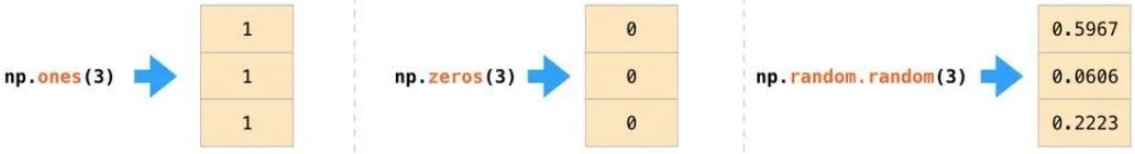
创建数组

我们可以通过传递一个 python 列表并使用 np.array（）来创建 NumPy 数组（极大可能是多维数组）。在本例中，python 创建的数组如下图右所示：

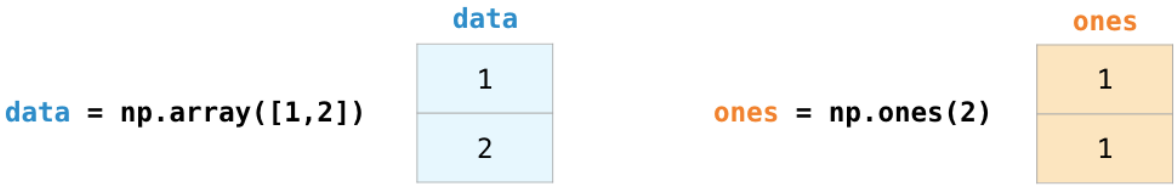
通常我们希望 NumPy 能初始化数组的值，为此 NumPy 提供了 ones()、zeros() 和 random.random() 等方法。我们只需传递希望 NumPy 生成的元素数量即可：



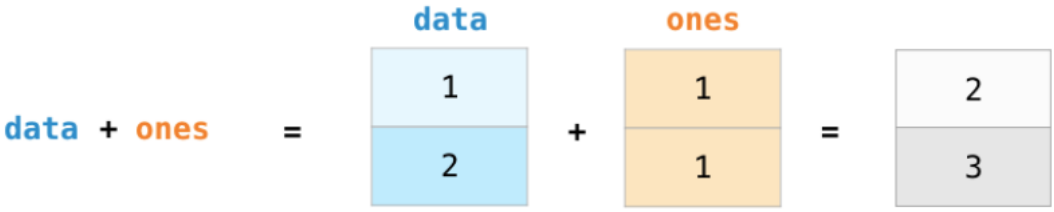
一旦创建了数组，我们就可以尽情对它们进行操作。

数组运算

让我们创建两个 NumPy 数组来展示数组运算功能。我们将下图两个数组称为 data 和 ones：

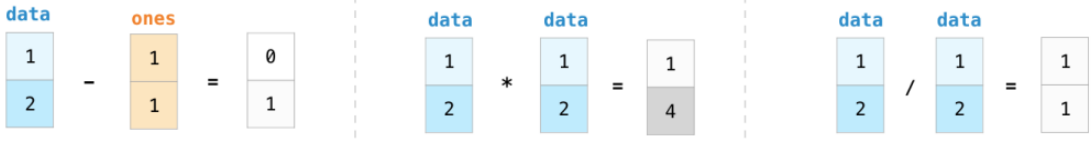


将它们按位置相加（即每行对应相加），直接输入 data + ones 即可：



当我开始学习这些工具时，我发现这样的抽象让我不必在循环中编写类似计算。此类抽象可以使我在更高层面上思考问题。

除了「加」，我们还可以进行如下操作：

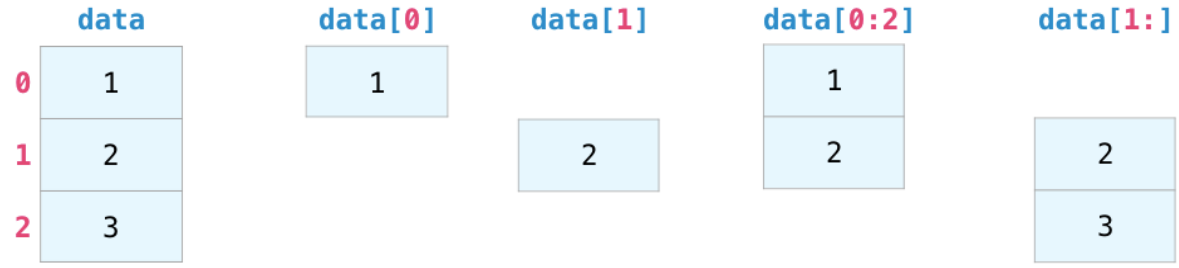


通常情况下，我们希望数组和单个数字之间也可以进行运算操作（即向量和标量之间的运算）。比如说，我们的数组表示以英里为单位的距离，我们希望将其单位转换为千米。只需输入 data * 1.6 即可：

看到 NumPy 是如何理解这个运算的了吗？这个概念叫做广播机制（broadcasting），它非常有用。

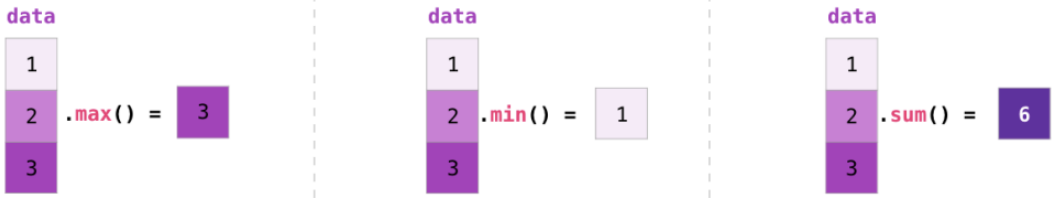
索引

我们可以我们像对 python 列表进行切片一样，对 NumPy 数组进行任意的索引和切片：



聚合

NumPy 还提供聚合功能：



除了 min、max 和 sum 之外，你还可以使用 mean 得到平均值，使用 prod 得到所有元素的乘积，使用 std 得到标准差等等。

更多维度

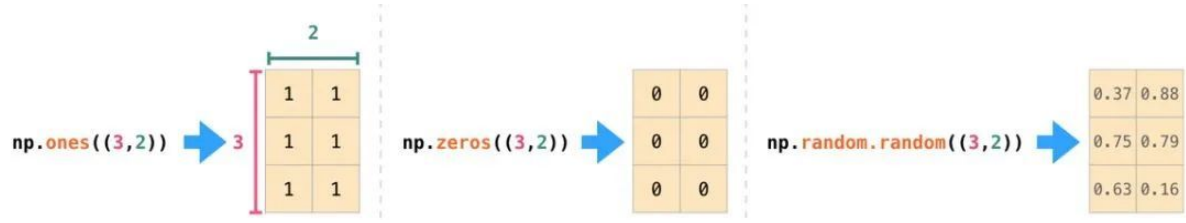
上述的例子都在一个维度上处理向量。NumPy 之美的关键在于，它能够将上述所有方法应用到任意数量的维度。

