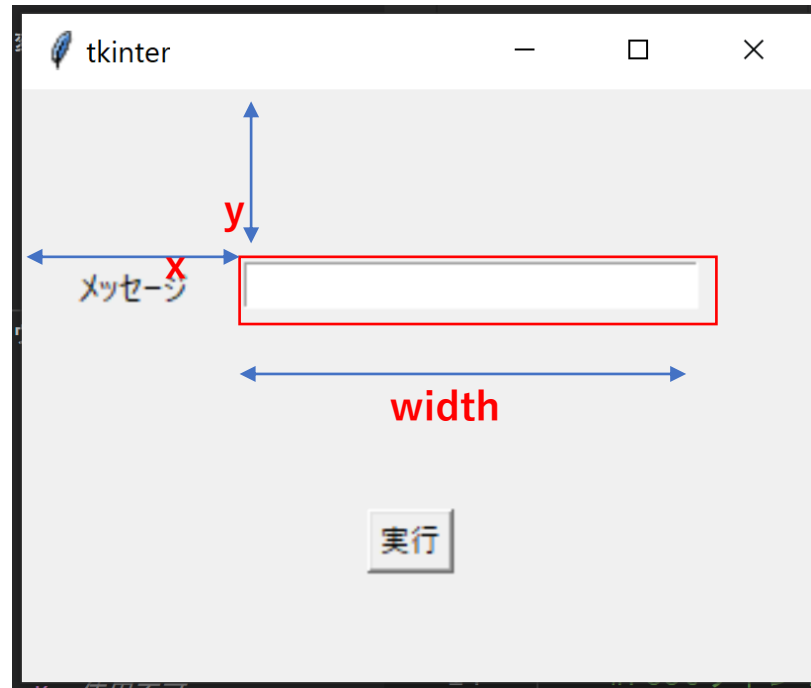
ラベル(lbl.py)

```
#-----  
# ラベル ラベルの文字列を指定  
#-----  
lbl_test = tk.Label(text='メッセージ')  
lbl_test.place(x=20, y=70) # 表示位置
```



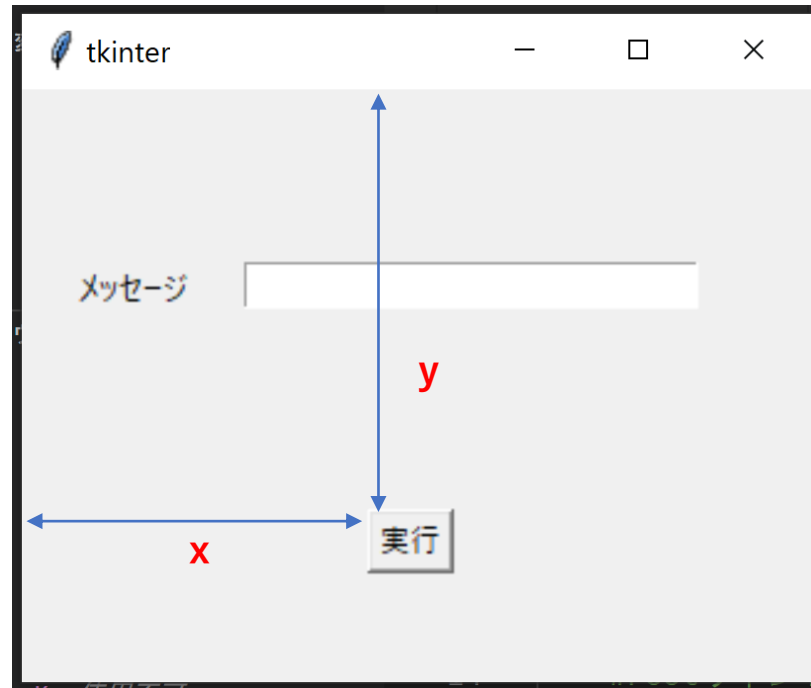
テキストボックス(txt.py)

```
#-----  
# テキストボックス サイズを指定  
#-----  
txt_test = tk.Entry(width=30)  
txt_test.place(x=90, y=70)    # 表示位置
```



ボタン


```
#-----  
#ボタン ボタンの文字と実行する関数を指定  
#-----  
btn = tk.Button(root, text='実行'  
, command=btn_click)  
btn.place(x=140, y=170) #表示位置
```



ボタンはイベントをトリガとして持っているのでメソッドを定義

```
def btn_click:  
    print("ボタンを押された!")
```

```
btn = tk.Button(root, text='実行'  
, command=btn_click)  
    btn.place(x=140, y=170) #表示位置
```



ボタンのイベント(btn.by)

```
btn = tk.Button(root, text='実行',
```

```
command=btn_click)
```

```
btn.place(x=140, y=170) #表示位置
```

```
def btn_click():  
    txt_test.insert(tk.END,"Hello")
```

関数btn_clickを作る

関数btn_clickにテキストボックスに値を入れる

```
def btn_click():  
    txt_test.insert(tk.END,"Hello")
```

この場合はテキストボックスにHelloの文字列を入れる

参照

テキストボックスをクリアするとき

```
txt_test.delete(0, tk.END)#いったんテキストボックスクリア。  
#0から最後までtk.END
```

仕様追加(matrtmente1.pyに追加)

認証画面

UID

PWD

- 1,削除機能をつけてください
- 2,uidがないときや削除したときメッセージを出してください

```
btn3 = tk.Button(root, text='クリア', command=btn_click3)
btn3.place(x=170, y=160) #表示位置
```

```
btn4 = tk.Button(root, text='削除', command=btn_click4)
btn4.place(x=210, y=160) #表示位置
```

```
lblmsg = tk.Label(text=u" ") #
lblmsg.place(x=30, y=180)
```

btn3.placeの下に追加する
ボタンとラベル

btn_click4を実装

#クリアボタン

```
def btn_click3():  
    txtuid.delete(0,tk.END)  
    txtpwd.delete(0,tk.END)
```

クリアボタンの後ろに削除ボタンの処理を書く

#削除ボタン

```
def btn_click4():  
    uid=txtuid.get()  
    print(uid)  
    delete(uid)
```

delete関数を作り実装

delete関数を実装

```
#-----  
#delete  
#引数 uid  
#-----  
def delete(uid):  
    dbname='TestDB.db'  
    conn=sqlite3.connect(dbname)  
    c = conn.cursor()  
    delete_sql =XXXXXXXXXXXXXXXXXXXXX  
    print("SQL=",delete_sql)  
    c.execute(XXXXXXX)  
    conn.commit()  
    conn.close()  
    #テキストボックスクリア  
    txtuid.delete(0,tk.END)  
    txtpwd.delete(0,tk.END)  
    lblmsg['text']="削除しました"
```

lblmsgのラベルを書き換えるときは
オブジェクト名['text']="文字列"

kensaku関数にメッセージを出す

```
def kensaku(uid):  
    dbname='TestDB.db'  
    conn=sqlite3.connect(dbname)  
    select_sql = "select uid,pwd from user where  
uid='"+uid+"'"  
    print(select_sql)  
    flg=0  
    c = conn.cursor()  
    for row in c.execute(select_sql):  
        print(row[0],"*****",row[1])  
        print("OK")  
        uid=row[0]  
        pwd=row[1]  
        txtpwd.insert(tk.END,pwd)  
        flg=1  
    conn.close()  
    if flg==0:  
        print("データない")  
        XXXXXX="データない"
```

データがないときメッセージをだす

解答例 `mastrtmente3.py`

折れ線グラフ

パラメータ

- title,xlabel,ylabel,grid,xtick,legend,figure
- addsubplot,subplots_adjust,set_xlim,set_ylim
set_xlabel,set_ylabel

折れ線グラフ例

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], # xの値
         [1, 4, 9, 16])# yの値
plt.ylabel('y-label') # y軸のラベルをプロット
plt.xlabel('x-label') # x軸のラベルをプロット
plt.show()
```

折れ線グラフを表示する(plot,show)

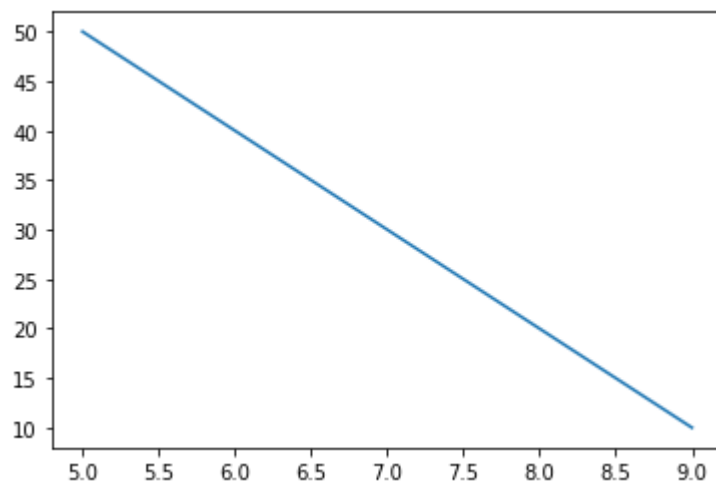
```
import matplotlib.pyplot as plt
```

```
x_list = [ 5, 6, 7, 8, 9]
```

```
y_list = [50, 40, 30, 20, 10]
```

```
plt.plot(x_list, y_list)
```

```
plt.show()
```



グラフのタイトル(plot2.py)(title)

```
import matplotlib.pyplot as plt
```

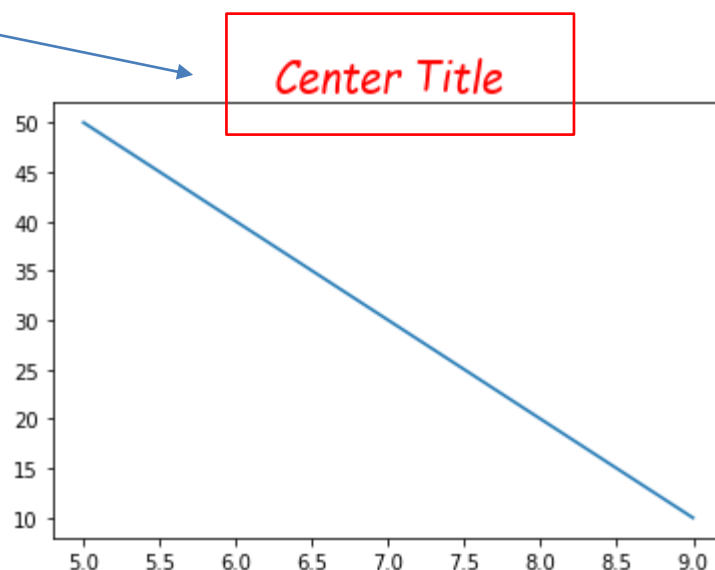
```
x_list = [ 5, 6, 7, 8, 9]
```

```
y_list = [50, 40, 30, 20, 10]
```

```
plt.title('Center Title',  
          color='red',  
          size=20,  
          family='fantasy',  
          )
```

