

作业一 Python 练习

1、编写一个程序，要求用户输入一个数字，输出这个数字的所有除数。

基本思路：遍历 1 到这个数字，如果能被整除，则证明是除数，加入数组，最终返回，但是开销太大，遂采用遍历到对该数开根号处，另一半由算出来的除数与这个数相除得到，最终排序即可，细节上，需要注意到可以开平方的数，这种数字需要遍历到开方处（包含），且不应该在这里把数字除以除数加入到结果，因为它们会产生重复。

源代码：

```
class DivisorCalculator:
    def __init__(self, num: int):
        self.num = num
        self.upper = int(num * 0.5) if num > 1 else 1

    def calculate_division_num(self) -> List:
        res = []
        for i in range(1, self.upper):
            if self.num % i == 0:
                if i > self.num ** 0.5:
                    break
                res.append(i)
                another = int(self.num / i)
                if another != i:
                    res.append(another)
        res.sort()
        return res

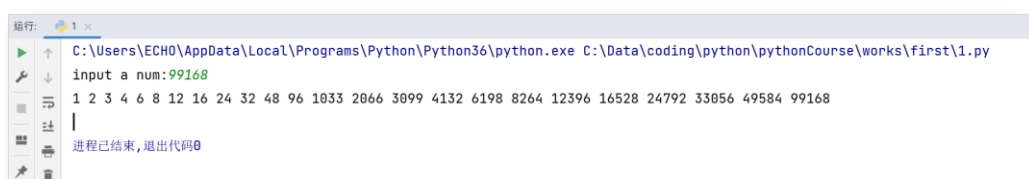
if __name__ == '__main__':
    print(' '.join(str(i) for i in DivisorCalculator(int(input('input a num:'))).calculate_division_num()))
```

测试用例：99168

运行结果：

input a num:99168

1 2 3 4 6 8 12 16 24 32 48 96 1033 2066 3099 4132 6198 8264 12396 16528 24792 33056 49584 99168



```
运行: C:\Users\ECHO\AppData\Local\Programs\Python\Python36\python.exe C:\Data\coding\python\pythonCourse\works\first\1.py
input a num:99168
1 2 3 4 6 8 12 16 24 32 48 96 1033 2066 3099 4132 6198 8264 12396 16528 24792 33056 49584 99168
|
进程已结束,退出代码0
```