创建矩阵

我们可以传递下列形状的 python 列表，使 NumPy 创建一个矩阵来表示它：

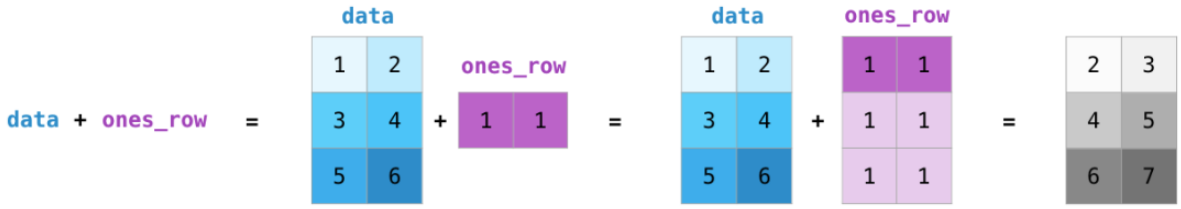
```
1 np.array([[1,2],[3,4]])
```

我们也可以使用上面提到的方法（ones()、zeros() 和 random.random()），只要写入一个描述我们创建的矩阵维数的元组即可：



矩阵运算

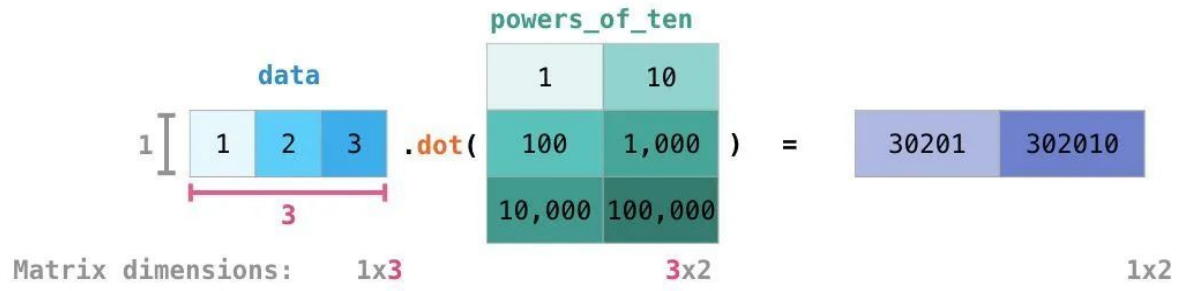
如果两个矩阵大小相同，我们可以使用算术运算符（+-*/）对矩阵进行加和乘。NumPy 将它们视为 position-wise 运算：



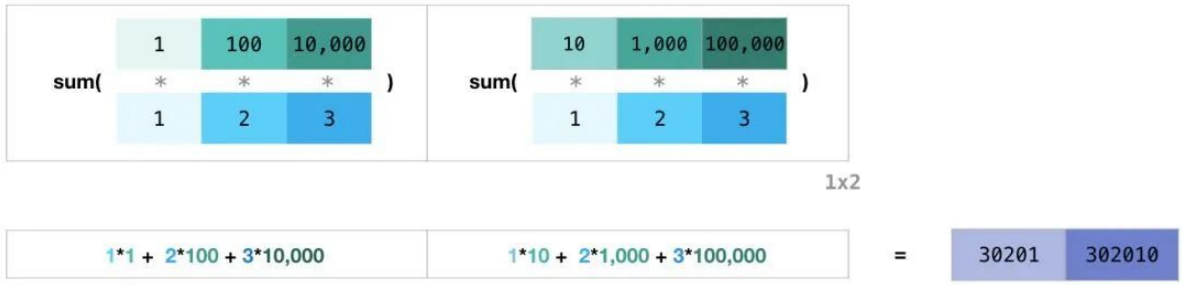
我们也可以对不同大小的两个矩阵执行此类算术运算，但前提是某一个维度为 1（如矩阵只有一列或一行），在这种情况下，NumPy 使用广播规则执行算术运算：

点乘

算术运算和矩阵运算的一个关键区别是矩阵乘法使用点乘。NumPy 为每个矩阵赋予 dot() 方法，我们可以用它与其他矩阵执行点乘操作：

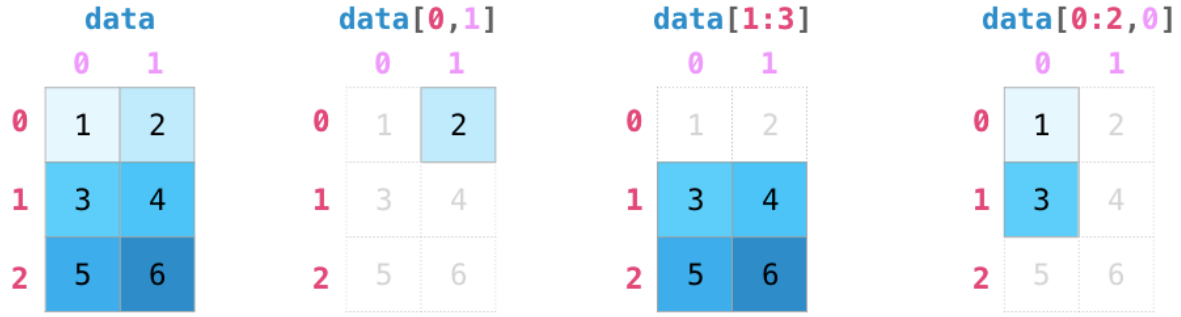


我在上图的右下角添加了矩阵维数，来强调这两个矩阵的临近边必须有相同的维数。你可以把上述运算视为：



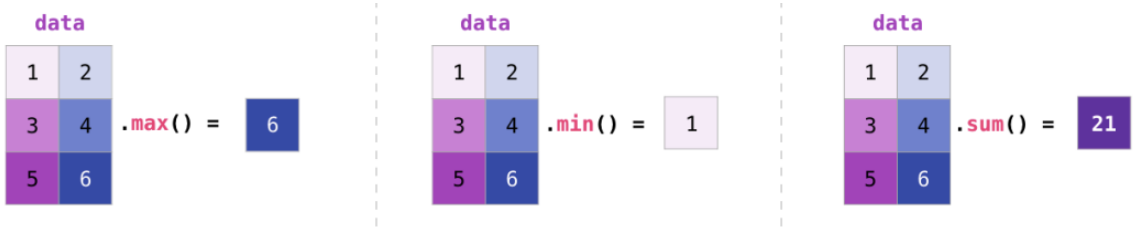
矩阵索引

当我们处理矩阵时，索引和切片操作变得更加有用：

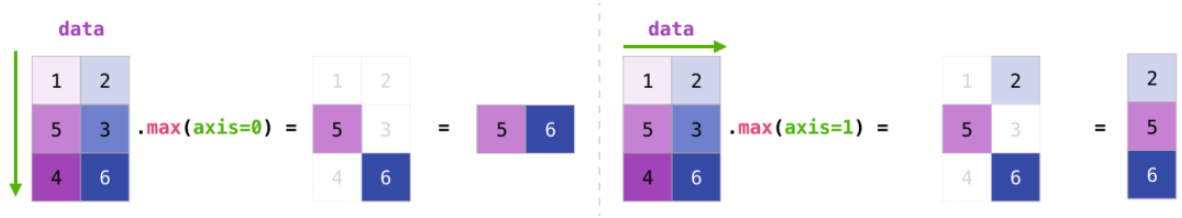


矩阵聚合

我们可以像聚合向量一样聚合矩阵：



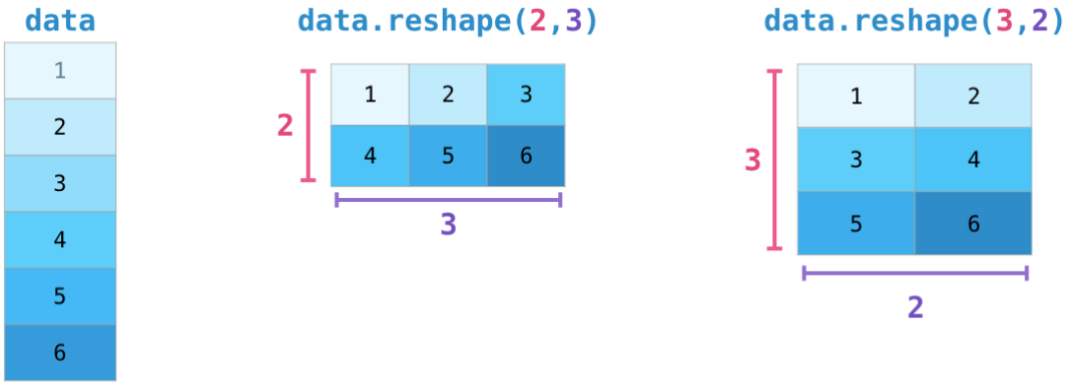
我们不仅可以聚合矩阵中的所有值，还可以使用 axis 参数执行跨行或跨列聚合：



转置和重塑

处理矩阵时的一个常见需求是旋转矩阵。当需要对两个矩阵执行点乘运算并对齐它们共享的维度时，通常需要进行转置。NumPy 数组有一个方便的方法 T 来求得矩阵转置：

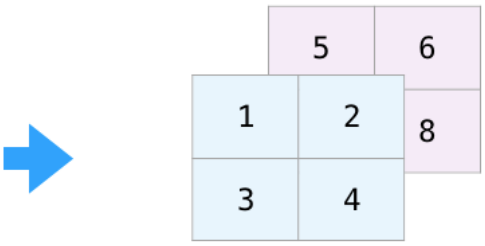
在更高级的实例中，你可能需要变换特定矩阵的维度。在机器学习应用中，经常会这样：某个模型对输入形状的要求与你的数据集不同。在这些情况下，NumPy 的 reshape() 方法就可以发挥作用了。只需将矩阵所需的新维度赋值给它即可。可以为维度赋值-1，NumPy 可以根据你的矩阵推断出正确的维度：