```
plt.plot(x_list, y_list)
```

```
plt.show()
```



x軸y軸のタイトル

```
import matplotlib.pyplot as plt
```

```
x_list = [ 5, 6, 7, 8, 9]
```

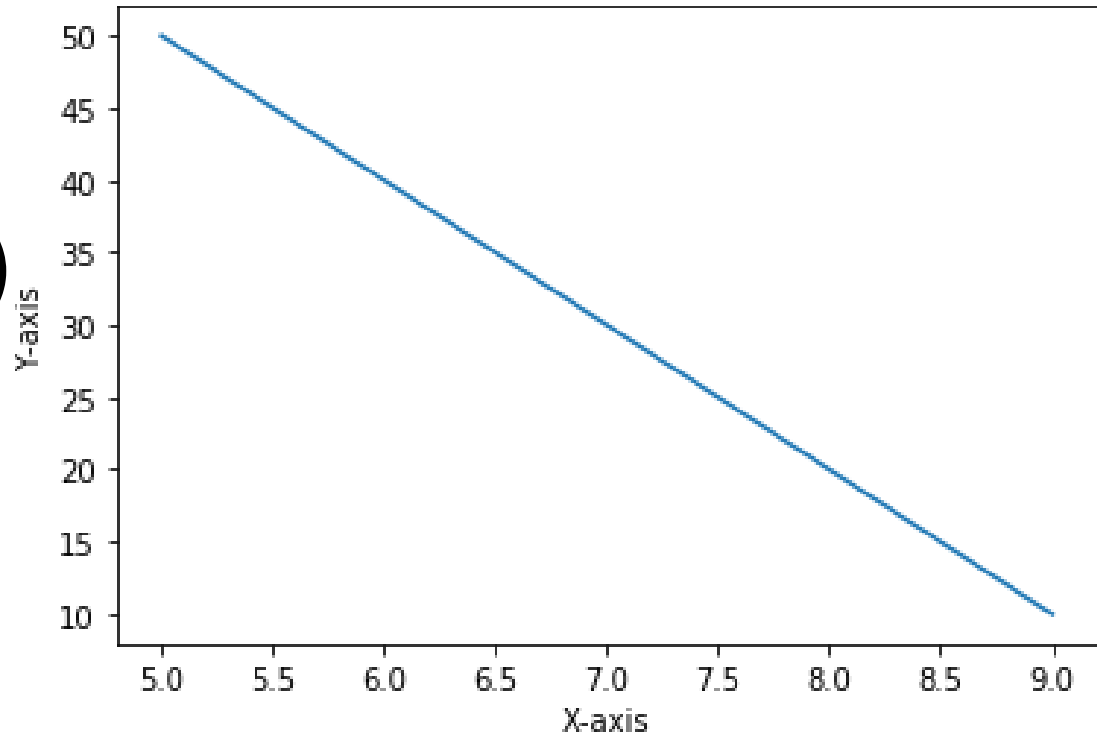
```
y_list = [50, 40, 30, 20, 10]
```

```
plt.xlabel("X-axis")
```

```
plt.ylabel("Y-axis")
```

```
plt.plot(x_list, y_list)
```

```
plt.show()
```



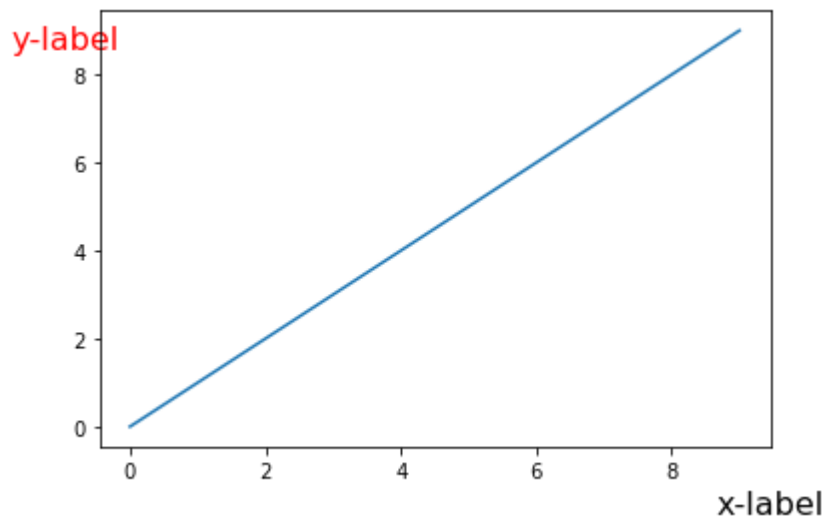
X軸,y軸(xlabel,ylabel)のタイトルのつけ方

```
plt.plot(range(10))      # y値を0～10にしてラインをプロット
```

```
plt.xlabel('x-label',    # x軸ラベルのテキスト
           size=16,
           position=(1,0), # x軸に対して1の位置(右端)に配置
           rotation=0)    # テキストの回転角度を0にする
```

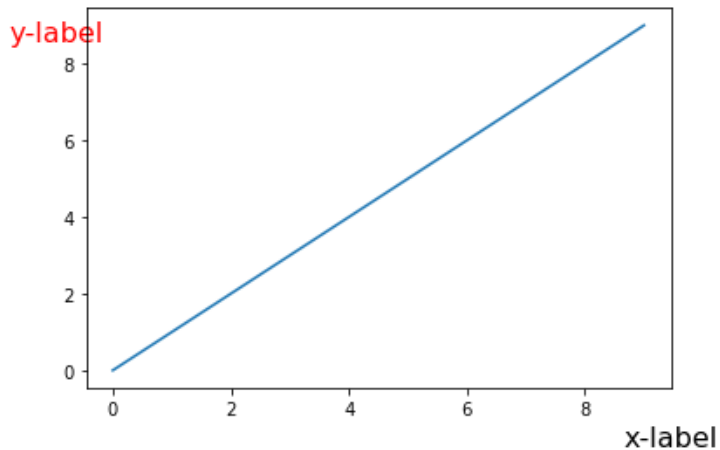
```
plt.ylabel('y-label',    # y軸ラベルのテキスト
           color='red',
           size=16,
           position=(0, 0.9), # y軸に対して0.9の位置に配置
           rotation=0)      # テキストの回転角度を0にする
```

```
plt.show()
```

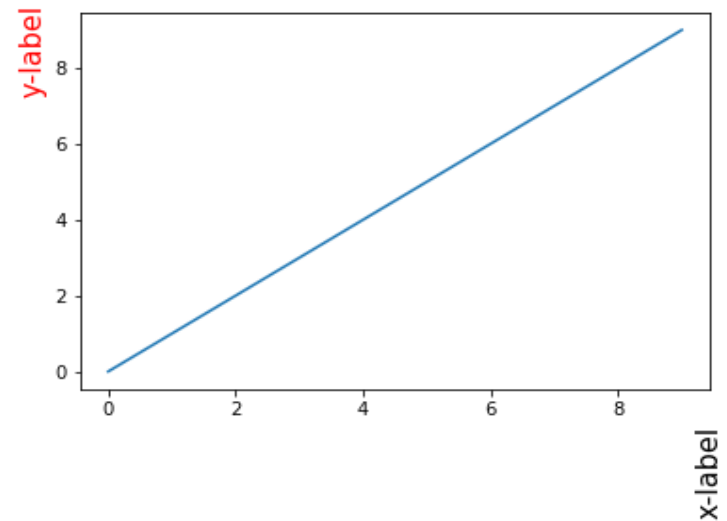


rotationパラメータを変えてみる

rotation=0

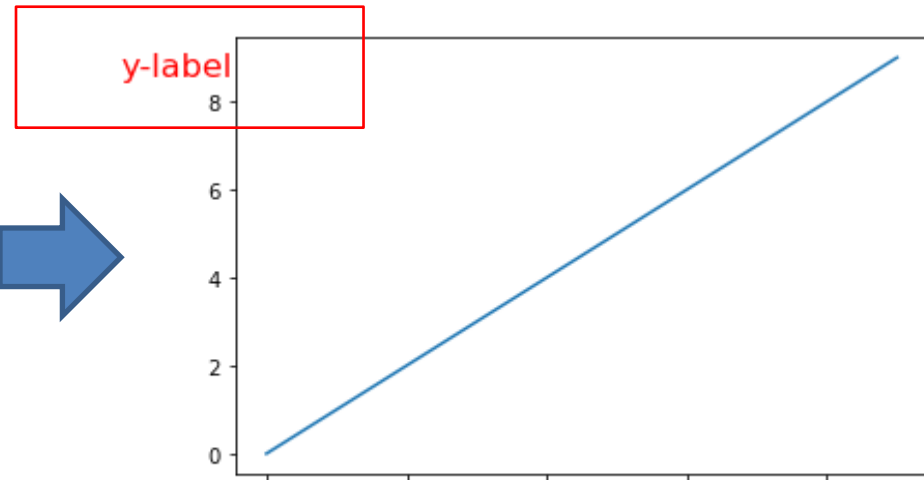
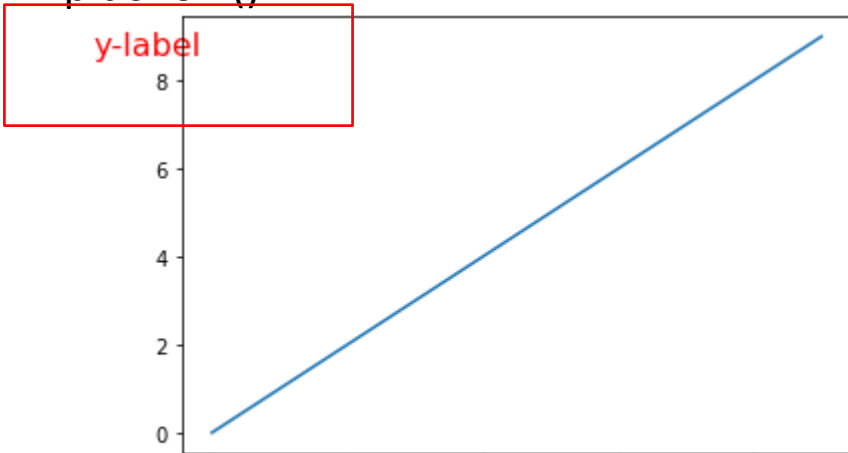


rotation=90



空白を入れてみる

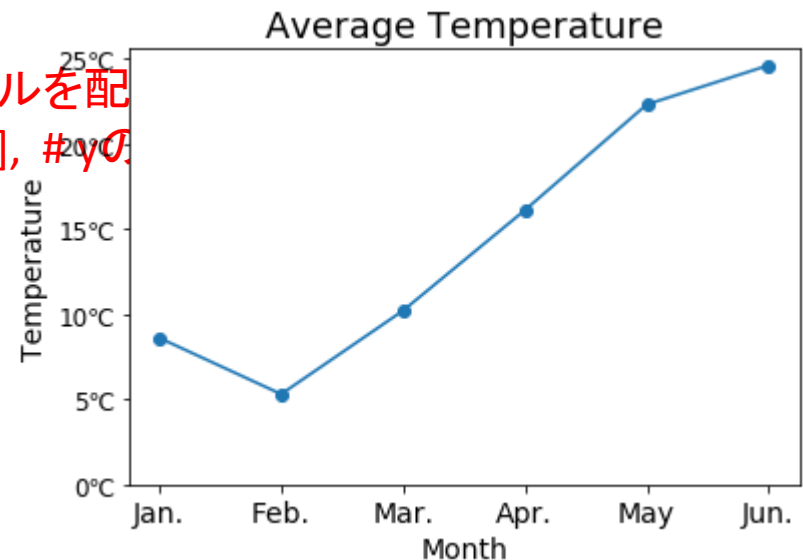
```
plt.plot(range(10))      # y値を0～10にしてラインをプロット
plt.xlabel('x-label',    # x軸ラベルのテキスト
          size=16,
          position=(1,0), # x軸に対して1の位置(右端)に配置
          rotation=0      # 90度反時計回りに回転
)
plt.ylabel('y-label',    # y軸ラベルのテキスト
          color='red',
          size=16,
          position=(0, 0.9), # y軸に対して0.9の位置に配置
          labelpad=15,
          rotation=0      # テキストの回転角度を0にする
)
plt.show()
```



軸の目盛りに単位を入れる (xticks,yticks)

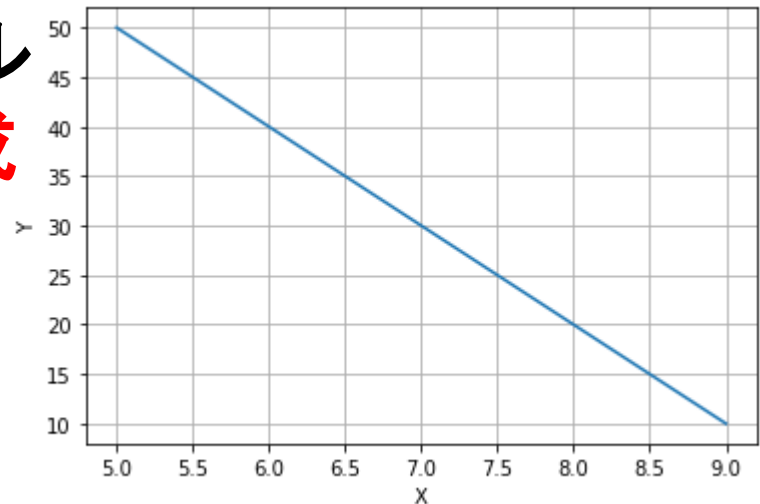
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4, 5, 6],          # xの値 月
         [8.6, 5.3, 10.2, 16.1, 22.3, 24.6], # yの値 気温
         marker='o',                  # サークル型のマーカー
         )
plt.title('Average Temperature', size=18) # タイトル
plt.xlabel('Month', size=14)              # x軸のラベルをプロット
plt.ylabel('Temperature', size=14)        # y軸のラベルをプロット
plt.xticks([1, 2, 3, 4, 5, 6],          # 目盛ラベルを配置するx軸の位置
           ['Jan.', 'Feb.', 'Mar.', 'Apr.', 'May', 'Jun.'], # xの目盛ラベル
           size=14
           )
plt.yticks([0, 5, 10, 15, 20, 25],      # 目盛ラベルを配置するy軸の位置
           ['0°C', '5°C', '10°C', '15°C', '20°C', '25°C'], # yの目盛ラベル
           size=12
           )

plt.show()
```



