《Python程序设计基础》考查

程序设计作品说明书

|  |  |
| --- | --- |
| 题目 | 数据可视化 |
| 学院 | 计算机工程与应用数学学院 |
| 专业(班级) | 19软件03班 |
| 姓名 | 康泽 |
| 学号 | B20190304316 |
| 指导教师 | 周景 |
| 起止日期 | 2022.4.25—2022.6.4 |

**互评分数表**

|  |  |
| --- | --- |
| 答题人信息 | |
| 答题人姓名 | 康泽 |
| 答题人班级 | 19软件03班 |
| 答题人学号 | B20190304316 |
| 互评分数 | |
| 总分 | 158 |
| 评语 | 完成了所有功能的实现、python语法符合规范，文档编写完整，使用了git版本控制，编写了单元测试。 |
| 互评人信息 | |
| 互评人姓名 | 龙志强 |
| 互评人学号 | B20190304321 |
| 互评人班级 | 19软件03班 |

## 第1章 需求分析

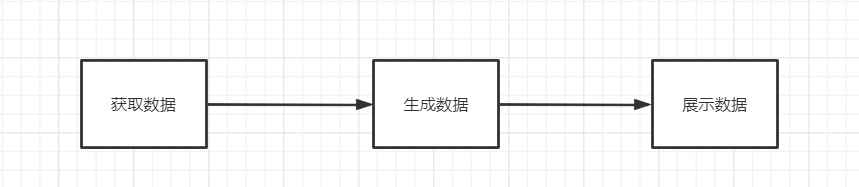
本系统需要实现了对数据的可视化操作，包括制作折线图、散点图以及直方图等等，需要通过用户所提供的数据可以制作相应的可视化图形，从而解决用户的需求，主要的需求就是能够绘制模拟掷骰子、天气数据、全球地震散点图以及使用Web API获取数据并绘制相应的图形。

# 第2章 设计与实现

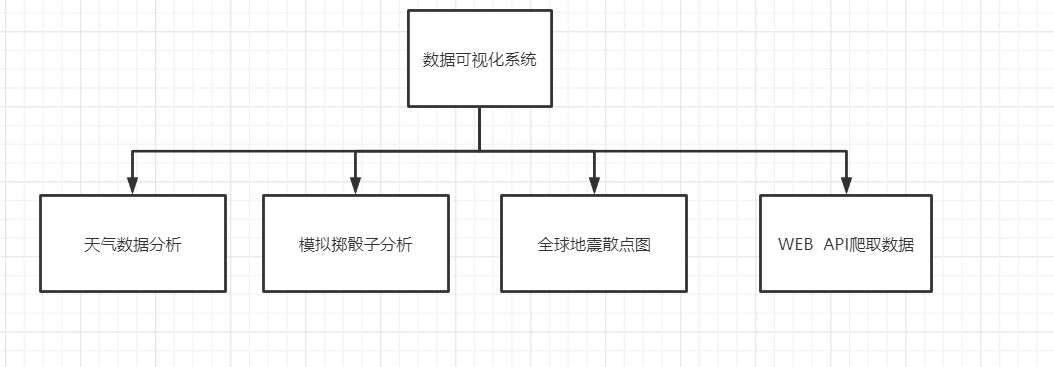
**系统架构：**

使用python的工具包Matplotlib和plotly来生成数据，使用python的工具包requests来使用Web API接口来爬取数据。

**系统流程：**



**系统模块：**



# 第3章 单元测试

* 对随机生成骰子的测试：

import unittest

from die import Die

class DieTestCase(unittest.TestCase):

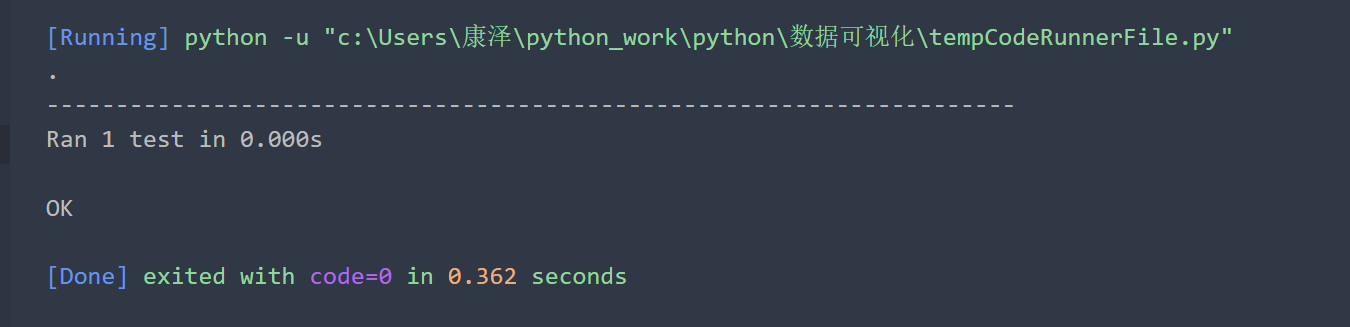
    def test\_num\_side(*self*):