再多维度

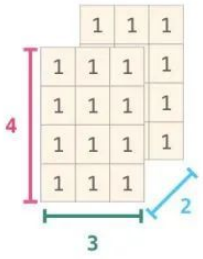
NumPy 可以在任意维度实现上述提到的所有内容。其中心数据结构被叫作 ndarray（N 维数组）不是没道理的。

```
np.array([ [[1,2],[3,4]],
           [[5,6],[7,8]] ])
```

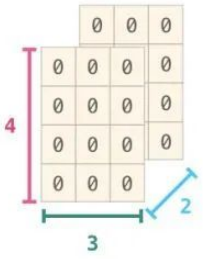


在很多情况下，处理一个新的维度只需在 NumPy 函数的参数中添加一个逗号：

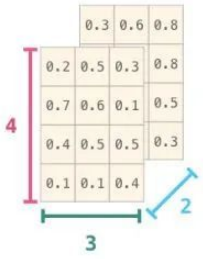
```
np.ones((4,3,2))
```



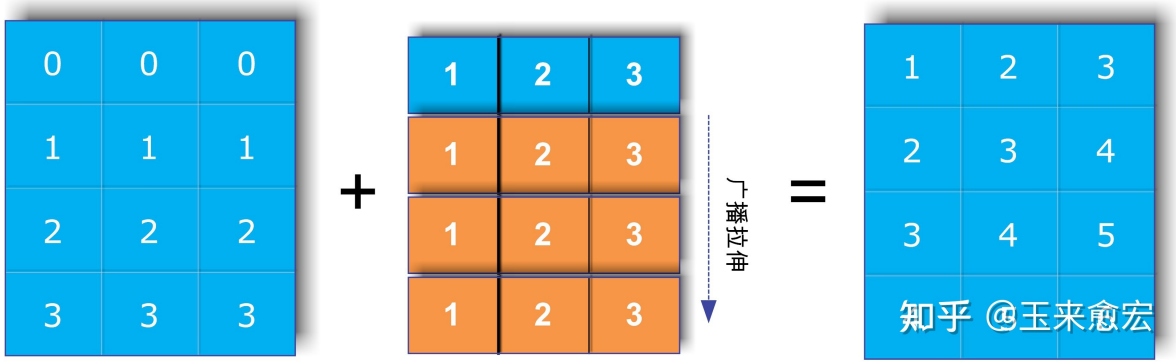

```
np.zeros((4,3,2))
```

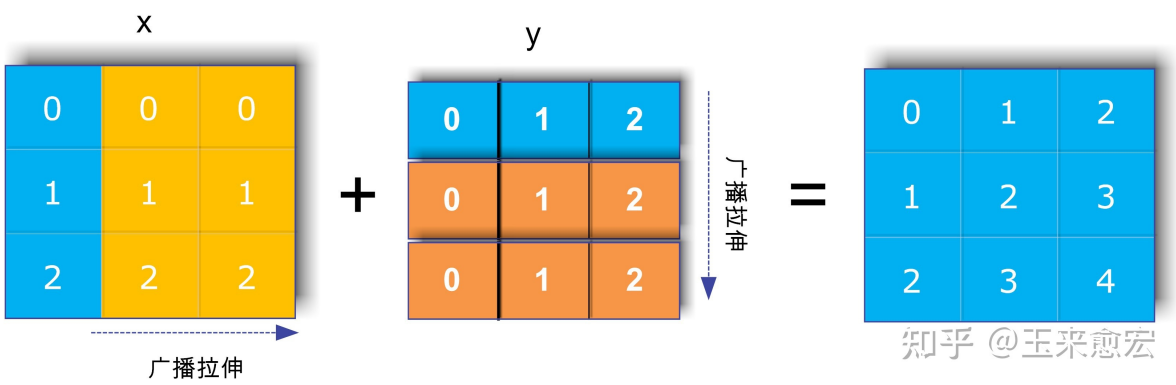


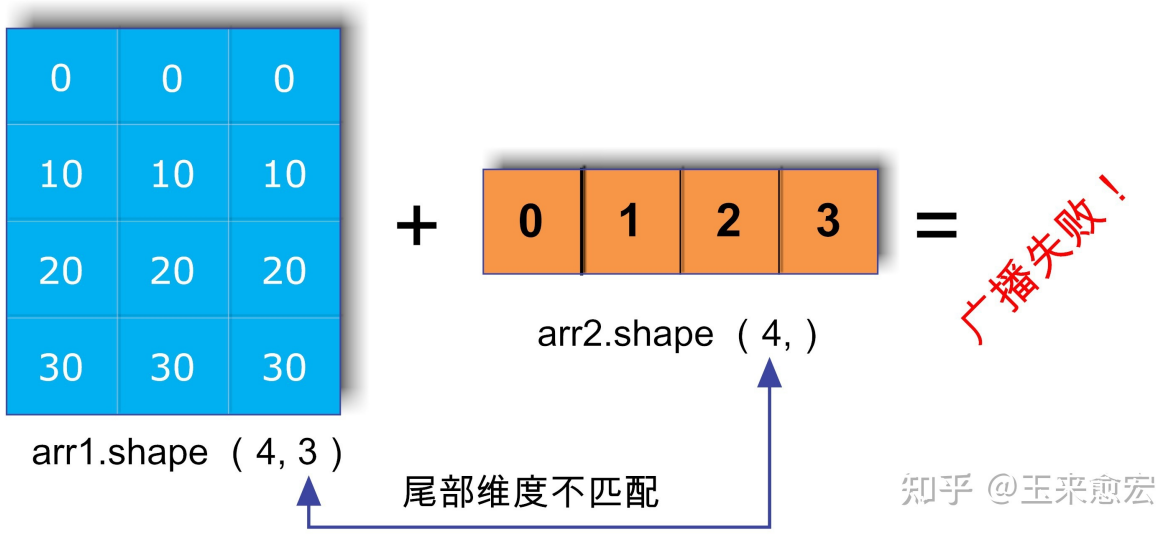

```
np.random.random((4,3,2))
```



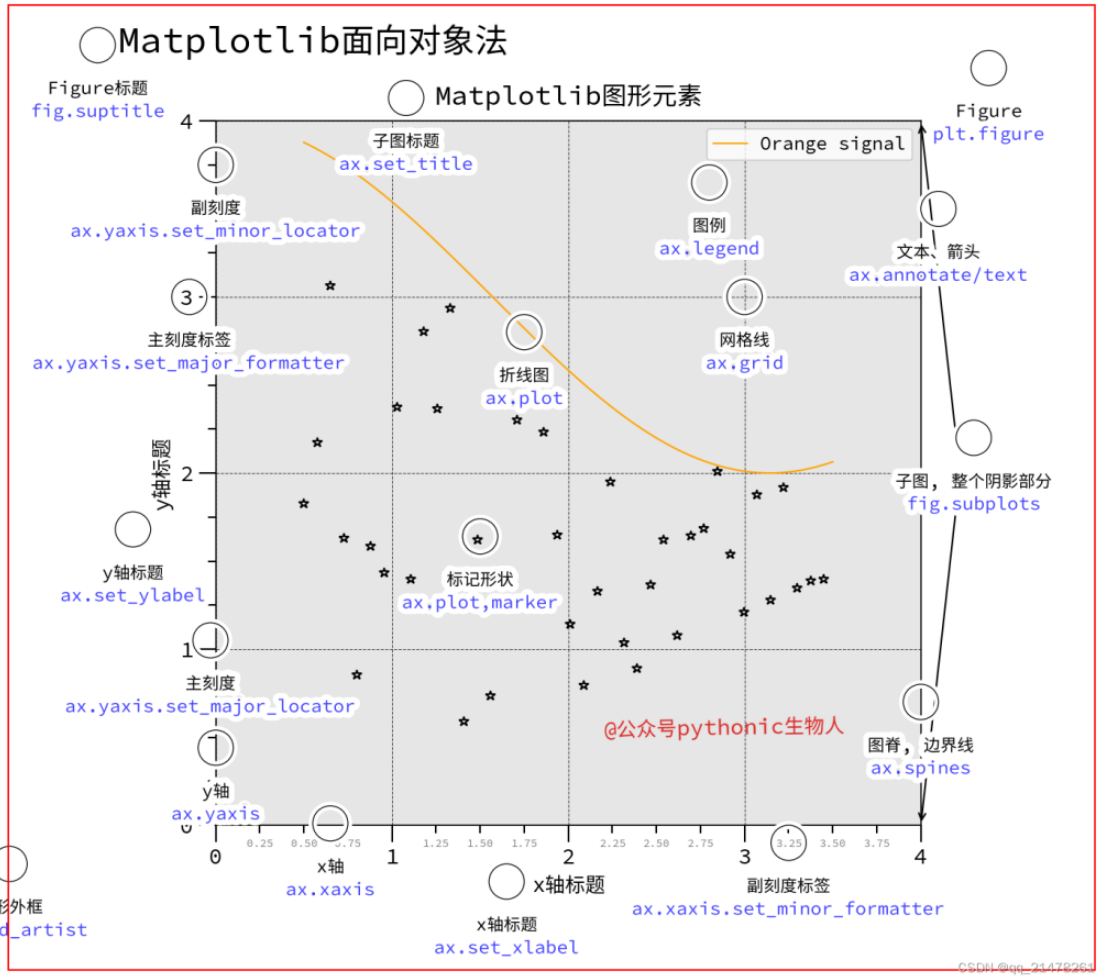
广播







画图




```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
'''
```

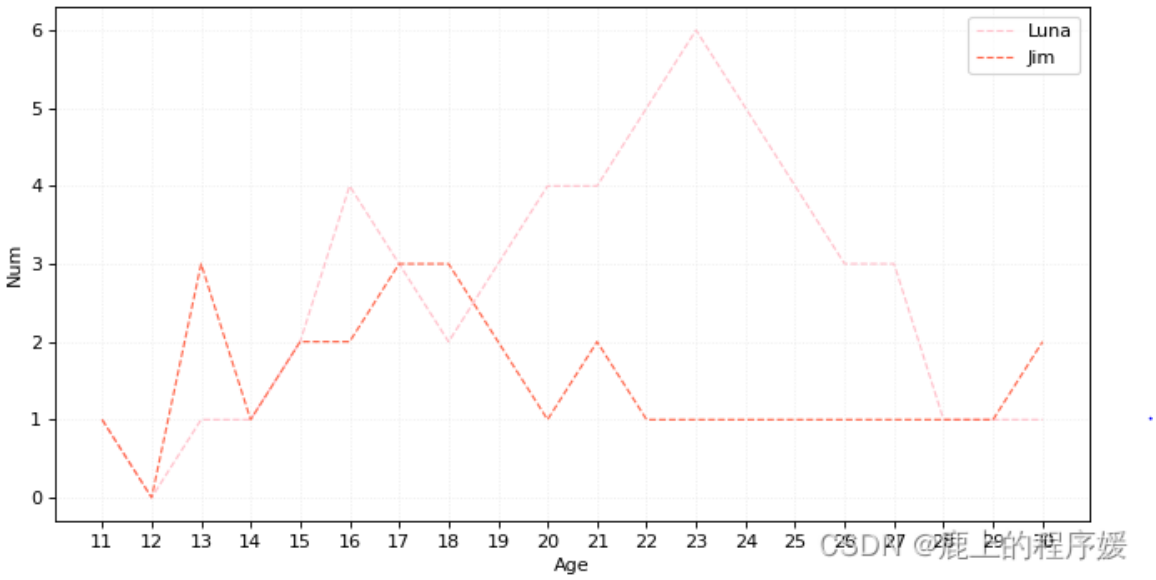
```
转载请标明来源！转载请标明来源！转载请标明来源！
@Time    :   2022年五一劳动节
@Author  :   matplotlib.org，公众号:pythonic生物人
@Contact :   公众号:pythonic生物人
@Desc    :   图解Matplotlib面向对象方法
'''

# 导入模块
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle, Rectangle
from matplotlib.patheffects import withStroke
from matplotlib.ticker import AutoMinorLocator, MultipleLocator

# 指定字体
from mplfonts import use_font

use_font('Source Han Mono SC')

# 添加画布Figure，图中红框包围的部分为一个Figure
fig = plt.figure(figsize=(9, 8), facecolor='1', dpi=150)

# 为Figure添加标题
fig.suptitle('Matplotlib面向对象法', x=0.46, fontsize=20, ha='right')

# 在Figure上添加子图Axes
marg = 0.15
ax = fig.add_axes([marg, marg, 1 - 1.8 * marg, 1 - 1.8 * marg],
                  aspect=1,
                  facecolor='0.9')

# 准备绘图数据
np.random.seed(19680801)
X = np.linspace(0.5, 3.5, 120)
Y1 = 3 + np.cos(X)
Y2 = 1 + np.cos(1 + X / 0.75) / 2
Y3 = np.random.uniform(Y1, Y2, len(X))