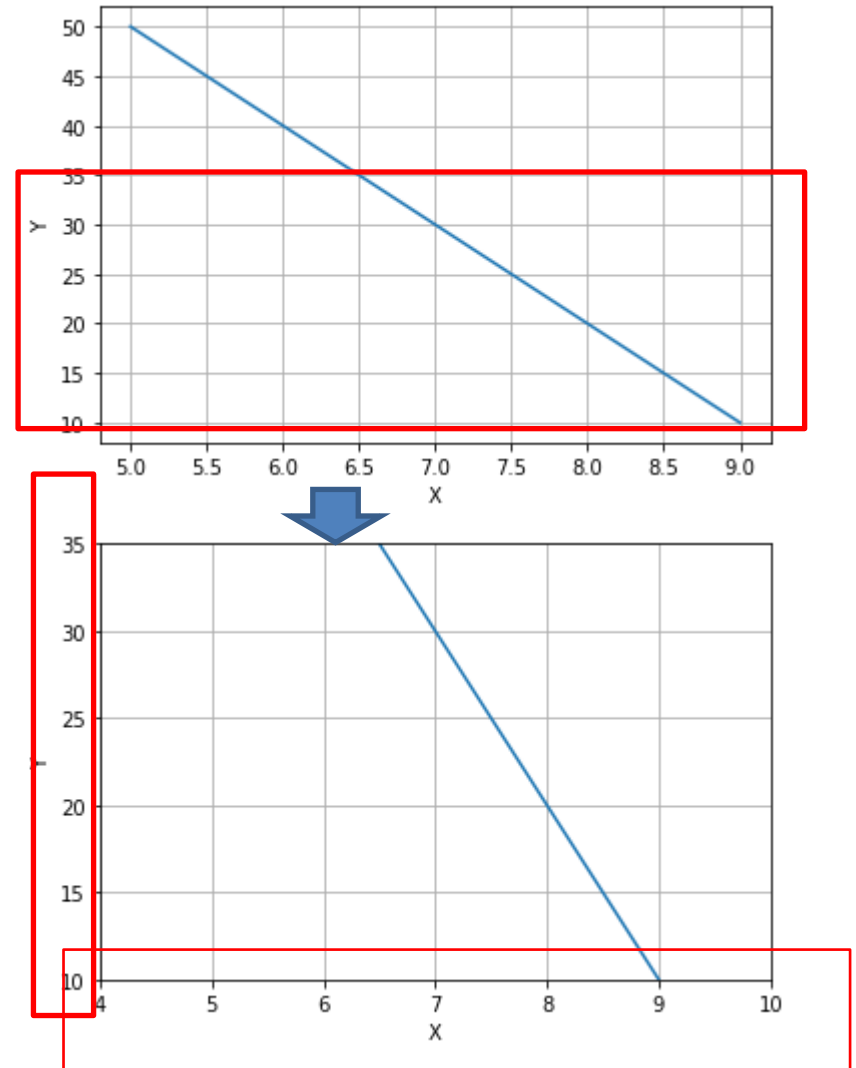
グリッド作成

```
import matplotlib.pyplot as plt
x_list = [ 5, 6, 7, 8, 9]
y_list = [50, 40, 30, 20, 10]
plt.title="Title"
plt.xlabel('X')#x軸のラベル
plt.ylabel('Y')#y軸のラベル
plt.grid()      #グリッド作成
plt.plot(x_list, y_list)
plt.show()
```



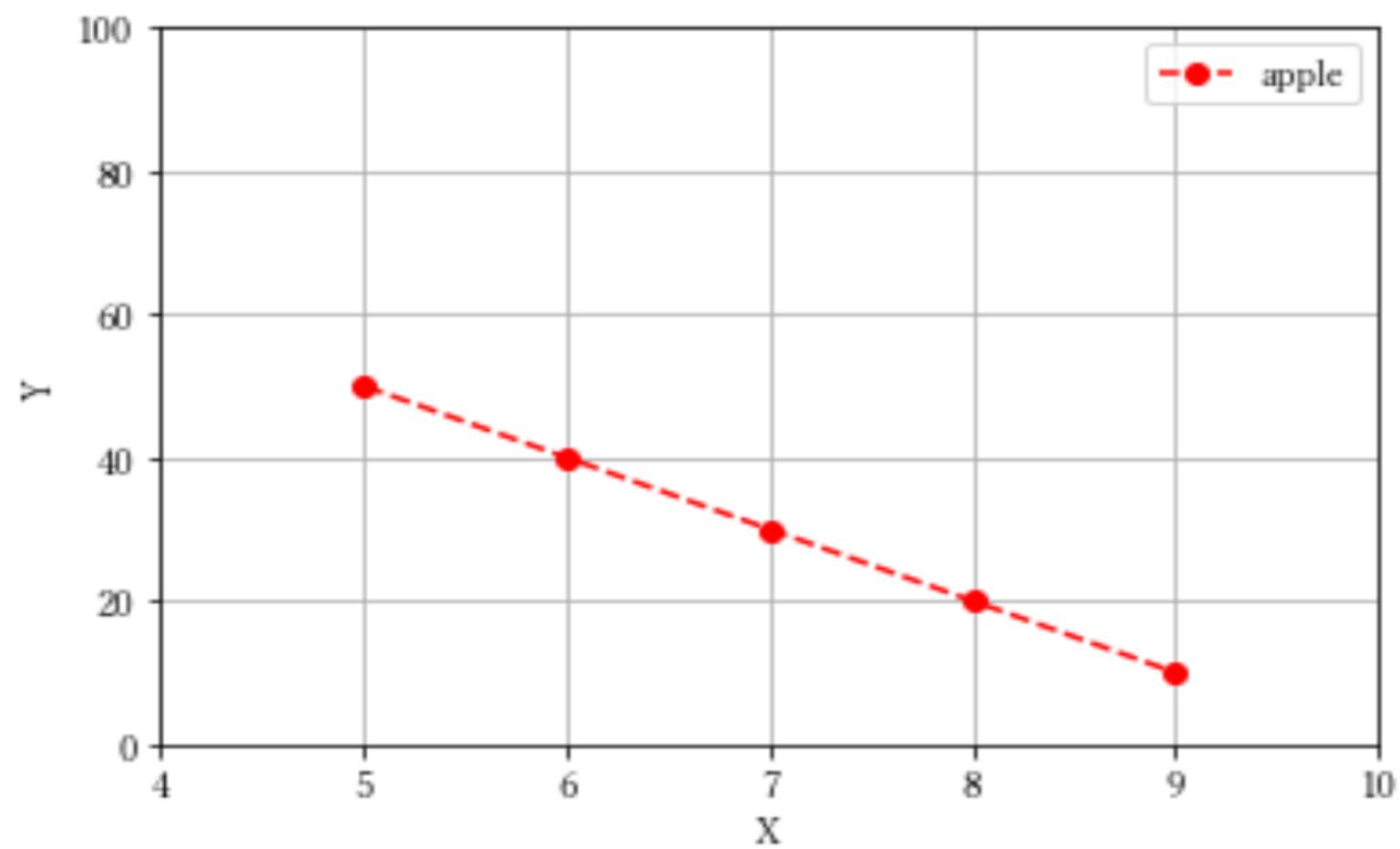
xlim,ylim(表示範囲を制限する)

```
import matplotlib.pyplot as plt
x_list = [ 5, 6, 7, 8, 9]
y_list = [50, 40, 30, 20, 10]
plt.title="Title"
plt.xlim([4,10])
plt.ylim([10,35])
plt.xlabel('X')#x軸のラベル
plt.ylabel('Y')#y軸のラベル
plt.grid()      #グリッド作成
plt.plot(x_list, y_list)
plt.show()
```



マーカー、ライン、色

```
import matplotlib.pyplot as plt
x_list = [ 5, 6, 7, 8, 9]
y_list = [50, 40, 30, 20, 10]
plt.title('Title') # グラフのタイトル
plt.xlabel('X')# X軸のラベル
plt.ylabel('Y')# Y軸のラベル
plt.xlim([4, 10]) # xグラフの表示範囲
plt.ylim([0, 100]) # yグラフの表示範囲
plt.grid() # グリッドの表示
# 書式
marker = 'o'
line = '--'
color = 'r' # b:青 g:緑 r:赤 c:シアン m:マゼンダ y:黄 k:黒 w:白
fmt = marker + line + color
plt.plot(x_list, y_list, fmt, label = 'apple')# グラフデータの設定
plt.legend() # 凡例の表示
plt.show() # グラフの表示
```



Line

line = '--'

line = '-'

line = ':'

line = '-.'

— linestyle : solid, '-'
- - - - - linestyle : dashed, '--'
- · - · - · linestyle : dashdot, '-.'
· · · · · linestyle : dotted, ':'

marker

marker	symbol	description
"."	•	point
","	·	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	ⴚ	tri_down
"2"	ⴞ	tri_up
"3"	ⴠ	tri_left
"4"	ⴡ	tri_right
"8"	⬢	octagon
"s"	■	square
"p"	⬠	pentagon
"P"	⊕	plus (filled)
"*"	★	star