        die = Die(10)

        self.assertEqual(10,die.num\_sides)

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()



* 对随机漫步的测试：

import unittest

from random\_walk import RandomWalk

class randomWalkTestCase(unittest.TestCase):

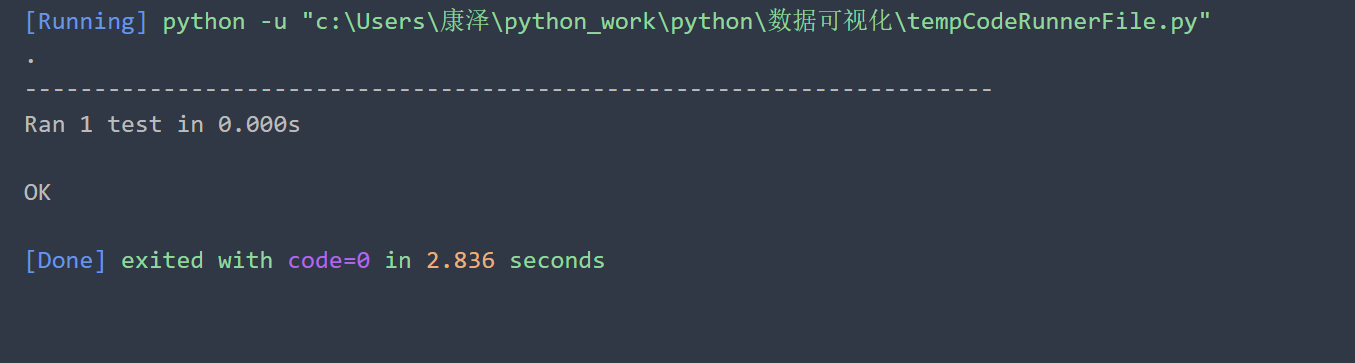
    def test\_num\_points(*self*):

       randomwalk = RandomWalk()

       self.assertEqual(5000,randomwalk.num\_points)

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()