# 同一个axes上绘图
ax.plot(X, Y1, c='orange', lw=1, label="Orange signal", zorder=10)
ax.plot(X[::3],
        Y3[::3],
        linewidth=0,
        markersize=6,
        marker='*',
        markerfacecolor='none',
        markeredgecolor='black',
        markeredgewidth=1)

# 设置子图标题
ax.set_title("Matplotlib图形元素", fontsize=15, verticalalignment='bottom')

# 设置图例
ax.legend(loc="upper right", fontsize=10)

```

```python
# 设置坐标轴标题
ax.set_xlabel("x轴标题", fontsize=12)
ax.set_ylabel("y轴标题", fontsize=12)

# 设置x，y轴刻度间隔
ax.xaxis.set_major_locator(MultipleLocator(1.000))  # x轴主刻度间隔
ax.xaxis.set_minor_locator(AutoMinorLocator(4))  # x轴副刻度间隔

ax.yaxis.set_major_locator(MultipleLocator(1.000))
ax.yaxis.set_minor_locator(AutoMinorLocator(4))


# 设置x轴副刻度格式
def minor_tick(x, pos):
    if not x % 1.0:
        return ""
    return f"{x:.2f}"


ax.xaxis.set_minor_formatter(minor_tick)

# 设置x，y轴刻度范围
ax.set_xlim(0, 4)
ax.set_ylim(0, 4)

# 设置x，y轴刻度字号、颜色等
ax.tick_params(which='major', width=1.0, labelsize=12)
ax.tick_params(which='major', length=10, labelsize=12)
ax.tick_params(which='minor', width=1.0, labelsize=10)
ax.tick_params(which='minor', length=5, labelsize=6, labelcolor='0.5')

# 设置网格线
ax.grid(linestyle="--", linewidth=0.5, color='.25', zorder=-10)

# 文本、箭头
ax.annotate(
    "",
    xy=(4, 4),
    xytext=(4.2, 2.2),
    color=(0.25, 0.25, 1.00),
    weight="regular",
    arrowprops=dict(arrowstyle="->", connectionstyle="arc3",
color="black"),
)

ax.annotate(
    "",
    xy=(4, 0),
    xytext=(4.2, 1.8),
    color=(0.25, 0.25, 1.00),
    weight="regular",
    arrowprops=dict(arrowstyle="->", connectionstyle="arc3",
color="black"),
)