以下のコードでマーカー確かめることができます

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
x = np.arange(1, 11)
```

```
y1 = np.repeat(3, 10) # 3を10回繰り返す
```

```
y2 = np.repeat(2, 10) # 2を10回繰り返す
```

```
y3 = np.repeat(1, 10) # 1を10回繰り返す
```

```
markers1 = [".", ",", "o", "v", "^", "<", ">", "1", "2", "3"]
```

```
markers2 = ["4", "8", "s", "p", "*", "h", "H", "+", "x", "D"]
```

```
markers3 = ["d", "|", "_", "None", None, "",  
"$x$", "$\alpha$", "$\beta$", "$\gamma$"]
```

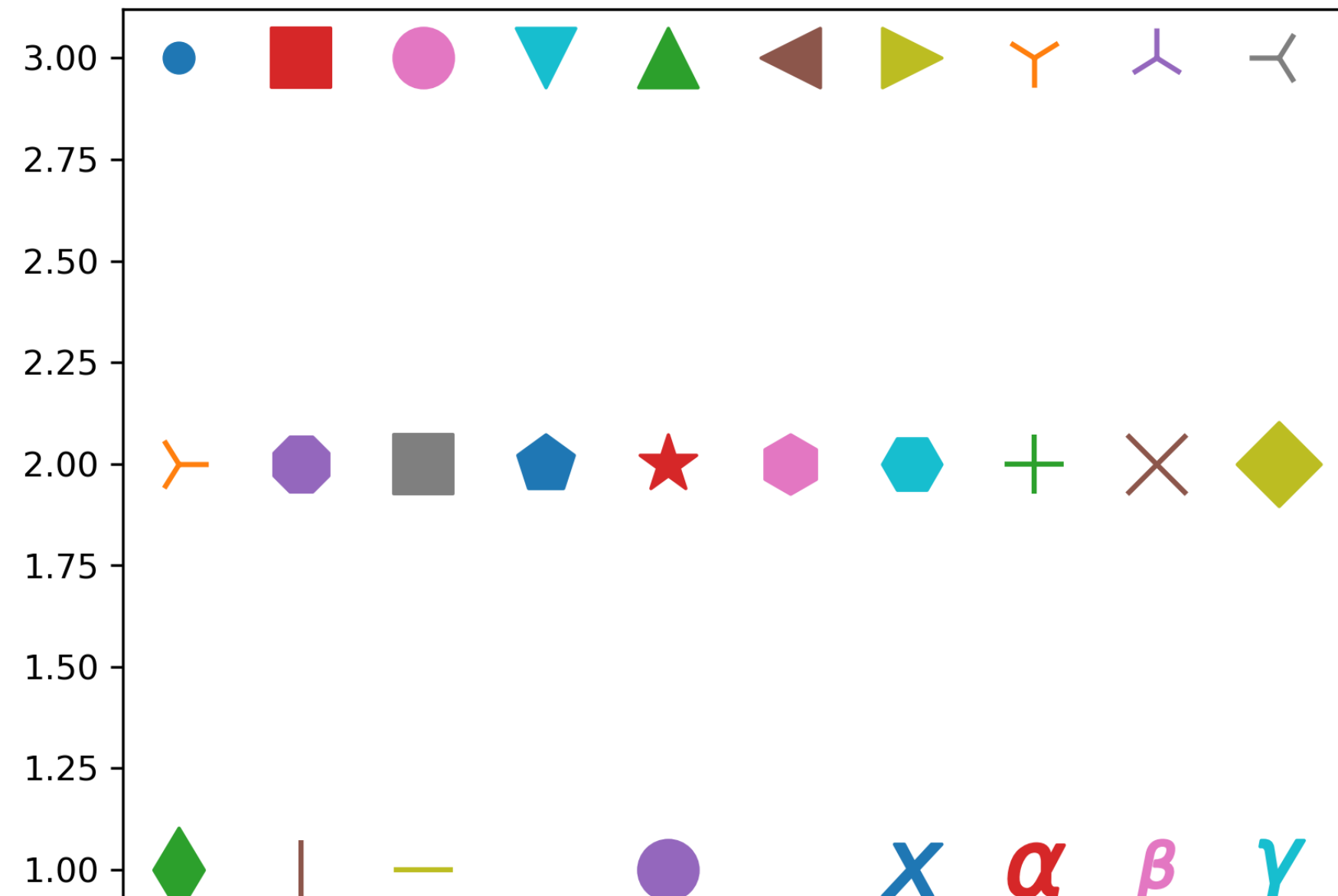
```
for i in x-1:
```

```
    plt.scatter(x[i], y1[i], s=300, marker=markers1[i])
```

```
    plt.scatter(x[i], y2[i], s=300, marker=markers2[i])
```

```
    plt.scatter(x[i], y3[i], s=300, marker=markers3[i])
```

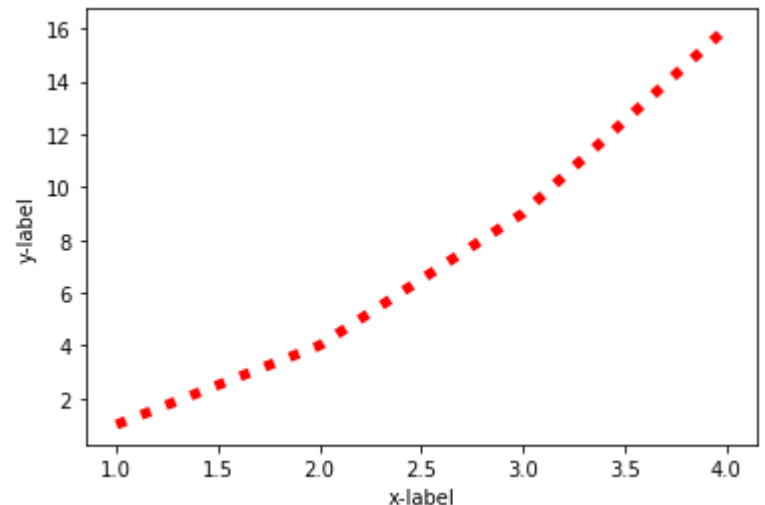

marker.pyの実行結果



ラインを点線にする

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4],    # xの値
         [1, 4, 9, 16],   # yの値
         linestyle='dotted', # ラインを点線にする
         linewidth=5,      # ライン幅は5pt
         color='red'       # ラインの色は赤
        )
plt.ylabel('y-label')     # y軸のラベルをプロット
plt.xlabel('x-label')     # x軸のラベルをプロット
plt.show()
```



11,1,5グラフの軸に目盛りを設定する (xticks,yticks)

- xtick(目盛りを挿入する位置,挿入する目盛り)
- ytick(目盛りを挿入する位置,挿入する目盛り)

コード例

positionsとlabelsを設定します

```
positions = [0, np.pi/2, np.pi, np.pi*3/2, np.pi*2]
```

```
labels = ["0° ", "90° ", "180° ", "270° ", "360° "]
```

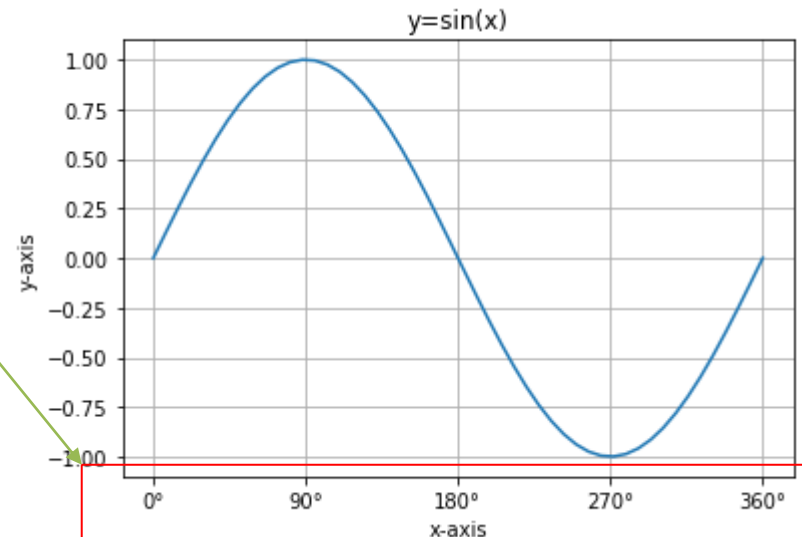
~~グラフのx軸に目盛りを設定してください~~

```
plt.xticks(positions, labels)
```

11, 1, 5コード

```
x = np.linspace(0, 2*np.pi)
y = np.sin(x)
# グラフのタイトルを設定します
plt.title("y=sin(x)")
# グラフのx軸とy軸に名前を設定します
plt.xlabel("x-axis")
plt.ylabel("y-axis")
# グラフにグリッドを表示します
plt.grid(True)
# positionsとlabelsを設定します
positions = [0, np.pi/2, np.pi, np.pi*3/2, np.pi*2]
labels = ["0° ", "90° ", "180° ", "270° ", "360° "]
# グラフのx軸に目盛りを設定してください
plt.xticks(positions, labels)

# データx,yをグラフにプロットし、表示します
plt.plot(x,y)
plt.show()
```

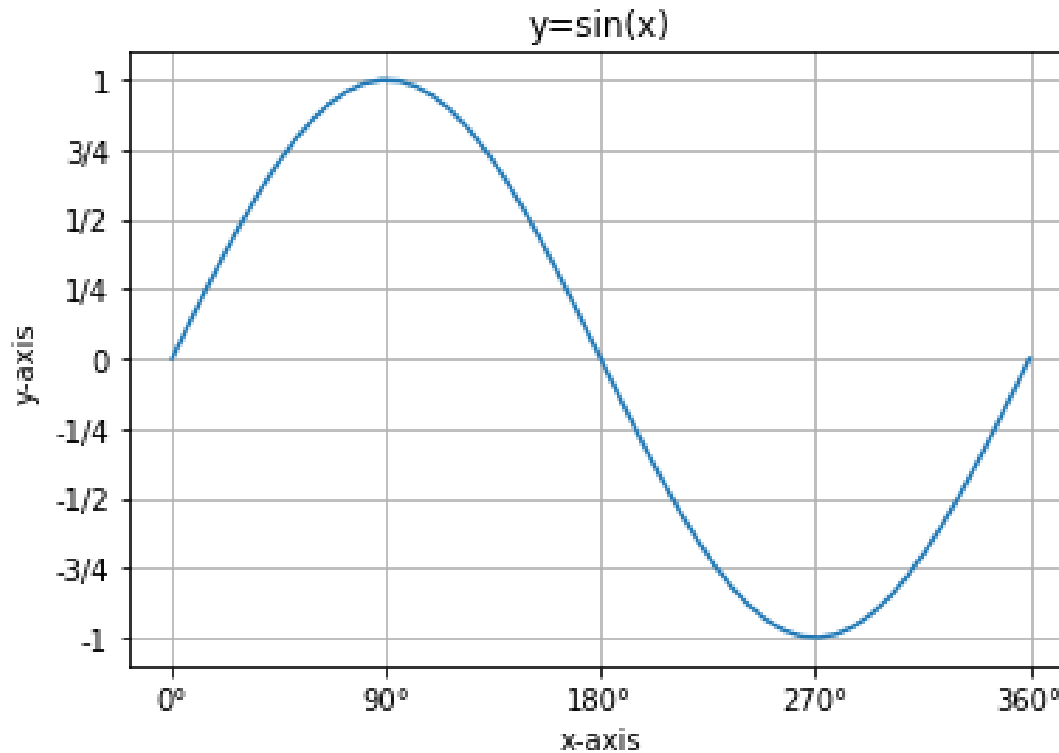


課題

- y_tickを分数表示に変えてみてください

解答

```
positions = [-1, -3/4, -1/2, -1/4, 0, 1/4, 1/2, 3/4, 1]  
labels = ["-1", "-3/4", "-1/2", "-1/4", "0", "1/4", "1/2", "3/4", "1"]  
plt.yticks(positions, labels)
```



演習

(1) 巨人軍の視聴率をグラフにしてください

(rate.csv)

(2) SQLから読み込みグラフにしてください

(graf1.py)

複数のデータを可視化する

1つのグラフに2種類のデータをプロットする

11,1,2 plotを二度書くと重なったグラフが描ける

```
plt.title("graphs of trigonometric functions")
# グラフのx軸とy軸に名前を設定します
plt.xlabel("x-axis")
plt.ylabel("y-axis")
# グラフにグリッドを表示します
plt.grid(True)
# グラフのx軸にラベルを設定します
plt.xticks(positions, labels)
# データx, y1をグラフにプロットし、黒で表示してください
plt.plot(x, y1, color="k")
# データx, y2をグラフにプロットし、青で表示してください
plt.plot(x, y2, color="b")
plt.show()
```

上3つだけの部分を書く

Figureオブジェクトを作成します

```
fig = plt.figure(figsize=(9, 6))
```

```
ax = fig.add_subplot(2, 3, 1)
```

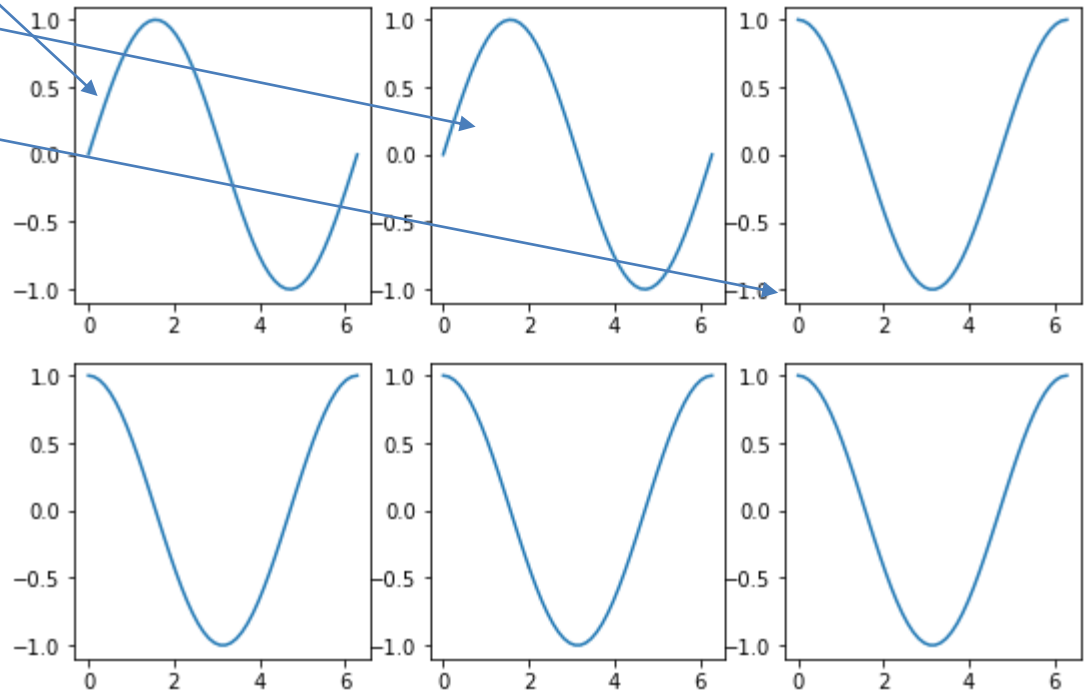
```
ax.plot(x,y)
```

```
ax = fig.add_subplot(2, 3, 2)
```

```
ax.plot(x,y)
```