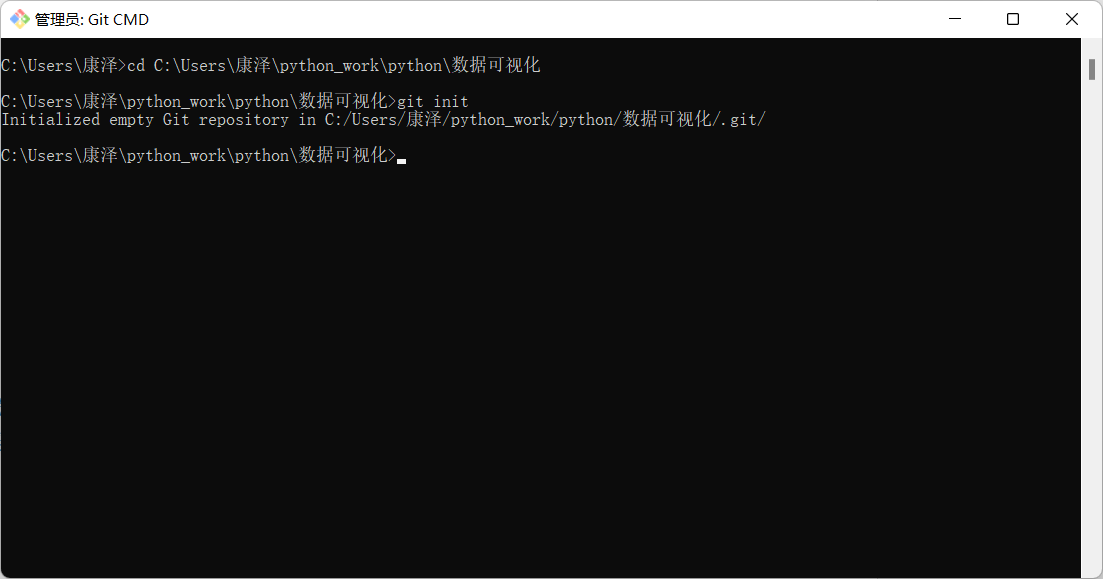
## 第4章 Git版本控制管理

1. 安装git
2. 配置git

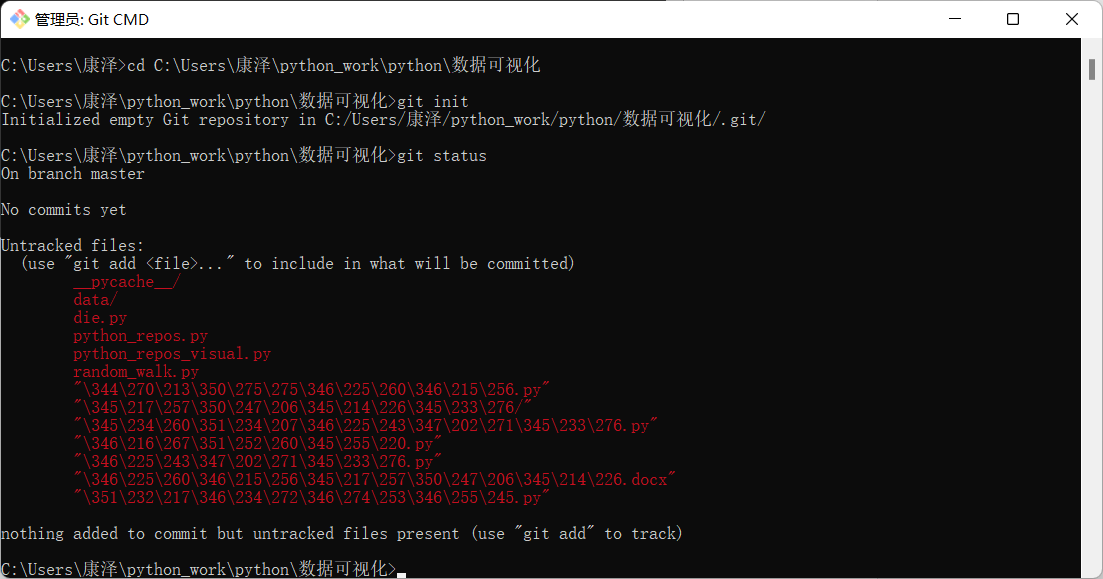
git config -global user.name "Andy"

git config -global user.email "2286094487@qq.com"