```

```
# 矩形外框
fig.add_artist(
    Rectangle((0, 0),
              width=1,
              height=1,
              facecolor='none',
              edgecolor='red',
              linewidth=1.0))


# 图中添加圆圈注释
def just_circle(x, y, radius=0.15):
    c = Circle((x, y),
               radius,
               clip_on=False,
               zorder=10,
               linewidth=0.6,
               edgecolor='black',
               facecolor='none',
               path_effects=[withStroke(linewidth=5, foreground=(1, 1, 1, 1))])
    ax.add_artist(c)


# 图中添加文本注释
def text(x, y, text):
    ax.text(x,
            y,
            text,
            zorder=100,
            ha='center',
            va='top',
            weight='bold',
            color='black',
            style='italic',
            path_effects=[withStroke(linewidth=7, foreground=(1, 1, 1, 1))])


# 图中添加Matplotlib对应方法文本
def code(x, y, text):
    ax.text(x,
            y,
            text,
            zorder=100,
            ha='center',
            va='top',
            weight='normal',
            color=(0.25, 0.25, 1.00),
            fontsize='medium',
            path_effects=[withStroke(linewidth=7, foreground=(1, 1, 1, 1))])


def circle(x, y, txt, cde, radius=0.1):
```

```
    just_circle(x, y, radius=radius)
    text(x, y - 0.2, txt)
    code(x, y - 0.33, cde)


circle(4.385, 4.3, "Figure", "plt.figure")
circle(4.3, 2.2, "子图，整个阴影部分", "fig.subplots")

circle(-0.67, 4.43, "Figure标题", "fig.suptitle")
circle(1.08, 4.13, "子图标题", "ax.set_title")

circle(1.75, 2.80, "折线图", "ax.plot")
circle(1.5, 1.64, "标记形状", "ax.plot,marker")
circle(3.00, 3.00, "网格线", "ax.grid")
circle(2.8, 3.65, "图例", "ax.legend")

circle(-0.03, 1.05, "主刻度", "ax.yaxis.set_major_locator")
circle(-0.15, 3.00, "主刻度标签", "ax.yaxis.set_major_formatter")
circle(0.00, 3.75, "副刻度", "ax.yaxis.set_minor_locator")
circle(3.25, -0.10, "副刻度标签", "ax.xaxis.set_minor_formatter")

circle(0.65, 0.01, "x轴", "ax.xaxis")
circle(0, 0.44, "y轴", "ax.yaxis")
circle(1.650, -0.32, "x轴标题", "ax.set_xlabel")
circle(-0.47, 1.68, "y轴标题", "ax.set_ylabel")

circle(4.0, 0.7, "图脊，边界线", "ax.spines")
circle(-1.17, -0.22, "矩形外框", "fig.add_artist")

circle(4.1, 3.5, "文本、箭头", "ax.annotate/text")

plt.show()
```

1.绘制折线图（pyplot.plot(x,y)）




```
from matplotlib import pyplot


x = range(2,28,2)
y = [15,13,14,17,20,25,26,26,24,22,18,15,1]

//设置图片大小
fig = pyplot.figure(figsize=(5,5),dpi=80)
//绘图
pyplot.plot(x,y)
//保存图片
pyplot.savefig("./sig_size.png")
//展示图片
pyplot.show()
1234567891011121314
```

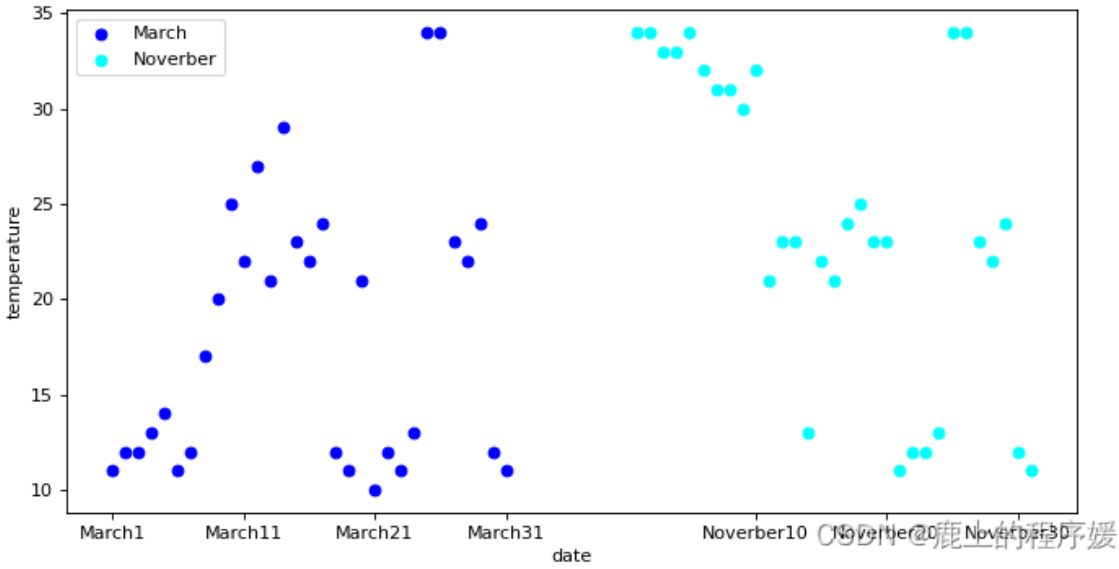
绘制两条折线

```
# import matplotlib
from matplotlib import pyplot

x = range(11,31)
y1 = [1,0,1,1,2,4,3,2,3,4,4,5,6,5,4,3,3,1,1,1]
y2 = [1,0,3,1,2,2,3,3,2,1,2,1,1,1,1,1,1,1,1,2]
#set the pic size
pyplot.figure(figsize=(10,5),dpi=80)

#ploting
pyplot.plot(x,y1,label="Luna",color="pink",linestyle='--',linewidth=1)
pyplot.plot(x,y2,label="Jim",color="tomato",linestyle='--',linewidth=1)

# set x/y-axis step
_xtick_labels = ["{}".format(i) for i in x]
pyplot.xticks(x,_xtick_labels)
pyplot.yticks(range(0,7))

# set x-asix desciption
```

```
pyplot.xlabel("Age")

# set y-asix description
pyplot.ylabel("Num")

# add legend
pyplot.legend()
# plot grid
pyplot.grid(alpha=0.2,linestyle=':')

# show
pyplot.show()
```

```
12345678910111213141516171819202122232425262728293031
```

2.绘制散点图（pyplot.scatter(x,y)）

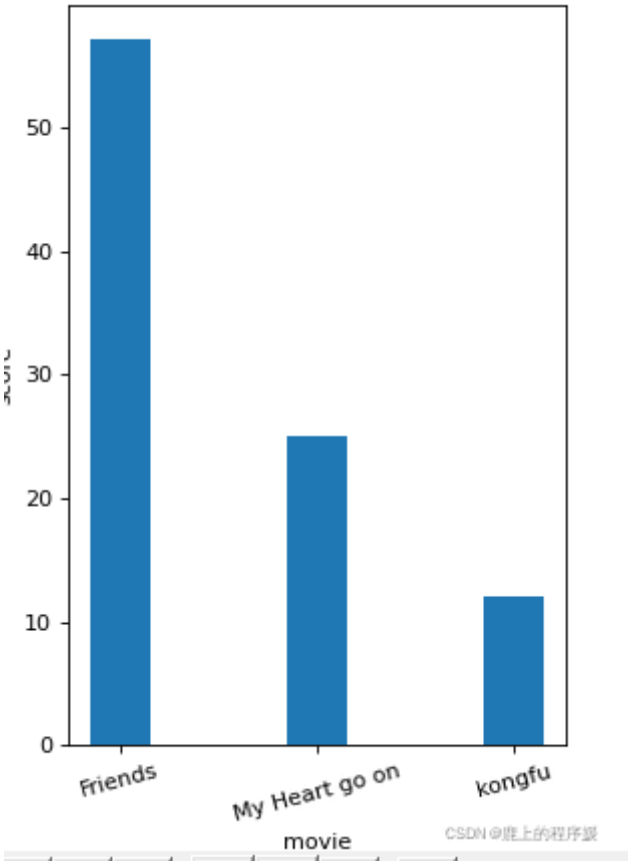



```
from matplotlib import pyplot
from matplotlib import font_manager

y_3 =
[11,12,12,13,14,11,12,17,20,25,22,27,21,29,23,22,24,12,11,21,10,12,11,13,34,34,23,22,24,12,11]
y_10 =
[34,34,33,33,34,32,31,31,30,32,21,23,23,13,22,21,24,25,23,23,11,12,12,13,34,34,23,22,24,12,11]

x_3 = range(1,32)
x_10 = range(41,72)

pyplot.figure(figsize=(20,10),dpi=80)

pyplot.scatter(x_3,y_3,color="blue",label="March")
pyplot.scatter(x_10,y_10,color="cyan",label="Noverber")

_x = list(x_3)+list(x_10)
_xticks_label = ["March{}".format(i) for i in x_3]
_xticks_label += ["Noverber{}".format(i-40) for i in x_10]
```

```
pyplot.xticks(_x[::10],_xticks_label[::10])
pyplot.xlabel("date")
pyplot.ylabel("temperature")
pyplot.legend()
pyplot.show()
```

```
123456789101112131415161718192021222 3
```