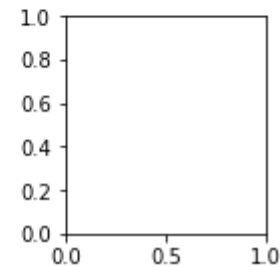
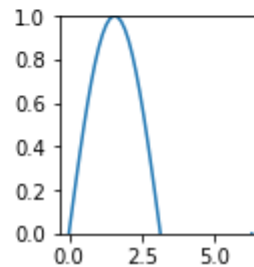
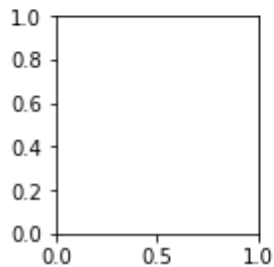
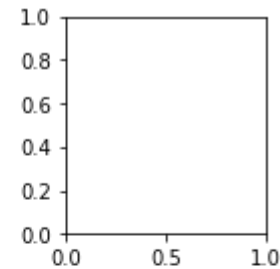
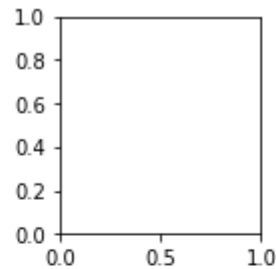
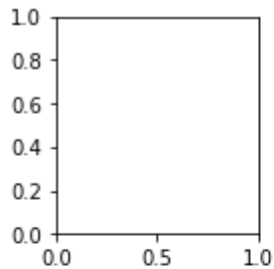
```
ax = fig.add_subplot(2, 3, 3)
```

```
ax.plot(x,y)
```

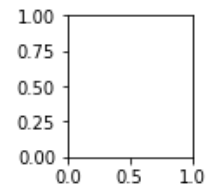
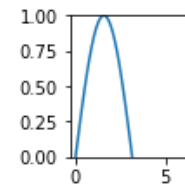
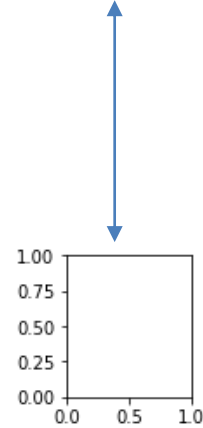
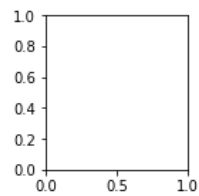
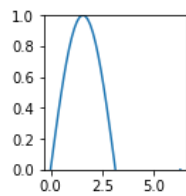
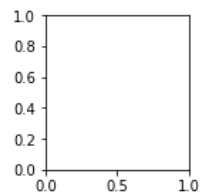
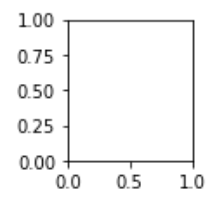
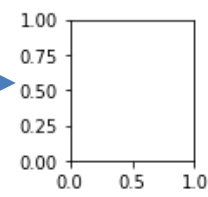
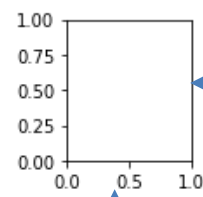
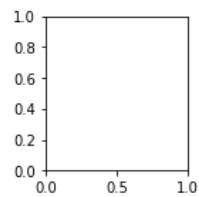
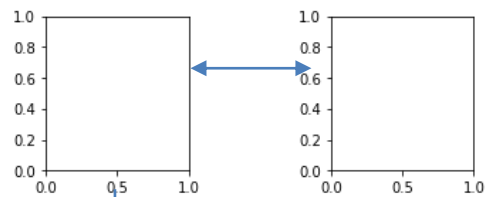


`plt.subplots_adjust(wspace=1, hspace=1)`

- `subplots_adjust(wspace=横間隔をあける割合, 縦間隔をあける割合)`



$(wspace=1, hspace=1) \nless (wspace=2, hspace=2)$



11.3.4サブプロット内のグラフの表示範囲を設定する

set_xlim,set_ylim,set_xlabel,set_ylabel,set_title

set_xlim (範囲)

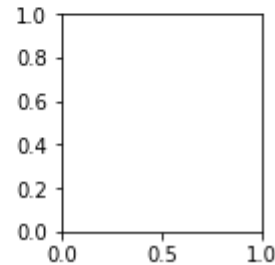
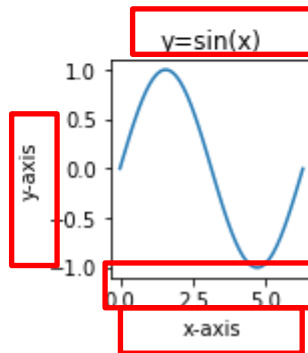
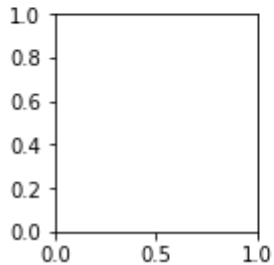
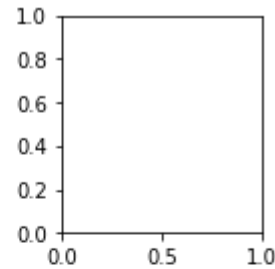
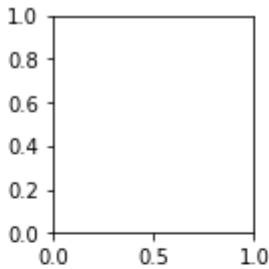
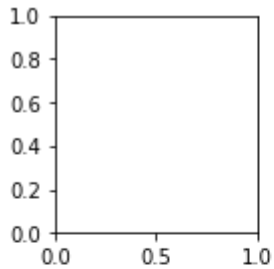
set_ylim (範囲)

set_title(“タイトル”)

set_xlabel(“x軸の名前”)

set_ylabel(“y軸の名前”)

set_xlim,set_xlabel,set_ylabel,set_title



11.3.6サブプロット内のグラフにグリッド を表示する

```
ax.grid(True)
```


11.3.7サブプロット内のグラフに目盛りを設定する

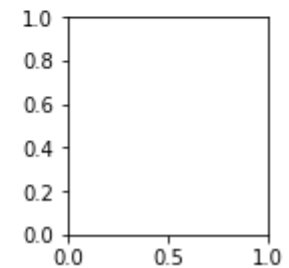
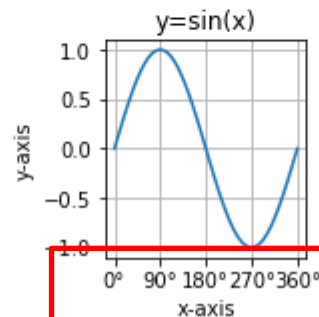
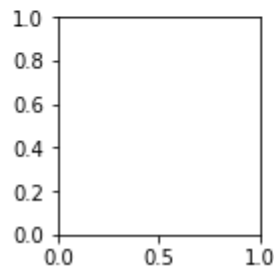
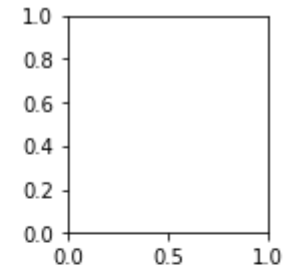
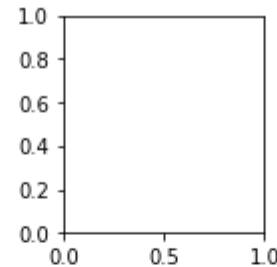
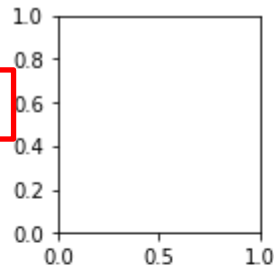
- `set_xtick(“挿入位置リスト”)`
- `set_xticklabels(“目盛りのリスト”)`

11.3.7

```
positions = [0, np.pi/2, np.pi, np.pi*3/2, np.pi*2]  
labels = ["0° ", "90° ", "180° ", "270° ", "360° "]
```

`ax.set_xticks(position)`

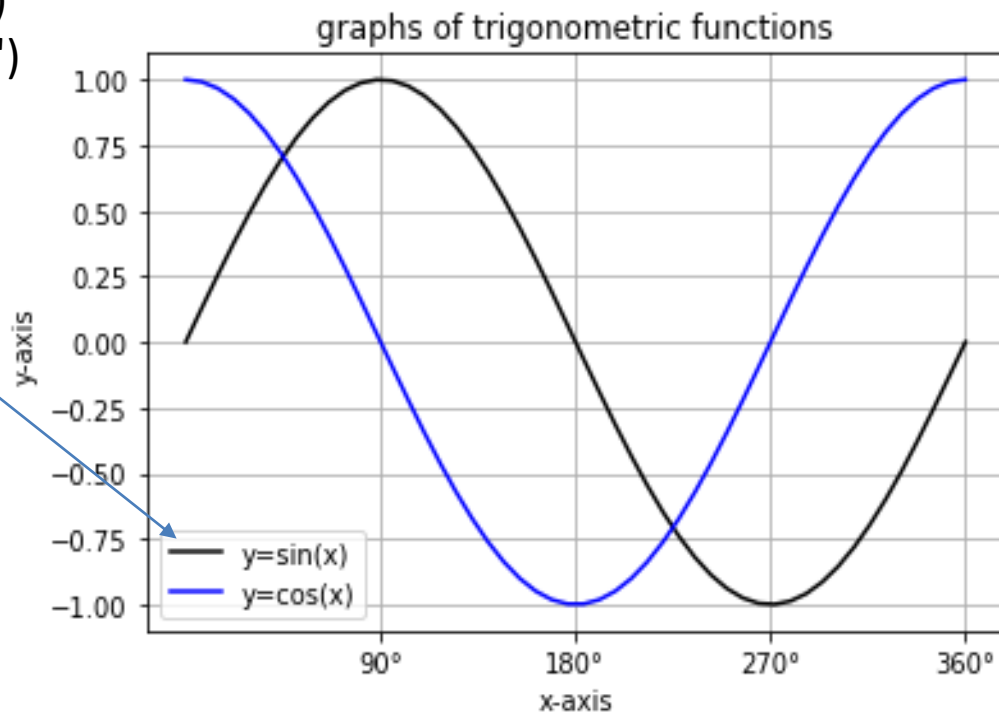
`ax.set_xticklabels(labels)`



11,2系列ラベルを設定する(legend)

```
plt.title("graphs of trigonometric functions")  
plt.xlabel("x-axis")  
plt.ylabel("y-axis")  
plt.grid(True)  
plt.xticks(positions, labels)  
plt.plot(x, y1, color="k", label="y=sin(x)")  
plt.plot(x, y2, color="b", label="y=cos(x)")  
plt.legend(["y=sin(x)", "y=cos(x)"])
```