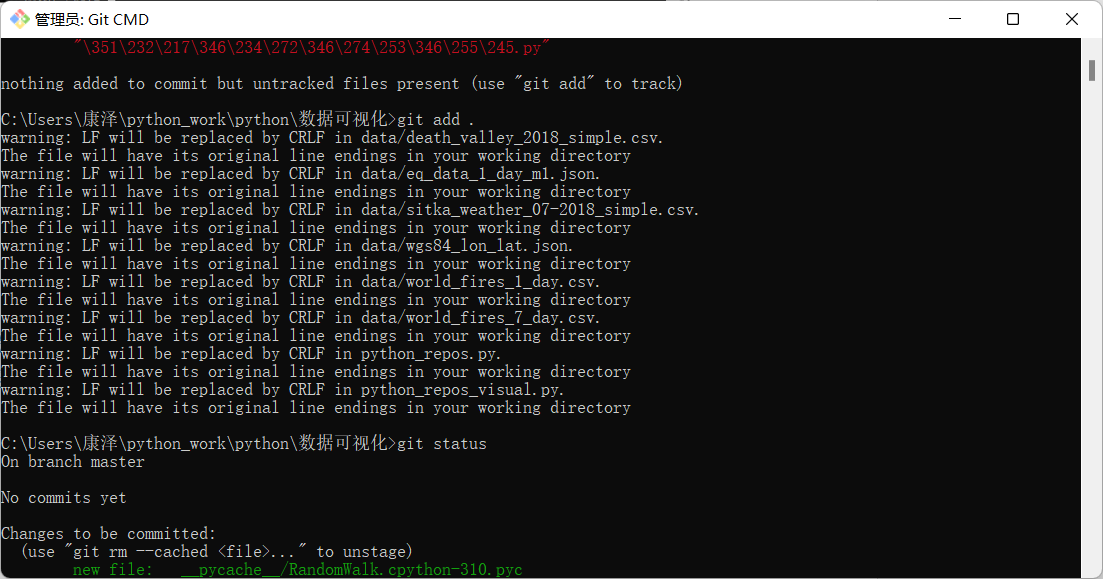
1. 初始化仓库



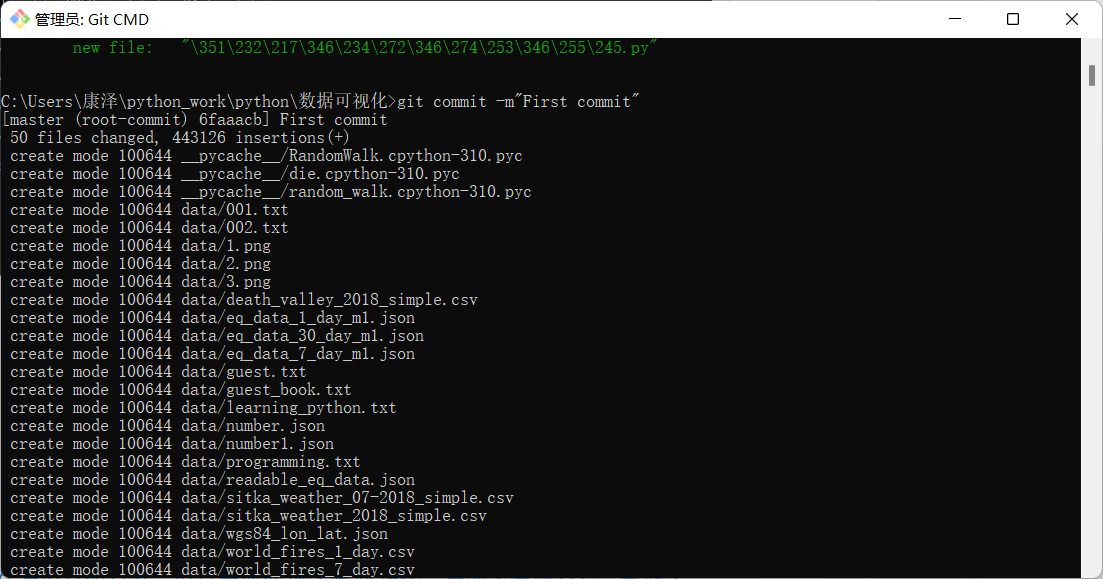
1. 检查状态



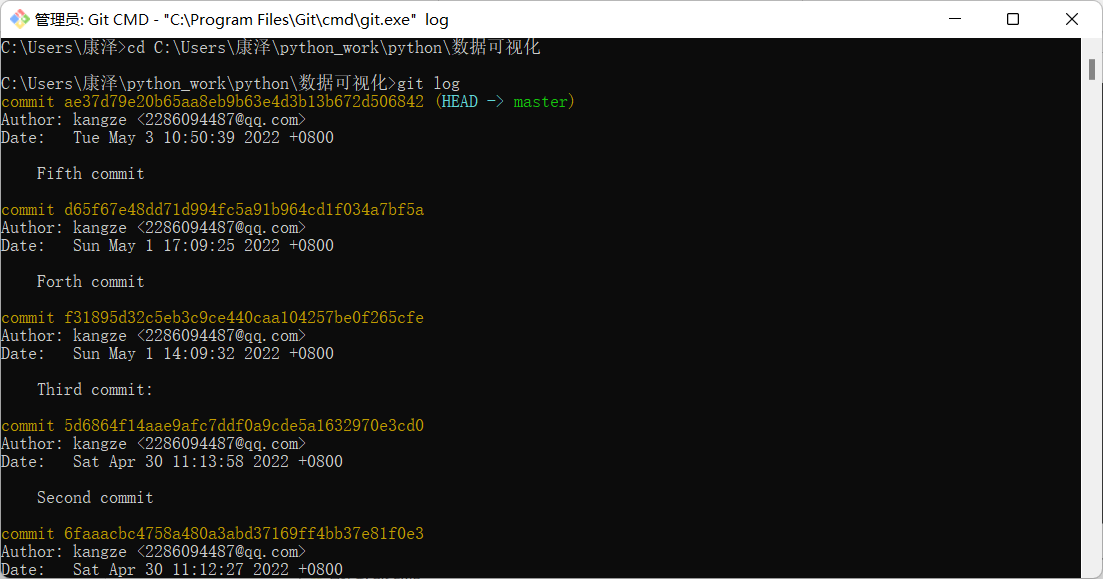
1. 添加文件



1. 提交



1. 查看提交历史



# 总 结

本系统主要实现了对数据的可视化操作，包括制作折线图、散点图以及直方图等等，通过拿到用户的数据可以制作相应的可视化图形，从而解决用户的需求，主要的操作是能够绘制模拟掷骰子、天气数据、全球地震散点图以及使用Web API获取数据并绘制相应的图形。

# 参考文献

埃里克—马瑟斯. Python编程从入门到实践[M]. 第2版. 人民邮电出版社:中国工信出版集团, 2021年.

## 附 录

### 源代码：

* **散点图：**

import json

import plotly.express as px

import pandas as pd

filename = 'data/eq\_data\_30\_day\_m1.json'

with open(filename) as f:

    all\_eq\_data = json.load(f)

readable\_file = 'data/readable\_eq\_data.json'

with open(readable\_file, 'w') as f:

    json.dump(all\_eq\_data, f, *indent*=4)

all\_eq\_dicts = all\_eq\_data['features']

mags = []

titles = []

lons = []

lats = []

for eq\_dict in all\_eq\_dicts:

    mag = eq\_dict['properties']['mag']

    title = eq\_dict['properties']['title']

    lon = eq\_dict['geometry']['coordinates'][0]

    lat = eq\_dict['geometry']['coordinates'][1]

    titles.append(title)

    lons.append(lon)

    lats.append(lat)

    mags.append(mag)

data = pd.DataFrame(*data*=zip(lons, lats, titles, mags),

*columns*=['经度', '纬度', '位置', '震级'])

data.head()

fig = px.scatter(data,

*x*='经度',

*y*='纬度',

*range\_x*=[-200, 200],

*range\_y*=[-90, 90],

*width*=800,

*height*=800,

*title*='全球地震散点图',

*size*='震级',