2、编写一个石头剪刀布游戏，系统随机生成石头、剪刀、布的一种，要求玩家输入（使用 input 函数）其中的一种，比较并输出游戏结果，并询问是否继续游戏。

基本思路：进入一个死循环，每次循环生成一个随机的选项作为系统猜拳，读取用户输入，比较并输出，询问是否继续游戏，并由此判断是否跳出这个死循环。

源代码：

```
import random as rd

class GuessTheBox:
    """
    define: 0 → ✂  1 → stone  2 → cloth
    """
    sys_guess = -1

    def sys_re_guess(self):
        self.sys_guess = rd.choice(range(3))

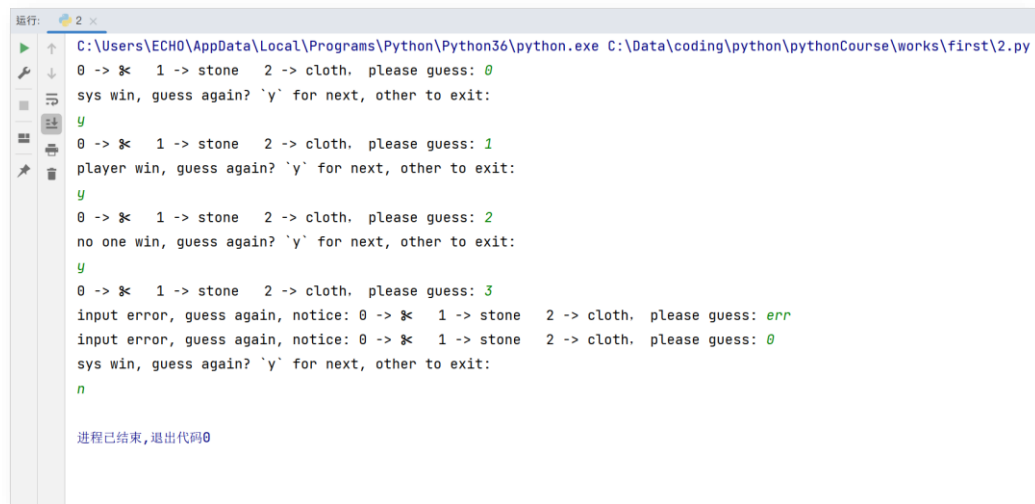
    def judge(self, player_guess):
        sys, player, none = 'sys', 'player', 'no one'
        table = {
            0: {
                0: none, 1: player, 2: sys
            },
            1: {
                0: sys, 1: none, 2: player
            },
            2: {
                0: player, 1: sys, 2: none
            }
        }
        return table[self.sys_guess][player_guess]

    def begin(self):
        while True:
            self.sys_re_guess()
            while True:
                try:
                    player_guess = input('0 → ✂  1 → stone  2 → cloth, please guess: ')
                    player_guess = int(player_guess)
                    if player_guess not in range(3):
                        raise ValueError()
                except Exception:
                    print('input error, guess again, notice: ', end='')
                else:
                    break
            print(f'{self.judge(player_guess)} win, guess again? `y` for next, other to exit:')
            if input() not in ('y', 'Y'):
                break

if __name__ == '__main__':
    GuessTheBox().begin()
```

测试用例：0 y 1 y 2 y 3 err 0 n

运行结果：



```
运行: 2 x
C:\Users\ECHO\AppData\Local\Programs\Python\Python36\python.exe C:\Data\coding\python\pythonCourse\works\first\2.py
0 -> %< 1 -> stone 2 -> cloth. please guess: 0
sys win, guess again? 'y' for next, other to exit:
y
0 -> %< 1 -> stone 2 -> cloth. please guess: 1
player win, guess again? 'y' for next, other to exit:
y
0 -> %< 1 -> stone 2 -> cloth. please guess: 2
no one win, guess again? 'y' for next, other to exit:
y
0 -> %< 1 -> stone 2 -> cloth. please guess: 3
input error, guess again, notice: 0 -> %< 1 -> stone 2 -> cloth. please guess: err
input error, guess again, notice: 0 -> %< 1 -> stone 2 -> cloth. please guess: 0
sys win, guess again? 'y' for next, other to exit:
n
进程已结束,退出代码0
```

3、随机生成 1-9 之间的数字，要求用户猜这个数字，当用户输入错误时，提示比该数字大还是小，当用户输入 exit 时退出游戏，当用户猜中时提示成功，并显示猜了几次，询问是否继续游戏。

基本思路：开启一个死循环，每次循环随机生成一个数字，同时定义一个计数器，用户猜测的时候+1，再进入一个死循环，不断地获取用户的输入，根据用户输入提示成功与否或者结束游戏，猜到后根据用户输入决定是否继续游戏。

源代码：



```
import random

if __name__ == '__main__':
    while True:
        num = random.randint(1, 10)
        count = 0
        print('input your guess(between 1 and 9): ')
        while True:
            count += 1
            try:
                s = input()
                if s.lower() == 'exit':
                    exit(0)
                val = int(s)
                if val not in range(1, 10):
                    raise ValueError
            except Exception:
                count -= 1
                print('input error, please try again:')
            else:
                if val < num:
                    print('Too small, try again please:')
                elif val > num:
```

```

        print('Too big, try again please:')
    else:
        break
    print(f'Bingo, you have tried {count} times! Would u like to play again(`y` for next and other to end):')
    if input() not in ('y', 'Y'):
        break