リストで区切ると複数の凡例が入る



locパラメータ

```
plt.legend(["y=sin(x)", "y=cos(x)"], loc='upper right')
```

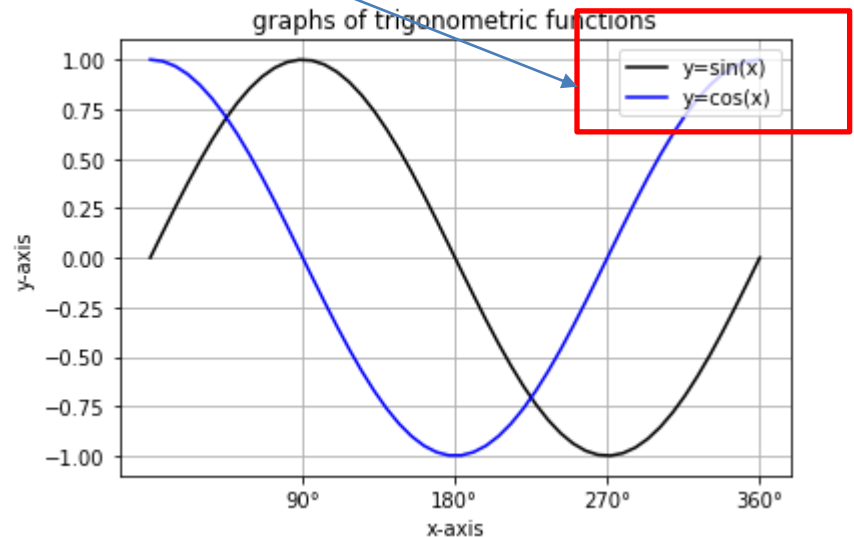
locパラメータは凡例を

loc= 'upper right' 上右方向

loc= 'upper left' 上左方向

loc= 'lower right' 下右方向

loc= 'lower left' 下左方向

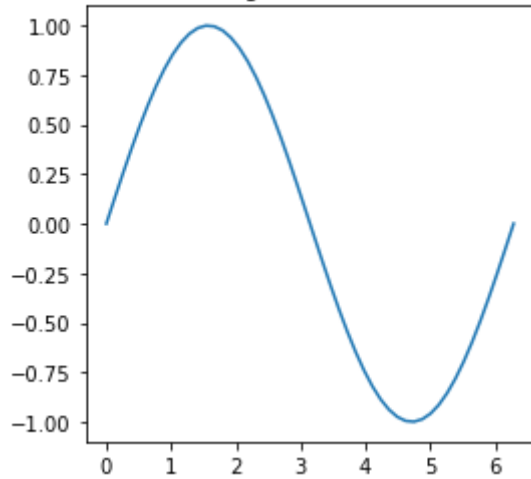


11,3,1図の大きさを設定する figure

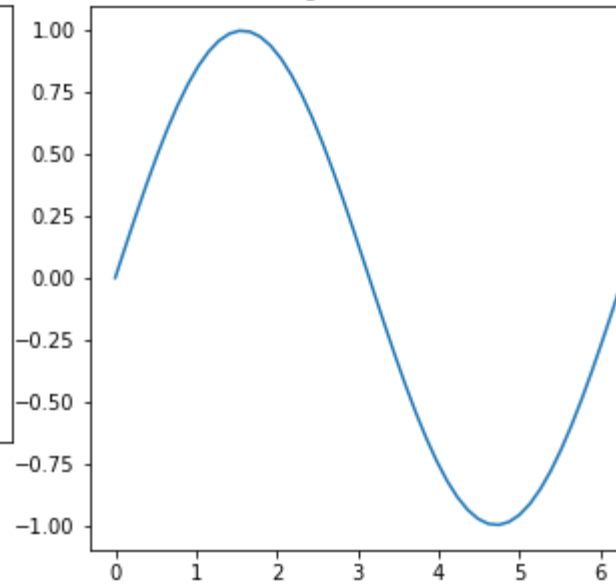
- `figure(figsize(横の大きさ,縦の大きさ))`
(単位はインチ)

figsize=(x,y)

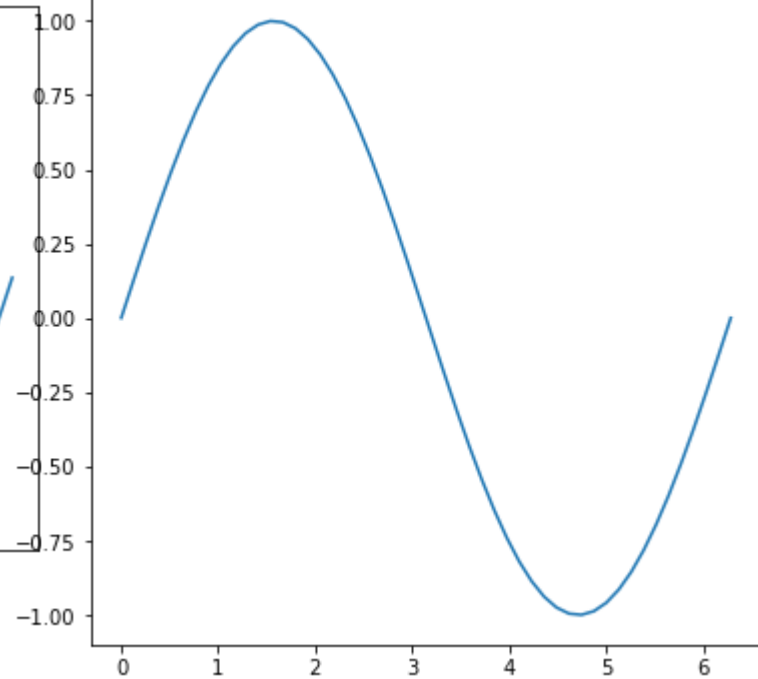
figsize=(4, 4)



figsize=(5, 5)



figsize=(6, 6)



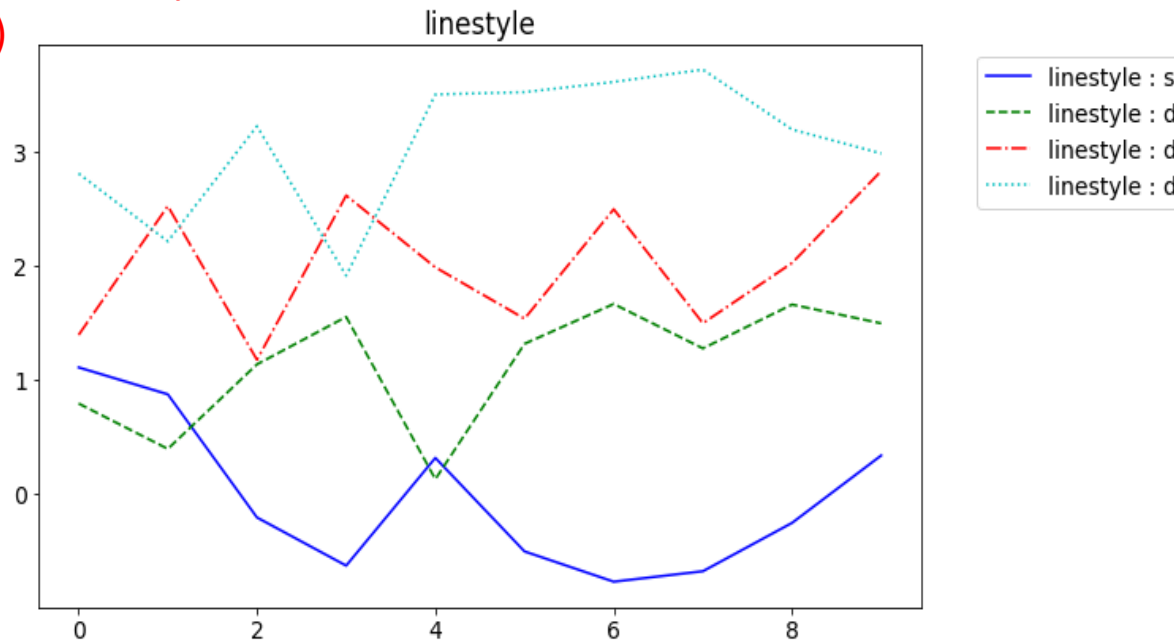
11,3,2サブプロットを作成する (figureオブジェクトとadd_subplot)

```
fig = plt.figure(figsize=(9, 6))
ax = fig.add_subplot(2, 3, 5)
ax.plot(x,y)
axi = []
for i in range(6):
    if i==4:
        continue
    fig.add_subplot(2, 3, i+1)
plt.show()
```

```

import matplotlib.pyplot as plt
import numpy as np
x = np.arange(10)
y0 = np.random.normal(0, 0.5, 10)
y1 = np.random.normal(1, 0.5, 10)
y2 = np.random.normal(2, 0.5, 10)
y3 = np.random.normal(3, 0.5, 10)
plt.rcParams["font.size"] = 14
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(1, 1, 1, title="linestyle")
ax.plot(x, y0, linestyle="-", c="b", label="linestyle : solid, '-'")
ax.plot(x, y1, linestyle="--", c="b", label="linestyle : dashed, '--'")
ax.plot(x, y2, linestyle="-.", c="b", label="linestyle : dashdot, '-.'")
ax.plot(x, y3, linestyle=":", c="b", label="linestyle : dotted, ':'")
ax.legend(bbox_to_anchor=(1.05, 1))
plt.show()

```



課題

temperature.csvに各地域の気温データがあります。これを折れ線グラフにしてください
(先頭から20ぐらいで)

(1)一つのグラフに複数書く場合

(2)複数のグラフに書く場合(6つのグラフ)時間のある人はしてください

2次元のグラフを書く

$y = x^2 + 2x - 1$ のグラフを書く

```
def f(x):  
    return x*x+2*x-1
```

```
x=np.linspace(-3,3,100)  
print(np.round(x,2))
```

```
plt.plot(x,f(x))  
plt.grid(True)  
plt.show()
```

課題

$y = x^3 + 2x - 1$ のグラフを書く

```
# -*- coding: utf-8 -*-  
from sympy import *
```

```
var("a:z")      # a～zまで変数として扱う
```

```
#f = x**2 + 3*x + 2    # 関数f(x)の定義
```

```
f = x*x+2*x-1
```

```
y=[]
```

```
x3=[]
```

```
for x2 in np.linspace(-3.0, 3.0, 100):
```

```
    #print(x4)
```

```
    f1 = f.subs([(x, x2)])
```

```
    x3.append(x2)
```

```
    y.append(f1)
```

```
plt.plot(x3,y)
```

```
plt.show()
```

12.3 ヒストグラム

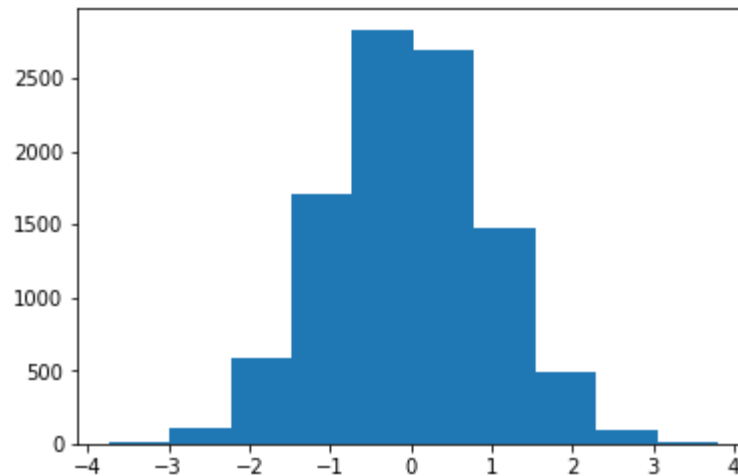
histを使う

```
import numpy as np  
import matplotlib.pyplot as plt  
%matplotlib inline
```

```
np.random.seed(0)  
data = np.random.randn(10000)
```

```
plt.hist(data)
```

```
plt.show()
```



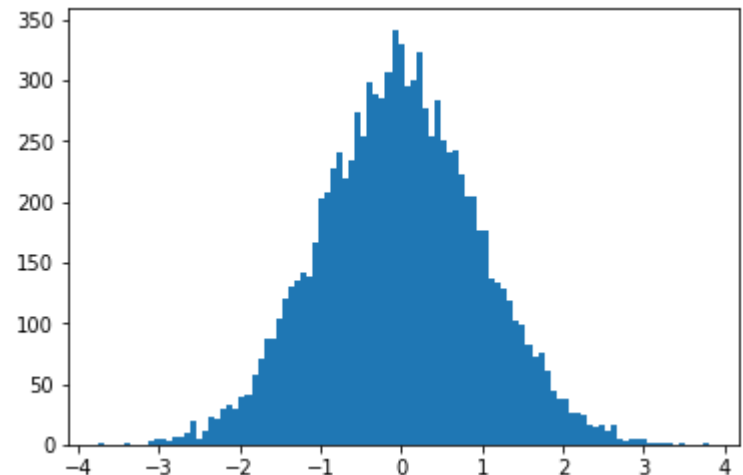
ピン数(階級)を増やす

```
import numpy as np  
import matplotlib.pyplot as plt  
%matplotlib inline
```

```
np.random.seed(0)  
data = np.random.randn(10000)
```

```
plt.hist(data, bins=100)
```

```
plt.show()
```