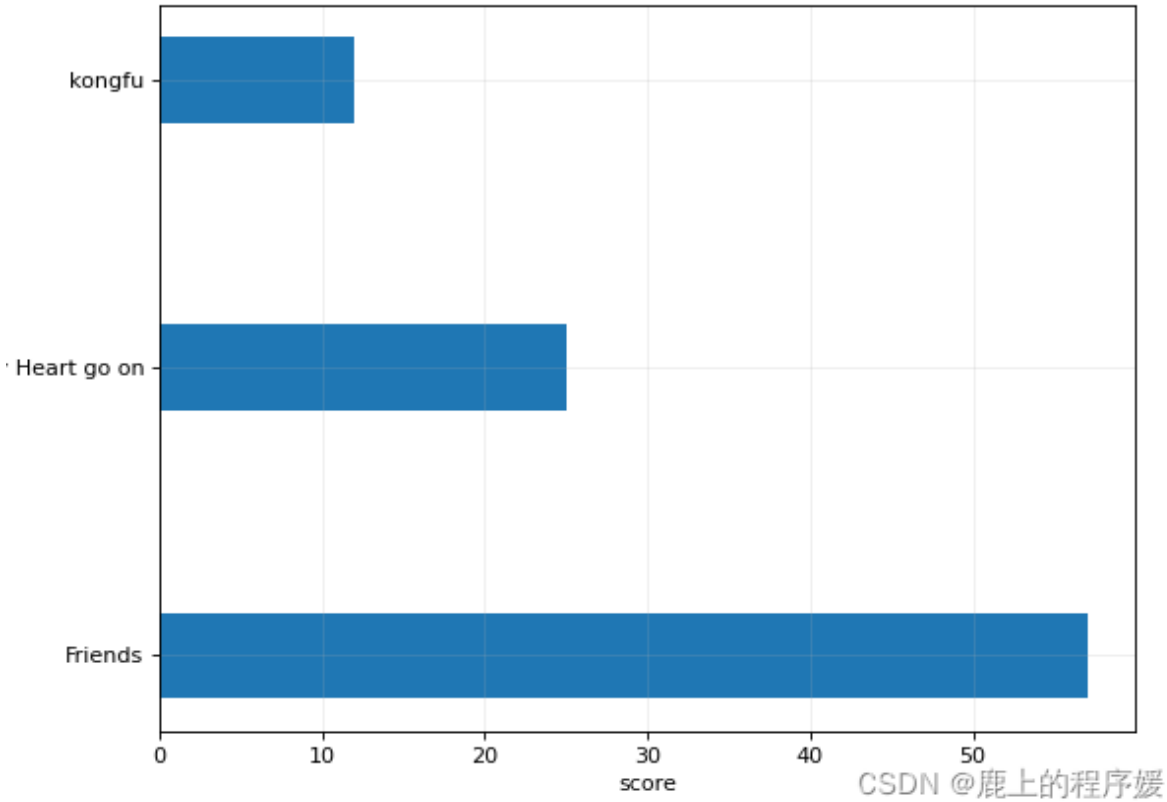
3.绘制条形图（pyplot.bar()）




```
from matplotlib import pyplot
from matplotlib import font_manager

my_font = font_manager.FontProperties(fname="")
a = ["Friends","My Heart go on","kongfu"]
b = [57.1,25,12]

pyplot.figure(figsize=(4,6),dpi=80)
pyplot.bar(a,b,width=0.3)
pyplot.xlabel("movie")
pyplot.ylabel("score")
pyplot.xticks(range(len(a)),a,rotation=15)
pyplot.show()
```

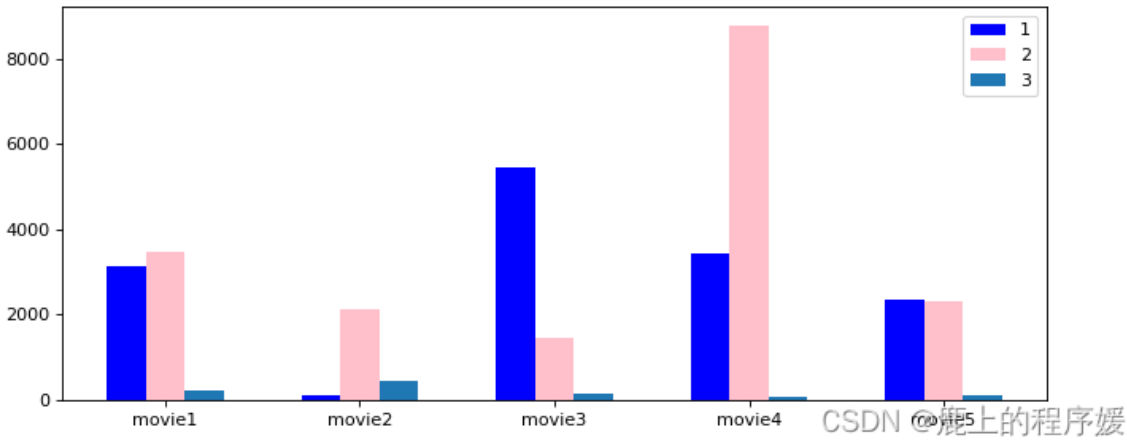
```
12345678910111213
```

pyplot.barh()=>绘制横着的条形图




```python
from matplotlib import pyplot
from matplotlib import font_manager

my_font = font_manager.FontProperties(fname="")
a = ["Friends","My Heart go on","kongfu"]
b = [57.1,25,12]

pyplot.figure(figsize=(8,6),dpi=80)
pyplot.barh(a,b,height=0.3)
pyplot.ylabel("movie")
pyplot.xlabel("score")
pyplot.yticks(range(len(a)),a)
pyplot.grid(alpha=0.2)
pyplot.show()
1234567891011121314
```

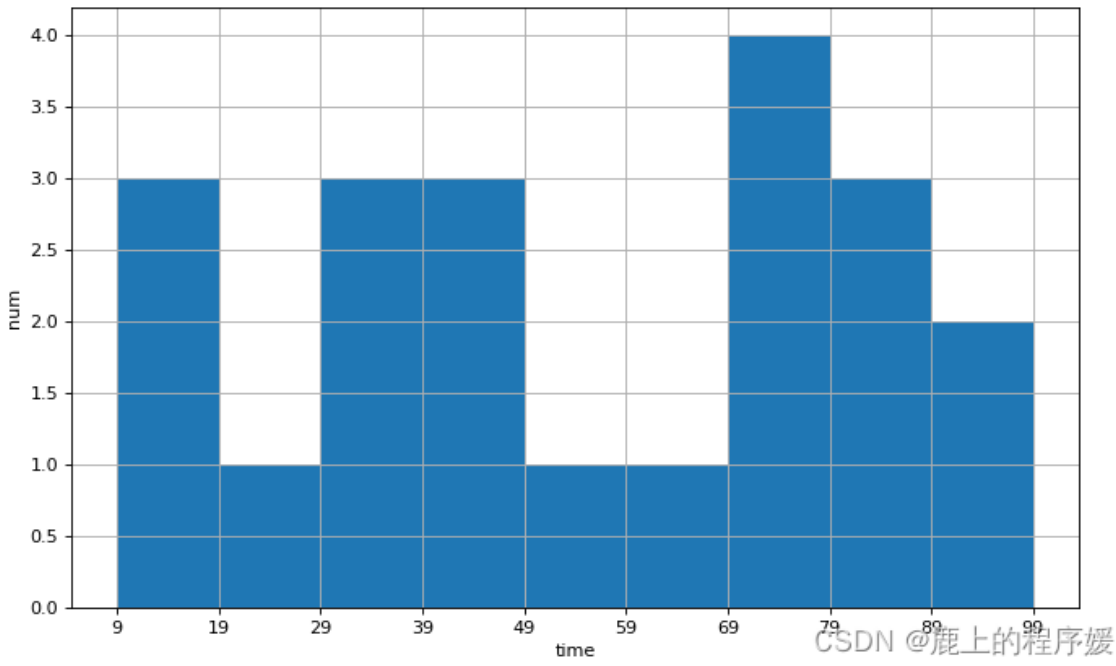
绘制对比条形图（绘制三次）



```python
from matplotlib import pyplot

a = ["movie1","movie2","movie3","movie4","movie5"]

b_1 = [3124,123,5431,3411,2344]
b_2 = [3456,2123,1455,8764,2323]
b_3 = [213,431,124,56,120]

bar_width=0.2

x_1 = list(range(len(a)))
x_2 = [i+bar_width for i in x_1]
x_3 = [i+bar_width*2 for i in x_1]

pyplot.figure(figsize=(10,4),dpi=80)

pyplot.bar(range(len(a)),b_1,width=bar_width,color="blue",label="1")
pyplot.bar(x_2,b_2,width=bar_width,color="pink",label="2")
pyplot.bar(x_3,b_3,width=bar_width,label="3")

pyplot.xticks(x_2,a)

pyplot.legend()
pyplot.show()
12345678910111213141516171819202122324
```