*size\_max*=10,

*color*='震级',

*hover\_name*='位置')

fig.write\_html('global\_earthquakes\_30\_day.html')

fig.show()

import matplotlib.pyplot as plt

from matplotlib.transforms import Bbox

# x\_values = [1, 2, 3, 4, 5]

# y\_values = [1, 4, 9, 16, 25]

x\_values = range(1, 1001)

y\_values = [x\*\*2 for x in x\_values]

plt.style.use('seaborn')

fig, ax = plt.subplots()

ax.scatter(x\_values, y\_values, *c*=y\_values, *cmap*=plt.cm.Blues, *s*=10)

# ax.plot(input\_values, squares, linewidth=3)

ax.set\_title("Square number", *fontsize*=24)

ax.set\_xlabel("values", *fontsize*=14)

ax.set\_ylabel("Square of value", *fontsize*=14)

ax.tick\_params(*axis*='both', *which*='major', *labelsize*=14)

ax.axis([0, 1100, 0, 1100000])

plt.savefig('散点图.png', *bbox\_inches*='tight')

* **随机漫步：**

import matplotlib.pyplot as plt

from random\_walk import RandomWalk

while True:

    rw = RandomWalk(50\_000)

    rw.fill\_walk()

    plt.style.use('classic')

    fig, ax = plt.subplots(*figsize*=(15, 9))

    point\_numbers = range(rw.num\_points)

    ax.scatter(rw.x\_values,

               rw.y\_values,

*c*=point\_numbers,

*cmap*=plt.cm.Blues,

*edgecolors*='none',

*s*=1)

    ax.scatter(0, 0, *c*='green', *edgecolors*='none', *s*=100)

    ax.scatter(rw.x\_values[-1],

               rw.y\_values[-1],

*c*='red',

*edgecolors*='none',

*s*=100)