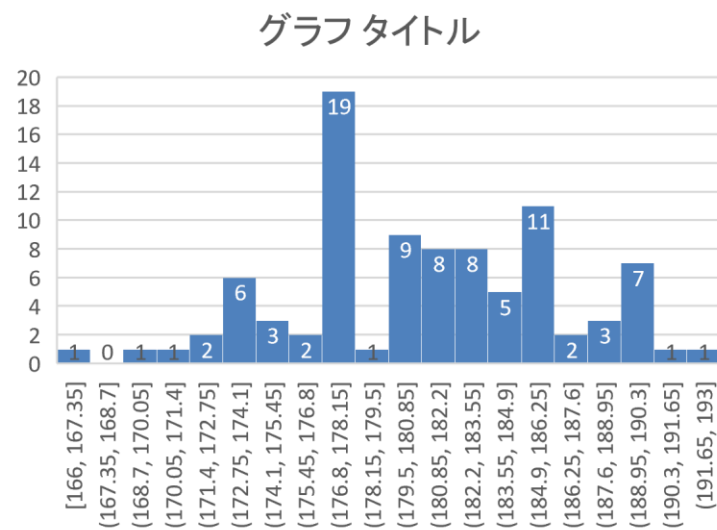
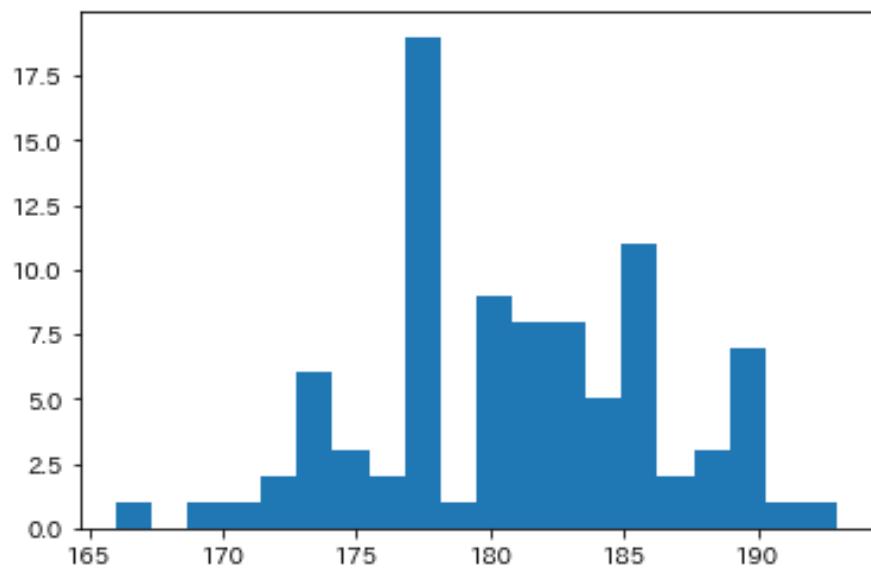
問題

- 巨人軍のデータから身長,体重のヒストグラムをつくりなさい

解答例

```
#-----  
#巨人軍の身長の一istogram  
#-----  
  
import pandas as pd  
import matplotlib.pyplot as plt  
%matplotlib inline  
df = pd.read_csv('g.csv')  
shisyo = np.array  
shisyo = df.values  
height=[]  
for s in shisyo:  
    #print(s[3])  
    height.append(s[3])  
plt.hist(height, bins=20)  
  
plt.show()
```

表示例



12.5 円グラフ

12.5.1 円グラフ

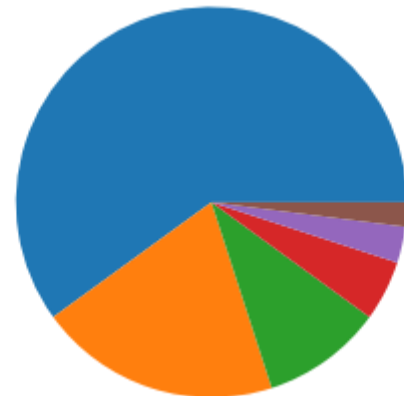
```
import matplotlib.pyplot as plt  
%matplotlib inline  
data = [60, 20, 10, 5, 3, 2]
```

```
plt.pie(data)
```

```
# 円グラフを円楕円から真円にしてください
```

```
plt.axis("equal")
```

```
plt.show()
```

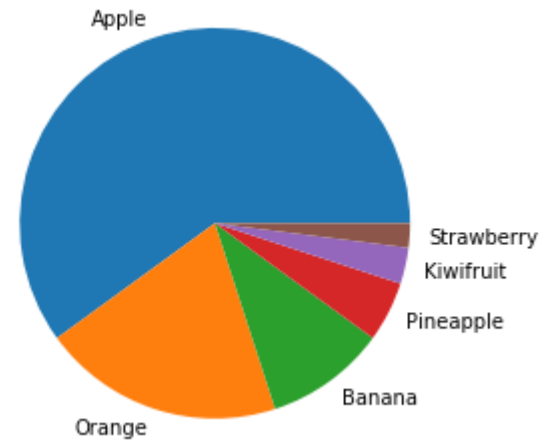


円グラフにラベルを設定する

```
data = [60, 20, 10, 5, 3, 2]
```

```
labels = ["Apple", "Orange", "Banana", "Pineapple", "Kiwifruit", "Strawberry"]
```

```
plt.pie(data, labels=labels)
```



12.5.3特定の要素を目立たせる

```
data = [60, 20, 10, 5, 3, 2]
```

```
labels = ["Apple", "Orange", "Banana", "Pineapple", "Kiwifruit", "Strawberry"]
```

```
explode = [0, 0, 0.1, 0, 0, 0]
```

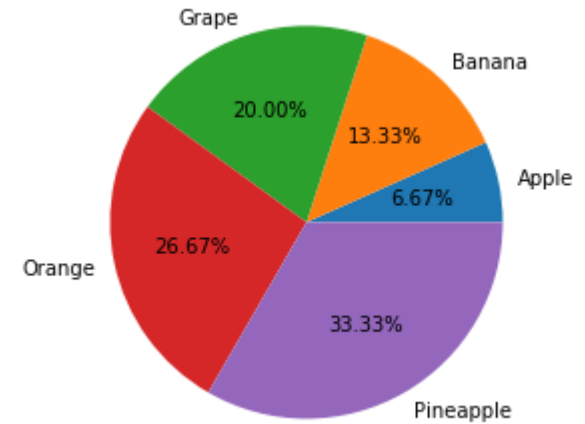
```
plt.pie(data, labels=labels, explode=explode)
```

円グラフ

```
%matplotlib inline
import matplotlib.pyplot as plt
```

```
values = [100, 200, 300, 400, 500] # グラフ要素の値
labels = [                                # グラフ要素のラベル
    'Apple', 'Banana', 'Grape', 'Orange', 'Pineapple'
]
```

```
plt.pie(x=values,                # グラフ要素の値を設定
        labels=labels,          # グラフ要素のラベルを設定
        autopct='%.2f%%')      # 構成割合として小数点以下2桁までをプロット
plt.axis('equal')               # グラフを
plt.show()
```



時計回りから表示

```
plt.pie(x=values,  
        labels=labels,  
        autopct='%.2f%%',  
        startangle=90,  
        counterclock=False  
    )
```

```
plt.axis('equal')
```

```
plt.show()
```

グラフ要素の値を設定

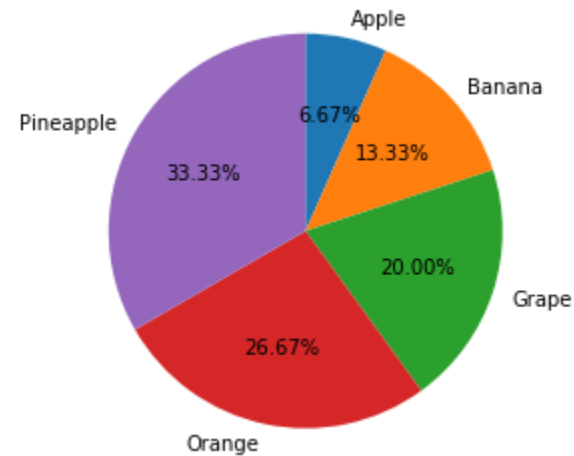
グラフ要素のラベルを設定

構成割合として小数点以下2桁までをプロット

90度(真上)の位置から開始

時計回りにする

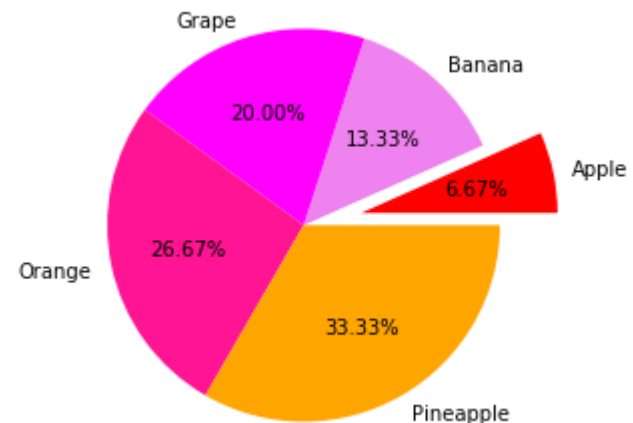
グラフを真円にする




```
# 要素のカラーを指定するリスト エッジライン
setcolors = ['red', 'violet', 'fuchsia', 'deeppink', 'orange']
plt.pie(x=values,          # グラフ要素の値を設定
        labels=labels,     # グラフ要素のラベルを設定
        colors=setcolors,  # グラフ要素のカラーを設定
        wedgeprops={
            'linewidth': 3,  # エッジラインの幅は3
            'edgecolor': 'white' # エッジラインの色はホワイト
        },
        labeldistance=0.5,  # ラベルを円周内の50%の位置に表示
        textprops={
            'color': 'white', # ラベルテキストのカラーはホワイト
            'weight': 'bold'} # 太字にする
    )
plt.axis('equal')          # グラフを真円にする
plt.show()
```



```
plt.pie(x=values,          # グラフ要素の値を設定
        labels=labels,     # グラフ要素のラベルを設定
        autopct='%0.2f%%', # 構成割合として小数点以下2桁までをプロット
        colors=setcolors,  # グラフ要素のカラーを設定
        explode=[0.3, 0, 0, 0, 0] # 1番目の要素の中心位置を円周上から0.3にする
    )
plt.axis('equal')          # グラフを真円する
plt.show()
```



問題

1, TestDb.dbのテーブルplayerの

(1) 体重70kg未満と

(2) 70kgから79kg

(3) 80kgから89kg

(4) 90kgから99kg

(5) 100kg以上の人数を求めて円グラフを書いて
みてください。SQLで検索してください

SQL(engraf1.py)

select count(*) from player where 体重<70; 22人

select count(*) from player where 体重>=70 and
体重<=79; 206人

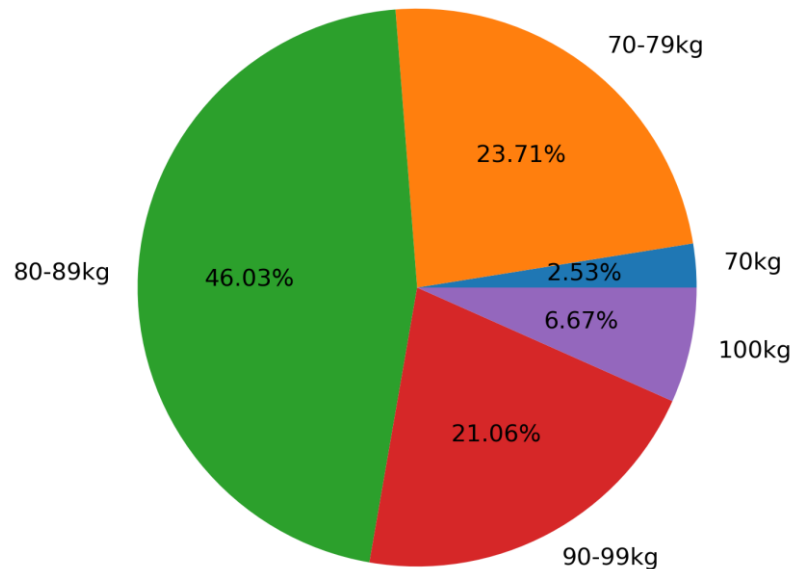
select count(*) from player where 体重>=80 and
体重<=89; 400人

select count(*) from player where 体重>=90 and
体重<=99; 183人

select count(*) from player where 体重>=100
; 58人

課題(コードなし)