```

测试用例：3 2 y 1 ip 6 y exit

运行结果：

```

运行: 3 x
C:\Users\ECHO\AppData\Local\Programs\Python\Python36\python.exe C:\Data\coding\python\pythonCourse\works\first\3.py
input your guess(between 1 and 9):
3
Too big, try again please:
2
Bingo, you have tried 2 times! Would u like to play again(`y` for next and other to end):
y
input your guess(between 1 and 9):
1
Too small, try again please:
ip
input error, please try again:
6
Bingo, you have tried 2 times! Would u like to play again(`y` for next and other to end):
y
input your guess(between 1 and 9):
exit
进程已结束,退出代码0

```

4、使用函数编写一段程序，要求用户输入一句英文，将其中的单词反向排列并输出。

基本思路：先获取输入，再按照空格切片，然后反向排列，再使用 join 方法插入空格，重新连接为一个句子。

源代码：

```

print(' '.join(i for i in input('input an english sentence, split with space:').split(' ')[::-1]))

```

测试用例：we all have a deep love for chinese ancient poetry

运行结果：

```

运行: 4 x
C:\Users\ECHO\AppData\Local\Programs\Python\Python36\python.exe C:\Data\coding\python\pythonCourse\works\first\4.py
input an english sentence, split with space:we all have a deep love for chinese ancient poetry
poetry ancient chinese for love deep a have all we
进程已结束,退出代码0

```

5、编写一个 Cows and bulls 游戏。

基本思路：运行时先生成一个开头不为零、各位置不重复的四位数，玩家每一次提交时检查其输入并给出判断，其中，需要注意处理异常情况，如输入的数字不是 4 位或者非数字（提示错误输入要求重试）、输入了重复数字或者开头为 0 的数字（提示违规要求重试）。

源代码：

```
import random

class GuessNumber:
    def __init__(self):
        self.number = None

    def generator(self):
        nums = [-1 for _ in range(4)]
        nums[0] = random.randrange(1, 10)
        for i in range(1, 4):
            while True:
                n = random.randrange(0, 10)
                if n not in nums:
                    nums[i] = n
                    break
            self.number = ''.join(map(str, nums))

    def check(self, user_ans) -> bool:
        """
        check player answer
        :param user_ans: user answer, len(str) = 4
        :return: guess right or not
        """
        try:
            if (user_ans[0] == '0' and int(user_ans) not in range(1000)) or int(user_ans) not in range(10000):
                raise ValueError()
        except Exception:
            print('bad input, try again: ')
            return False

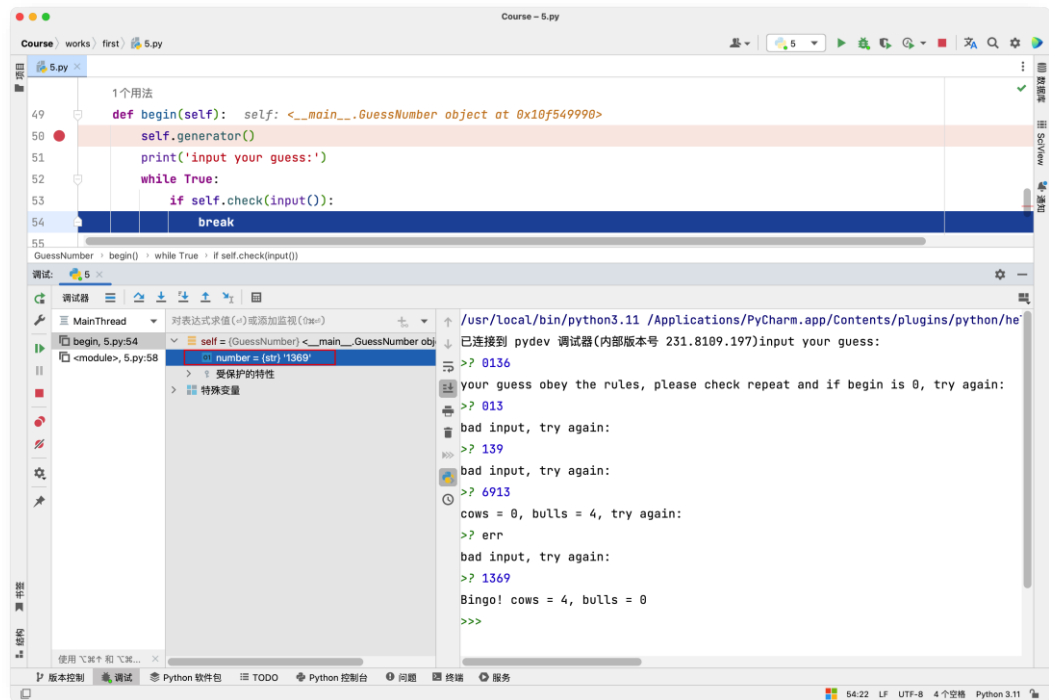
        if user_ans[0] == '0' or user_ans.count(user_ans[0]) != 1 or user_ans.count(user_ans[1]) != 1 or \
            user_ans.count(user_ans[2]) != 1 or user_ans.count(user_ans[3]) != 1:
            print('your guess obey the rules, please check repeat and if begin is 0, try again: ')
        else:
            cows = 0
            bulls = 0
            for i in range(4):
                if self.number[i] == user_ans[i]:
                    cows += 1
            bulls += self.number.count(user_ans[i])
            bulls += -1 * cows
            if cows == 4:
                print('Bingo! cows = 4, bulls = 0')
                return True
            else:
                print(f'cows = {cows}, bulls = {bulls}, try again: ')
        return False

    def begin(self):
        self.generator()
        print('input your guess:')
        while True:
            if self.check(input()):
                break

if __name__ == '__main__':
    GuessNumber().begin()
```