4.绘制直方图（pyplot.hist()）

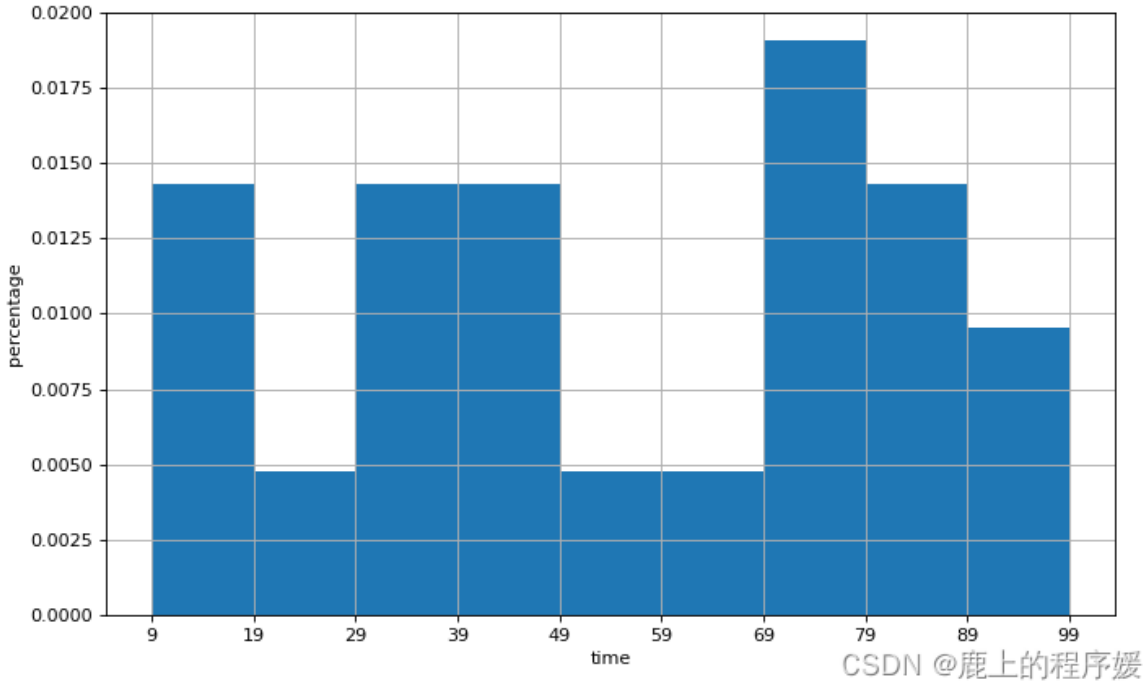
频数分布直方图pyplot.hist(a,num_bins)




```python
from matplotlib import pyplot

a = [9,34,13,73,44,34,76,34,72,17,96,46,84,52,72,26,81,64,79,45,99]

```

```
d = 10
num_bins = (max(a)-min(a))//d

pyplot.figure(figsize=(10,6),dpi=80)
pyplot.hist(a,num_bins)
pyplot.xlabel("time")
pyplot.ylabel("num")
pyplot.xticks(range(min(a),max(a)+d,d))
pyplot.grid()
pyplot.show()
1234567891011121314
```

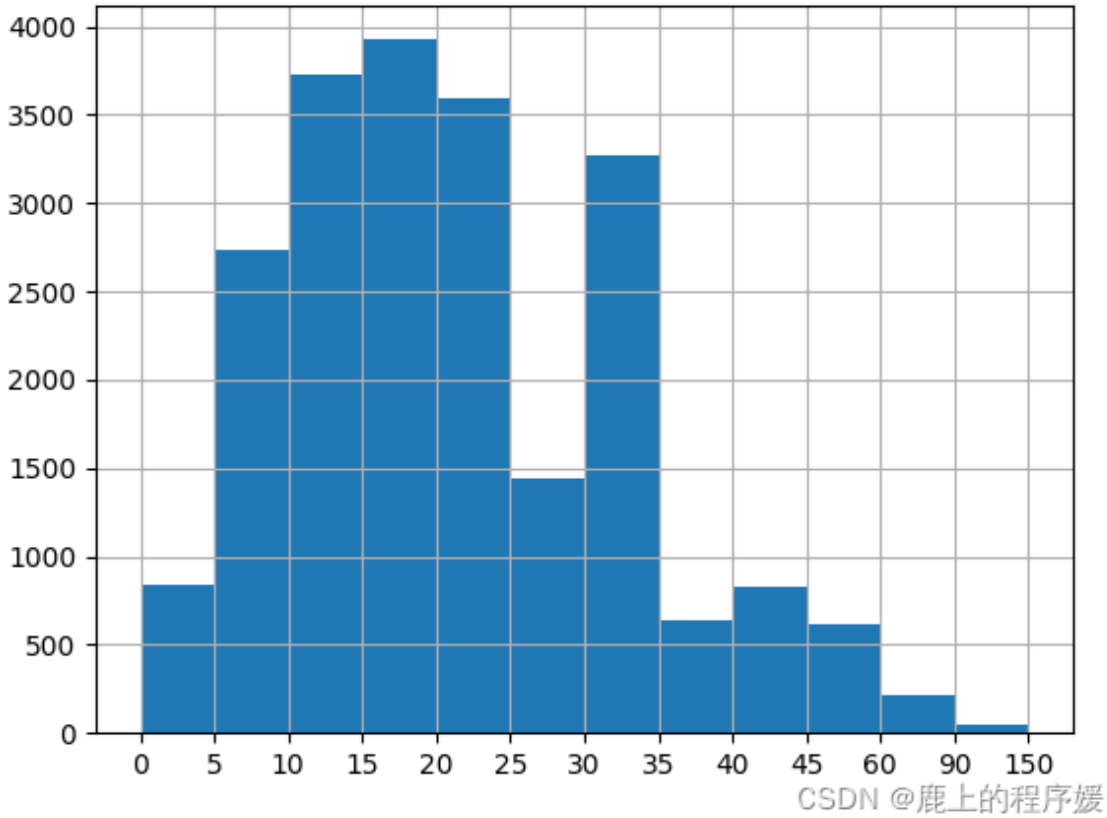
频率分布直方图pyplot.hist(a,num_bins,density=True)




```
from matplotlib import pyplot

a = [9,34,13,73,44,34,76,34,72,17,96,46,84,52,72,26,81,64,79,45,99]

d = 10
num_bins = (max(a)-min(a))//d

pyplot.figure(figsize=(10,6),dpi=80)
pyplot.hist(a,num_bins,density=True)
pyplot.xlabel("time")
pyplot.ylabel("percentage")
pyplot.xticks(range(min(a),max(a)+d,d))
pyplot.grid()
pyplot.show()
1234567891011121314
```