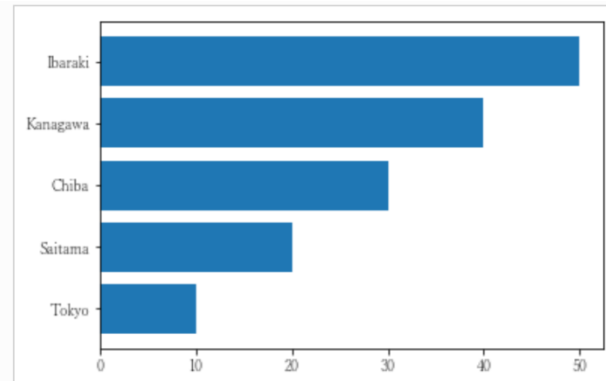
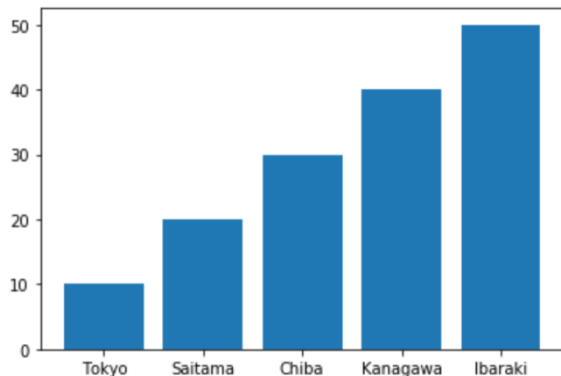
先ほどのSQLをコードに埋め込んで
人数を自動的に算出してグラフを書く方法を
考えてみてください。



12.2棒グラフ

棒グラフ

```
import matplotlib.pyplot as plt  
plt.bar( ['Tokyo', 'Saitama', 'Chiba', 'Kanagawa',  
          'Ibaraki'], [10, 20, 30, 40, 50] )  
# barh()にすると棒グラフが横になる  
plt.show()
```



複数の棒グラフ

```
xx1 = [1, 2, 3]
```

```
yy1 = [4, 5, 6]
```

```
xx2 = [1.3, 2.3, 3.3]
```

```
yy2 = [2, 4, 1]
```

```
label_x = ['Result1', 'Result2', 'Result3']
```

```
# 1つ目の棒グラフ
```

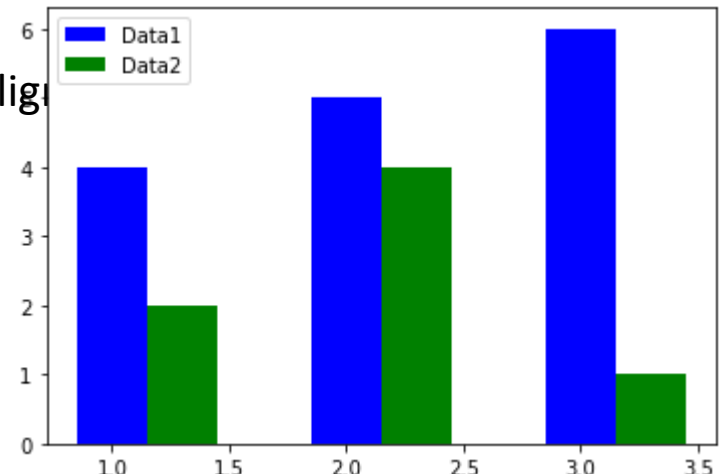
```
plt.bar(xx1, yy1, color='b', width=0.3, label='Data1', align="center")
```

```
# 2つ目の棒グラフ
```

```
plt.bar(xx2, yy2, color='g', width=0.3, label='Data2', align="center")
```

```
# 凡例
```

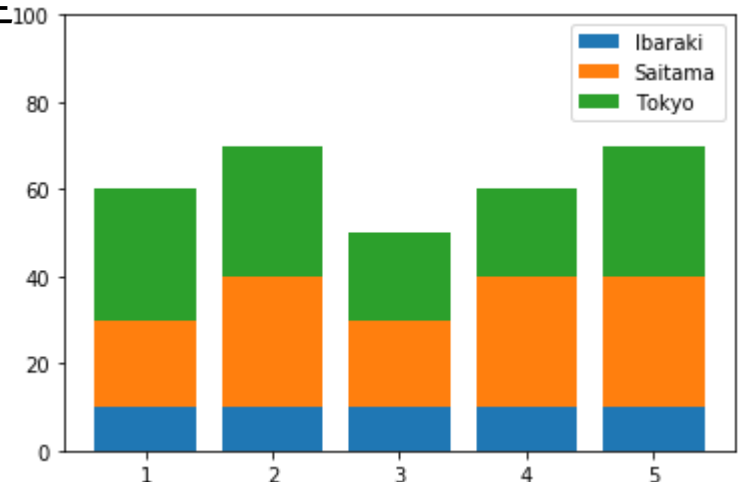
```
plt.legend(loc=2)
```



複数の棒グラフ

```
import numpy as np
import matplotlib.pyplot as plt
x_list = [0, 1, 2, 3, 4] # 目盛りの値の設定
y_list = np.arange(0, 101, 20)
tokyo = [30, 30, 20, 20, 30] # グラフの値
saitama = [20, 30, 20, 30, 30]
ibaraki = [10, 10, 10, 10, 10]
tokyo_bottom = np.array(saitama) + np.array(ibaraki) # 一番に積み上がる棒グラフの
    かさ上げ用
plt.bar(x_list, ibaraki, label = 'Ibaraki') # グラフデータの設定
plt.bar(x_list, saitama, label = 'Saitama', bottom = ibaraki)
plt.bar(x_list, tokyo, label = 'Tokyo' , bottom = tokyo_bottom)
plt.xticks(x_list, (['1', '2', '3', '4', '5'])) # 目盛りの設定
plt.yticks(y_list)
plt.legend() # 凡例の表示
plt.show()# グラフの表示
```

Tokyo
Saitama
Ibaraki



- 下からtokyo_bottom , saitama, ibaraki

tokyo = [30, 30, 20, 20, 30]

saitama = [20, 30, 20, 30, 30]

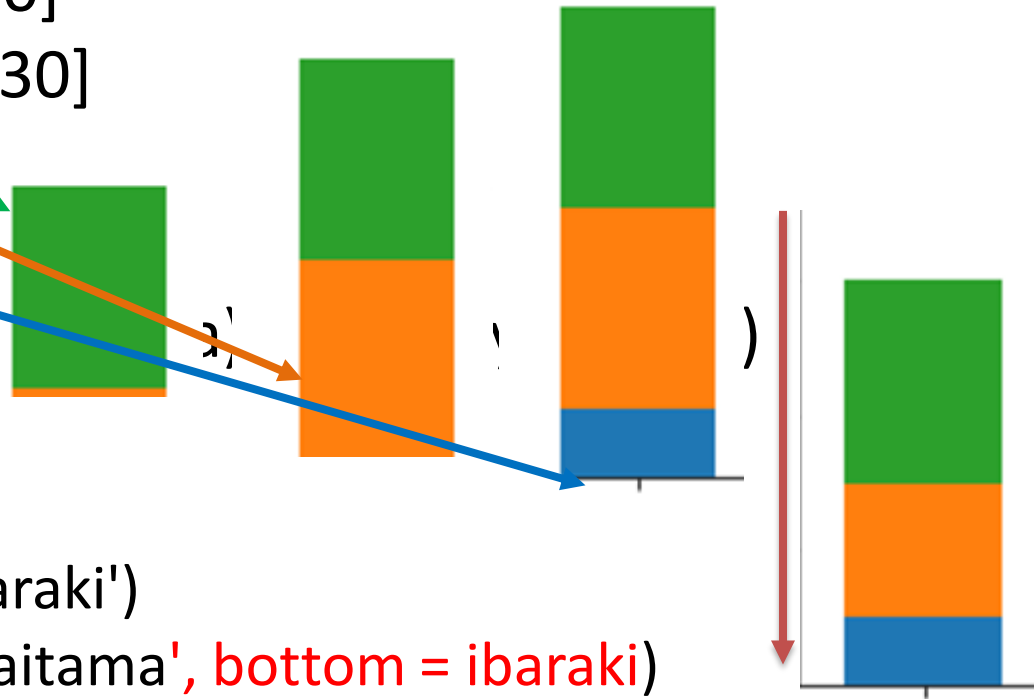
ibaraki = [10, 10, 10, 10, 10]

- tokyo_bottom = np.array

```
plt.bar(x_list, ibaraki, label = 'Ibaraki')
```

```
plt.bar(x_list, saitama, label = 'Saitama', bottom = ibaraki)
```

```
plt.bar(x_list, tokyo, label = 'Tokyo' , bottom = tokyo_bottom)
```



Pandasを使った場合

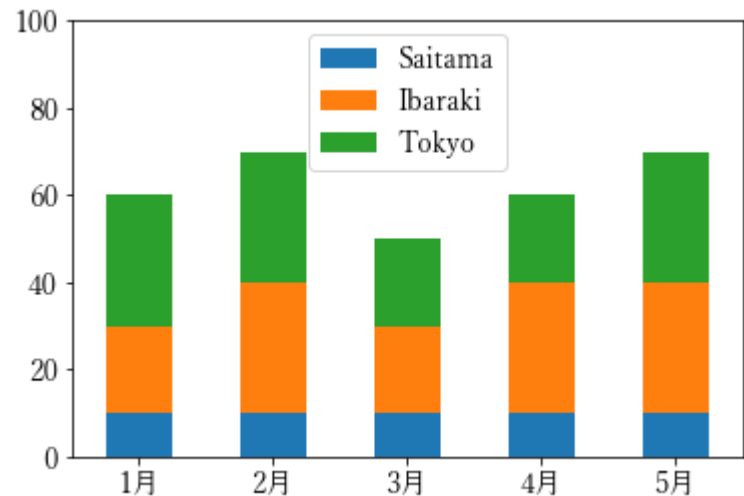
```
import pandas as pd
import matplotlib.pyplot as plt

plt.rcParams["font.family"] = 'Yu Mincho'

df = pd.DataFrame(
    [
        [10, 20, 30],
        [10, 30, 30],
        [10, 20, 20],
        [10, 30, 20],
        [10, 30, 30]
    ],
    index = ['1月', '2月', '3月', '4月', '5月'],
    columns = ['Saitama', 'Ibaraki', 'Tokyo']
)

df.plot.bar(stacked = True, yticks = range(0, 120, 20), rot = 0)
#df.plot(kind = 'bar', yticks = range(0, 81, 10))
#df.plot(kind = 'line', yticks = range(0, 51, 10))

plt.show()
```



課題

次の表はある中学の実力テスト分布表です。科目別の棒グラフを5つ表示してください。(リストからでもいいです)

	国語	社会	数学	理科	英語
90-100	4	0	8	2	2
80-89	7	17	19	10	13
70-79	18	13	17	6	12
60-69	28	12	23	13	14
50-59	36	17	16	14	26
40-49	27	17	18	20	21
30-39	10	22	17	24	16
0-29	7	39	19	48	33

複数のグラフ

複数のグラフ

```
import matplotlib.pyplot as plt
x_list = range(0, 5)
plt.plot(x_list, [40, 45, 35, 30, 20], 'b' , label = 'Tokyo') #折れ線グラフ
plt.plot(x_list, [10, 15, 30, 35, 35], '-.g', label = 'Saitama') #棒グラフ
plt.plot(x_list, [ 5, 10, 20, 40, 50], '--r', label = 'Ibaraki') #棒グラフ
plt.bar (x_list, [ 5, 10, 20, 25, 30]) #棒グラフ
plt.xticks(x_list, (['1月', '2月', '3月', '4月', '5月']))
plt.legend()
plt.show()
```