测试用例：0136 013 136 6913 err 1369

运行结果：



作业三 Matplotlib 绘图

从 matplotlib 官网 gallery 选择三幅图进行绘制。

(1)、折线图

参照示例 https://matplotlib.org/stable/gallery/lines_bars_and_markers/multicolored_line.html#sphx-glr-gallery-lines-bars-and-markers-multicolored-line-py, 该图形为绘制一个函数图像, 而且使用了函数的导数值作为线条颜色, 显得美观、好看, 也能直观的体现出函数的变化趋势, 如下:

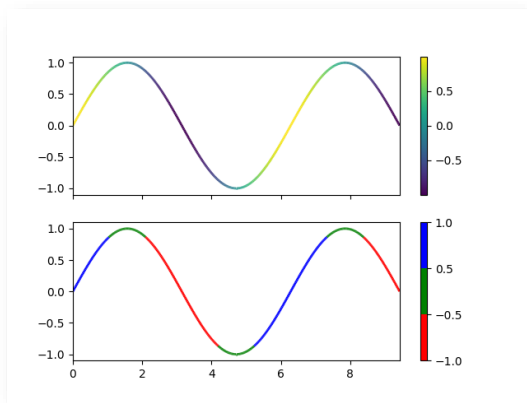


图 1 折线图示例

现仿照此图, 绘制一个正弦函数在两个周期内的图像, 其线条颜色对应该处的导数值, 代码如下:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.collections import LineCollection as Lc
from matplotlib.colors import ListedColormap as Lcp

class ColoringOfSinusoidalDerivatives:
    colors = ['#93AF9F', '#99B5A3', '#9FB8A7', '#A5C1AB', '#ABC7AF', '#B1CDB3', '#B7D3B7', '#BAD6B8',
              '#BFD8BA', '#C4DABC', '#C9DCBE', '#CEDECB', '#D3E0C2', '#D8E2C4', '#DBE4C6', '#E1E5C3', '#E7E6C0', '#EDE7BD',
              '#F3E8BA', '#F9E9B7', '#FFEAB4', '#FFE8B4', '#FFE4B3', '#FFDDB2', '#FFD6B1', '#FFCFB0', '#FFC8AF', '#FFC1AE',
              '#FFBFA9', '#FFB8A9', '#FFB6A9', '#FFB4A9', '#FFB3A9', '#FFB0A9', '#FFADA9', '#FFACAC', '#FFA0AA', '#FF94A8',
              '#FF88A6', '#FF7CA4', '#FF70A2', '#FF64A0', '#FF609E', '#F85D97', '#F15A90', '#EA5789', '#E35482', '#DC517B',
              '#D54E74']