绘制组距变化的直方图(pyplot.bar())

这里的组距为一个数组




```
from matplotlib import pyplot

interval = [0,5,10,15,20,25,30,35,40,45,60,90]
width = [5,5,5,5,5,5,5,5,5,15,30,60]
quality = [836,2737,3723,3926,3596,1438,3273,642,824,613,215,47]

pyplot.bar(range(12),quality,width=1)
_x = [i-0.5 for i in range(13)]
_xtick_label = interval+[150]
pyplot.xticks(_x,_xtick_label)
pyplot.grid()
pyplot.show()
123456789101112
```

三角函数

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

#让图标能显示汉字
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['axes.unicode_minus']=False

#设置画布，10,10等大小表示正方形
plot1 = plt.figure(figsize=(10,10),dpi=80)

#圆
```

```
plot1.add_subplot(2,2,1)
Q = np.arange(0,np.pi*2,0.01)
x = 2*np.cos(Q)
y = 2*np.sin(Q)
plt.plot(x,y,color='pink',marker="*")
plt.legend(["圆"])

#sin
plot1.add_subplot(2,2,2)
x = np.arange(0,np.pi*2,0.1)
y = np.sin(x)
plt.plot(x,y)
plt.legend(["sin"])

#cos
plot1.add_subplot(2,2,3)
x = np.linspace(-np.pi,np.pi,1000) # 线性拆分1000个点
y = np.cos(x)
plt.plot(x,y)
plt.legend(["cos"])

# #tan
plot1.add_subplot(2,2,4)
x = np.arange(0,np.pi*2,0.1)
y = np.tan(x)
plt.ylim(-10,10)
plt.plot(x,y)
plt.legend(["tan"])

plt.show()
```

总结

1.如何选择哪种图来呈现数据?

2.matplotlib.plot(x,y)

绘制的是折线图，x为代表x轴的list，y为代表y轴的值的list，这里的x和y的元素个数必须是一致的

```
def plot(*args, scalex=True, scaley=True, data=None, **kwargs):
    return gca().plot(
        *args, scalex=scalex, scaley=scaley,
        **({"data": data} if data is not None else {}), **kwargs)
```

3.matplotlib.bar(x,height,width)

绘制的是条形图，x为代表x轴的list，height为对应的x的值，width为条形图的宽度

```
def bar(
        x, height, width=0.8, bottom=None, *, align='center',
        data=None, **kwargs):
```

4.matplotlib.barh(y,width,height)

绘制横着的条形图，x和y的含义相反

```
def barh(y, width, height=0.8, left=None, *, align='center', **kwargs):
    return gca().barh(
        y, width, height=height, left=left, align=align, **kwargs)
```

5.matplotlib.scatter(x,y)

绘制散点图

```
def scatter(
        x, y, s=None, c=None, marker=None, cmap=None, norm=None,
        vmin=None, vmax=None, alpha=None, linewidths=None, *,
        edgecolors=None, plotnonfinite=False, data=None, **kwargs):
```

6.matplotlib.hist(x,bin,density)

绘制直方图，这里的x为源数据的数组，bin为分多少组显示，这里的图的y值代表在某个范围内的频率或频数，通过参数density可以绘制频数直方图或频率直方图，默认为频数直方图

一般设置一个组距d

bin = (max(a)-min(x))//d

```
def hist(
        x, bins=None, range=None, density=False, weights=None,
        cumulative=False, bottom=None, histtype='bar', align='mid',
        orientation='vertical', rwidth=None, log=False, color=None,
        label=None, stacked=False, *, data=None, **kwargs):
```

7.xticks和yticks的设置

设置x轴和y轴的坐标

8.label和title，grid的设置

9.绘图的大小(figure)和保存图片(savefig)

运算符优先级

很多情况下，一个表达式由多个运算符组成，优先级决定运算符的计算序。

运算符	描述
**	指数 (最高优先级)
~ + -	按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@)
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符
&	位 'AND'
^ |	位运算符
<= < > >=	比较运算符
<> == !=	等于运算符
= %= /= //= -= += *= **=	赋值运算符
is is not	身份运算符
in not in	成员运算符
not or and	逻辑运算符

实例方法、静态方法和类方法

方法包括：实例方法、静态方法和类方法，三种方法在内存中都归属于类，区别在于调用方式不同。

实例方法：由对象调用；至少一个self参数；执行实例方法时，自动将调用该方法的对象赋值给self；类方法：由类调用； 至少一个cls参数；执行类方法时，自动将调用该方法的类赋值给cls；静态方法：由类调用；无默认参数；

生成器

1. 生成器定义

在Python中，一边循环一边计算的机制，称为生成器：generator。

2. 为什么要有生成器

列表所有数据都在内存中，如果有海量数据的话将会非常耗内存。

如：仅仅需要访问前面几个元素，那后面绝大多数元素占用的空间都白白浪费了。

如果列表元素按照某种算法推算出来，那我们就可以在循环的过程中不断推算出后续的元素，这样就**不必创建完整的list，从而节省大量的空间。**

简单一句话：我又想要得到庞大的数据，又想让它占用空间少，那就用生成器！

3.如何创建生成器

第一种方法很简单，只要**把一个列表生成式的[]改成()**，就创建了一个generator：

```
>>> L = [x * xforxinrange(10)]
>>> L
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> g = (x * xforxinrange(10))
>>> g
<generator object <genexpr>  at 0x1022ef630>
```

创建L和g的区别仅在于最外层的[]和()，L是一个list，而g是一个generator。

方法二， 如果一个**函数中包含yield关键字**，那么这个函数就不再是一个普通函数，而是一个generator。调用函数就是创建了一个生成器（generator）对象。

4. 生成器的工作原理

（1）生成器(generator)能够迭代的关键是它有一个next()方法，工作原理就是**通过重复调用next()方法，直到捕获一个异**常。

（2）带有 yield 的函数不再是一个普通函数，而是一个生成器generator。

可用next()调用生成器对象来取值。next 两种方式 t.**next**() | next(t)。

可用for 循环获取返回值（每执行一次，取生成器里面一个值）

（基本上不会用next()来获取下一个返回值，而是直接使用for循环来迭代）。

（3）yield相当于 return 返回一个值，并且记住这个返回的位置，**下次迭代时，代码从yield的下一条语句开始执行。**

（4）.send() 和next()一样，都能让生成器继续往下走一步（下次遇到yield停），但send()能传一个值，这个值作为yield表达式整体的结果

——换句话说，就是send可以强行修改上一个yield表达式值。比如函数中有一个yield赋值，a = yield 5，第一次迭代到这里会返回5，a还没有赋值。第二次迭代时，使用.send(10)，那么，就是强行修改yield 5表达式的值为10，本来是5的，那么a=10

感受下yield返回值的过程（*关**注点：每次停在哪，下次又开始在哪***）及send()传参的通讯过程，

思考None是如何产生的（第一次取值：yield 返回了 i 值 0，停在yield i，temp没赋到值。第二次取值，开始在print，temp没被赋值，故打印None，i加1，继续while判断，yield 返回了 i 值 1，停在yield i）：

```
#encoding:UTF-8
def yield_test(n):
    for i in range(n):
        yield call(i)
        print("i=",i)
    print("Done.")
def call(i):
    return i*2
for i in yield_test(5):
    print(i,",")
```

结果：

```
>>>
0 ,
i= 0
2 ,
i= 1
4 ,
i= 2
6 ,
i= 3
8 ,
i= 4
Done.
>>>
```

理解的关键在于：下次迭代时，代码从yield的下一条语句开始执行。

5. 总结:

什么是生成器?

生成器仅仅保存了一套生成数值的算法，并且没有让这个算法现在就开始执行，而是我什么时候调它，它什么时候开始计算一个新的值，并给你返回。

练习题:

```
def count_down(n):
    whilen >= 0:
        newn =yield n
        print('newn', newn)
        if newn:
            print('if')
            n = newn
            print('n =', n)
        else:
            n -= 1
cd = count_down(5)
for i in cd:
    print(i,',')
    ifi == 5:
        cd.send(3)
5 ,
newn 3
if
n = 3
newn None
2 ,
newn None
1 ,
newn None
0 ,
newn None
```