    # plt.show()

    plt.savefig('随机漫游图.png', *bbox\_inches*='tight')

    keep\_running = input('Make another walk?(y/n):')

    if keep\_running == 'n':

        break

* **掷骰子：**

from die import Die

from plotly.graph\_objs import Bar, Layout

from plotly import offline

die\_1 = Die()

die\_2 = Die(10)

die\_3 = Die(20)

results = []

for roll\_num in range(50000):

    result = die\_1.roll() + die\_2.roll()+die\_3.roll()

    results.append(result)

frequencies = []

max\_result = die\_1.num\_sides + die\_2.num\_sides+die\_3.num\_sides

for value in range(3, max\_result + 1):

    frequency = results.count(value)

    frequencies.append(frequency)

x\_values = list(range(3, max\_result + 1))

data = [Bar(*x*=x\_values, *y*=frequencies)]

x\_axis\_config = {'title': '结果', 'dtick': 1}

y\_axis\_config = {'title': '结果的频率'}

my\_layout = Layout(*title*='掷一个D6、一个D10以及一个D20 50000次的结果',

*xaxis*=x\_axis\_config,

*yaxis*=y\_axis\_config)

offline.plot({'data': data, 'layout': my\_layout}, *filename*='d6\_d10\_d20.html')

* **下载数据：**
* import csv
* from datetime import datetime
* import matplotlib.pyplot as plt
* filename = 'data/death\_valley\_2018\_simple.csv'
* with open(filename) as f:
* reader = csv.reader(f)
* header\_row = next(reader)
* highs = []
* dates = []
* lows = []
* for row in reader:
* current\_date = datetime.strptime(row[2], '%Y-%m-%d')
* try:
* high = int(row[4])
* low = int(row[5])
* except ValueError:
* print(f'Missing data for {current\_date}')
* else:
* highs.append(high)
* dates.append(current\_date)
* lows.append(low)
* plt.style.use('seaborn')
* fig, ax = plt.subplots(*figsize*=(15, 9))
* ax.plot(dates, highs, *c*='red', *alpha*=0.5)
* ax.plot(dates, lows, *c*='blue', *alpha*=0.5)
* fig.autofmt\_xdate()
* ax.fill\_between(dates, highs, lows, *facecolor*='blue', *alpha*=0.1)
* ax.set\_title('Daily maximum temperature in 2018', *fontsize*=24)
* ax.set\_xlabel('', *fontsize*=16)
* ax.set\_ylabel('Temperature(F)', *fontsize*=16)
* ax.tick\_params(*axis*='both', *which*='major', *labelsize*=16)
* # plt.show()
* plt.savefig('死亡谷2018年每日最高最低温度.png', *bbox\_inches*='tight')
* **WebAPI:**

import requests

from plotly.graph\_objs import Bar

from plotly import offline

# Make an API call and store the response.

url = 'https://api.github.com/search/repositories?q=language:Go&sort=stars'

headers = {'Accept': 'application/vnd.github.v3+json'}

r = requests.get(url, *headers*=headers)

print(f"Status code: {r.status\_code}")

# Process results.

response\_dict = r.json()

repo\_dicts = response\_dict['items']

repo\_links, stars, labels = [], [], []

for repo\_dict in repo\_dicts:

    repo\_name = repo\_dict['name']

    repo\_url = repo\_dict['html\_url']

    repo\_link = f"<a href='{repo\_url}'>{repo\_name}</a>"

    repo\_links.append(repo\_link)

    stars.append(repo\_dict['stargazers\_count'])

    owner = repo\_dict['owner']['login']

    description = repo\_dict['description']

    label = f"{owner}<br />{description}"

    labels.append(label)

# Make visualization.

data = [{

    'type': 'bar',

    'x': repo\_links,

    'y': stars,

    'hovertext': labels,

    'marker': {

        'color': 'rgb(60, 100, 150)',

        'line': {

            'width': 1.5,

            'color': 'rgb(25, 25, 25)'

        }

    },

    'opacity': 0.6,

}]

my\_layout = {

    'title': 'GitHub上最受欢迎的Go项目',

    'titlefont': {

        'size': 28

    },

    'xaxis': {

        'title': '仓库',

        'titlefont': {

            'size': 24

        },

        'tickfont': {

            'size': 14

        },

    },

    'yaxis': {

        'title': '星数',

        'titlefont': {

            'size': 24

        },

        'tickfont': {

            'size': 14

        },

    },

}

fig = {'data': data, 'layout': my\_layout}

offline.plot(fig, *filename*='go\_repos.html')

### 运行结果图：