    def __init__(self, num):
        self.num = num

    def draw(self):
        x = np.linspace(0, self.num * np.pi, 500)
        derivative = np.cos(0.5 * (x[:-1] + x[1:]))

        points = np.array([x, np.sin(x)]).T.reshape(-1, 1, 2)
        segments = np.concatenate([points[:-1], points[1:]], axis=1)
        fig = plt.figure()
        ax = plt.subplot(111)
```

```

cmap = Lcp(self.colors)
norm = plt.Normalize(derivative.min(), derivative.max())
lc = Lc(segments, cmap=cmap, norm=norm)
lc.set_array(derivative)
lc.set_linewidth(2)
line = ax.add_collection(lc)
fig.colorbar(line, ax=ax)
ax.set_xlim(x.min(), x.max())
ax.set_ylim(-1.1, 1.1)
plt.title('sine function')
plt.show()

if __name__ == '__main__':
    ColoringOfSinusoidalDerivatives(4).draw()

```

首先，定义了一个类 `ColoringOfSinusoidalDerivatives`，它有一个构造方法，接受一个参数 `num`，表示正弦函数的周期数，然后，定义了一个方法 `draw`，它没有参数，用于绘制正弦函数和导数。

在 `draw` 方法中，首先生成一个指定周期内的数组 `x`，长度均分为 500，这个数组表示正弦函数的自变量，然后，计算 `x` 中相邻两点之间的中点处的余弦值即导数值，并赋值给 `derivative`。接下来，将 `x` 和正弦函数的值组合成一个二维数组，并使用 `reshape` 方法将其变形为一个三维数组 `points`，这个数组表示正弦函数上的点坐标。然后，将各点沿着第二个轴拼接起来，得到相邻点之间的线段数组 `segments`。最终设置线条相关属性，在图形对象添加一个颜色条，设置图形的轴标签、范围以及图形标题，进行绘图。

在主程序中，创建了一个 `ColoringOfSinusoidalDerivatives` 类的实例，并传入 4 作为周期数，最后调用了 `draw` 方法绘制正弦函数图像。

绘图结果如下：

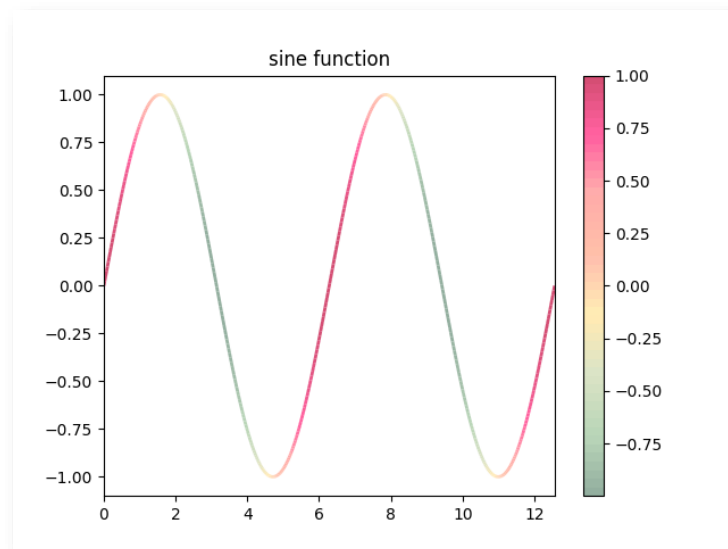


图 2 正弦函数导数渐变绘图结果

在绘制图形时，使用程序生成了从绿色到黄色到红色的一系列过渡色，将其作为颜色条和图形的变化颜色，可以看到，当正弦函数的图像处于“走下坡”时，图象为绿色，反之则为红色，到中间时颜色为黄色，符合正弦函数的规律。

(2)、条形图

参照示例 https://matplotlib.org/stable/gallery/lines_bars_and_markers/barchart.html

#sphx-glr-gallery-lines-bars-and-markers-barchart-py，该图形为分组条形图，其中包含不同对象的不同属性，可以方便的进行各数据的对比，且不同的属性具有不同的颜色，显得一目了然，易于观察，如下图：

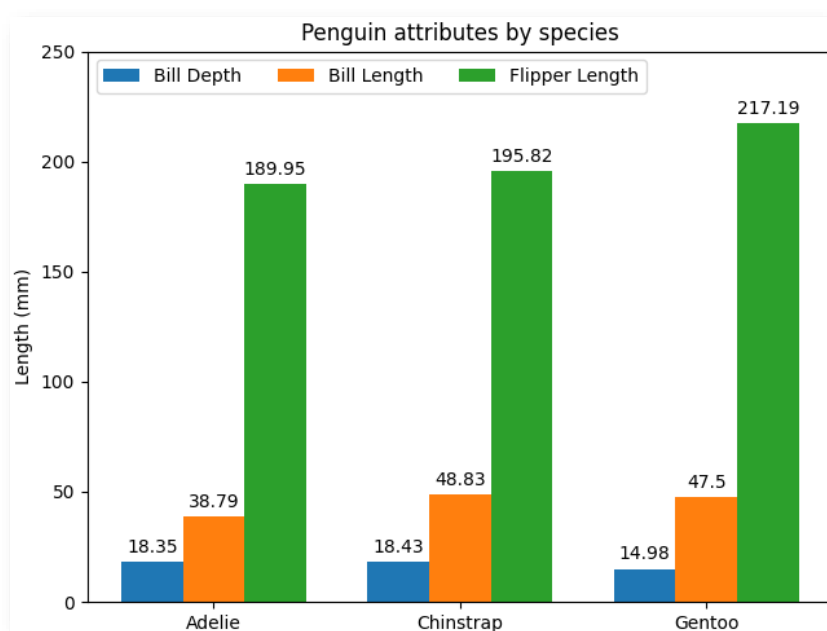


图 3 分组条形图示例

现仿照示例，绘制校园运动会各学院奖牌的条形图，使用不同的颜色标注金牌、银牌和铜牌(数据来源: <https://yiban.minmin.cloud/yundonghui/index/index>)，代码如下：

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

plt.rcParams['font.family'] = ['SimHei']

def draw_rank():
```

```

labels = ('轻工', '材料', '环境', '电智', '食品', '机电', '镐京', '经管')
x = np.arange(len(labels))
au = [3, 7, 0, 2, 0, 4, 14, 0]
ag = [1, 7, 1, 1, 3, 1, 10, 3]
cu = [1, 1, 1, 3, 1, 5, 9, 2]
wid = 0.23
p = plt.subplot(111)
r1 = p.bar(x - wid, au, wid, label='金牌', color=['#FFD95A'])
r2 = p.bar(x, ag, wid, label='银牌', color=['#DBDFEA'])
r3 = p.bar(x + wid, cu, wid, label='铜牌', color=['#F7B183'])
bars = [r1, r2, r3]
for rs in bars:
    for bar in rs:
        height = bar.get_height()
        x_pos = bar.get_x() + wid/2
        plt.text(x_pos, height, height, ha="center", va="bottom")
p.set_ylabel('奖牌数量')
p.set_title('校运会各学院奖牌榜（部分）')
p.set_xticklabels(labels, )
p.set_xticks(x)
plt.xticks(rotation=70)
p.legend()

plt.show()

if __name__ == '__main__':
    draw_rank()

```

在最开始导入 matplotlib 库之后，为了防止显示中文乱码，故使用 plt.rcParams 设置了字体为 SimHei。

在函数中，首先定义了一个元组 labels，表示各学院的名称。然后使用 numpy.arange 函数生成一个从 0 到 labels 长度的整数数组 x，表示各学院的位置。接下来定义了三个列表 au、ag 和 cu，分别表示各学院的金牌、银牌和铜牌的数量。然后定义柱子的宽度，接下来，使用 plt 的 bar 函数分别绘制三组柱子，并且计算了相应的柱子的宽高和坐标，同时还设置了其标签和颜色，然后，使用一个 for 循环遍历三组柱子，并在每个柱子上方显示其高度（即奖牌数量）。

最后，设置了 y 轴的标签为“奖牌数量”、标题为“校运会各学院奖牌榜（部分）”以及 x 轴的刻度标签为 labels 中的元素，并且还将 x 轴的刻度标签旋转 70 度，以便显示清楚。最后，添加了必要的标注和相关信息，并使用 p.legend 方法添加图例。

绘图结果如下：

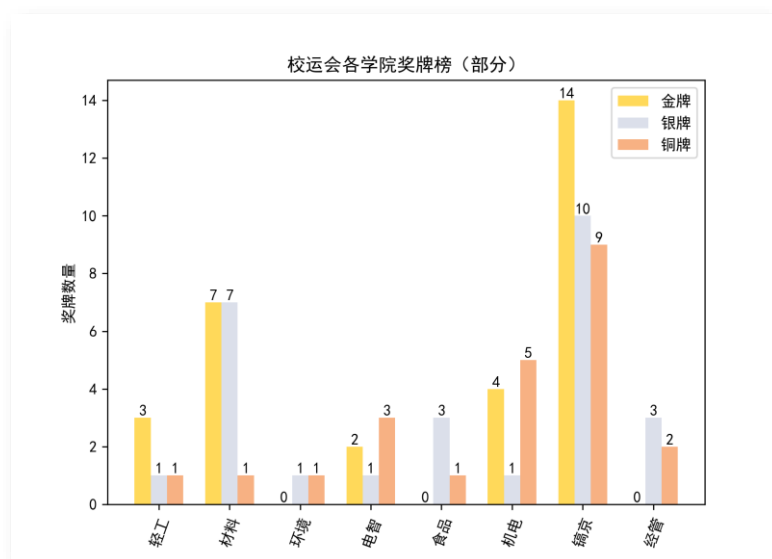


图 4 分组条形图绘图结果

如图，可以较为清晰的看到，绘制了各个学院的奖牌情况，金牌、银牌、铜牌使用不同的颜色进行标注，相应的奖牌的数量也有标注。

(3)、散点图

参照示例 https://matplotlib.org/stable/gallery/lines_bars_and_markers/scatter_demo2.html#sphx-glr-gallery-lines-bars-and-markers-scatter-demo2-py，这个代码绘制一个散点图，展示股票数据的相关性，是探索相邻两天的股价变化率之间是否有某种规律，以及当天的收盘价、开盘价和成交量是否对股价变化有影响，这个散点图可以用来发现一些数据的特征或者趋势：

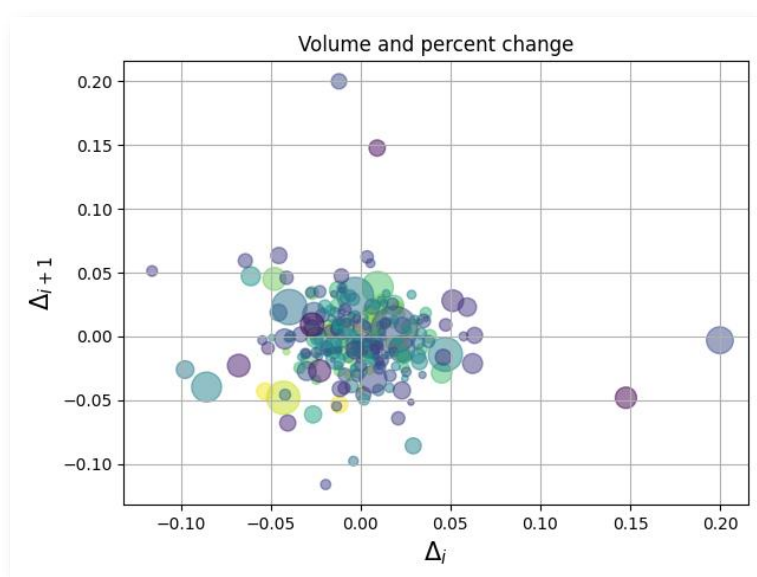


图 5 散点图示例

实现画图源码:

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

def draw_points(n=200):
    dates = np.arange(n)
    opens = np.random.uniform(1000, 2000, size=n)
    highs = opens + np.random.uniform(0, 100, size=n)
    lows = opens - np.random.uniform(0, 100, size=n)
    closes = lows + np.random.uniform(0, 200, size=n)
    volumes = np.random.randint(1000000, 5000000, size=n)
    adj_closes = closes * np.random.uniform(0.9, 1.1, size=n)
    price_data = np.rec.fromarrays([dates, opens, highs, lows, closes, volumes, adj_closes, ],
                                   names=['date', 'open', 'high', 'low', 'close', 'volume',
                                           'adj_close'])
    delta1 = np.diff(price_data.adj_close) / price_data.adj_close[:-1]
    volume = (15 * price_data.volume[:-2] / price_data.volume[0]) ** 2
    close = 0.003 * price_data.close[:-2] / 0.003 * price_data.open[:-2]
    fig, ax = plt.subplots()
    ax.scatter(delta1[:-1], delta1[1:], c=close, s=volume, alpha=0.5)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_title('Random points')
    ax.grid(True)
    fig.tight_layout()
    plt.show()

if __name__ == '__main__':
    draw_points()
```

由于缺少相关的真实数据, 为了能够画出图来, 使用 numpy 的 random 模块随机生成相关数据进行绘制, 最终结果如下:

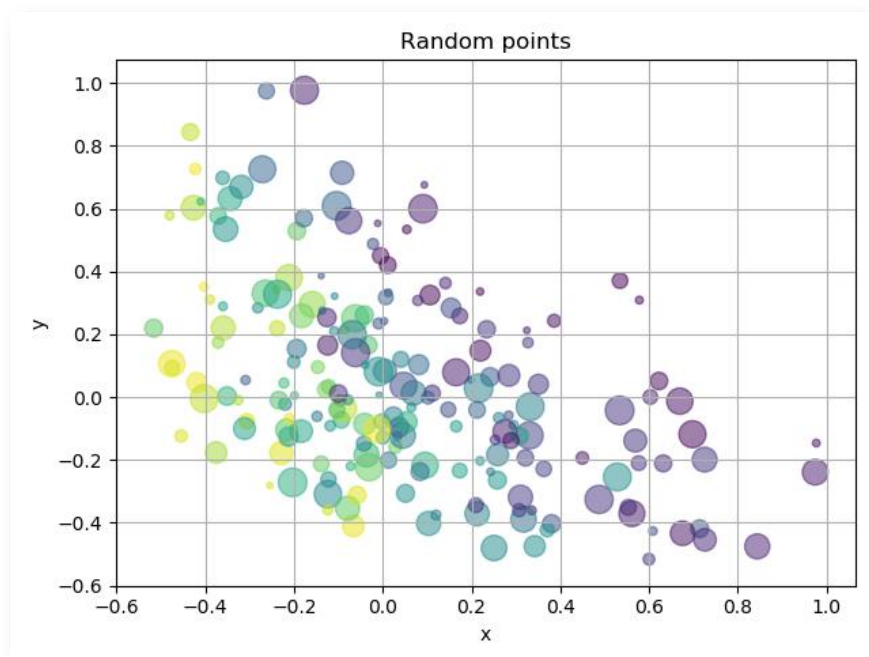


图 6 彩色随机散点图