《Python程序设计基础》程序设计作品说明书

题目： 数据可视化

学院： 21计科02

姓名： 刘嘉璐

学号： B20210906220

指导教师： 周景

起止日期：2023.11.10-2023.12.10

摘要

我根据代码规范要求完成数据可视化项目，并对系统的主要功能进行测试，最后将材料提交至远程仓库`Github`。此外，我还实现练习15-7的同时投掷三个骰子、练习16-3的对旧金山、死亡谷和锡斯卡的天气数据进行研究比较以及练习17-1的其他语言将七门编程语言的受欢迎仓库数据可视化。最后，我对`Die`类、`get_repos_info`函数和`get_response_dict`函数进行测试，并将材料提交至远程仓库`Github`。

关键词：数据可视化 Matplotlib Plotly CSV GeoJSON API

第1章 需求分析

需求功能和用处如下：

1. 实现教材[1]15章的功能：使用`Plotly`模拟投掷骰子。该功能可以帮助用户更好地理解概率和随机事件，并且在游戏和赌博等领域有一定的应用。
2. 实现教材16章的功能：16.1 `CSV`文件格式，绘制天气数据的折线图。此功能可以帮助用户更好地了解天气的状况，并对其进行可视化分析，从而将其简单直接应用于相关需求领域。
3. 实现教材16章的功能：16.2 制作全球地震散点图。此功能可以帮助用户更好地了解天地震等自然现象，并对其进行可视化分析，从而更好的为地震等灾害做准备，将损失最小化。
4. 实现教材17章的功能：使用`Web API`获取`Github`的数据。此功能可以帮助用户迅速快捷得到代码仓库的相关信息，并且能够对代码质量和开发进度进行有效的监控和评估。
5. 实现教材17章的功能：使用`Plotly`可视化仓库。此功能可以帮助用户更好地管理代码仓库，并且能够对代码质量和开发进度进行有效的监控和评估。
6. 实现教材练习的功能：练习15-7 同时投掷三个骰子。该功能可以帮助用户更好地理解概率和随机事件，并且在游戏和赌博等领域有一定的应用。同时有助于提升编写者的`Python`编程能力。
7. 实现教材练习的功能：练习16-3 对旧金山的天气数据进行研究分并绘制图表。此练习将旧金山、锡斯卡和死亡谷的数据进行直观比较分析，同时帮助编写者更好地掌握Python编程知识，并提高其对数据分析和可视化处理的能力。
8. 实现教材练习的功能：练习17-1 其他语言。此练习将不同的语言中最受欢迎的仓库可视化，使数据更加直观清晰。同时，帮助编写者更好地掌握`Python`编程知识，并提高其对数据分析和可视化处理的能力。

第2章 分析与设计

系统架构：

- 基于`Python`语言开发，使用`Plotly`等数据处理库对数据进行清洗和处理。
- 使用`Matplotlib`、`Seaborn`、`Plotly`等数据可视化库生成各种类型的图表。

系统流程：

1. 数据获取：从线上2开放数据源或本地数据库中获取需要进行可视化的数据集。
2. 数据预处理：对数据进行清洗、筛选、聚合等操作，使其适合进行可视化处理。
3. 可视化处理：使用`Matplotlib`、`Seaborn`、`Plotly`等库进行数据可视化处理，生成各种图表，如折线图、散点图、条形图等。
4. 数据展示：将生成的图表嵌入到Web页面中，让用户可以查看不同的可视化结果。

系统模块：

1. 数据获取模块：负责数据的获取和存储。
2. 数据预处理模块：对原始数据进行清洗、转换、汇总等处理。
3. 可视化处理模块：使用`Matplotlib`、`Seaborn`、`Plotly`等库进行数据可视化处理。

关键实现：

- 使用`Pandas`进行数据清洗和处理，以确保数据的准确性和完整性。
- 使用`Matplotlib`、`Seaborn`、`Plotly`库生成可视化图表，包括但不限于折线图、散点图等，提高用户体验和数据探索的便利性。

功能代码和实现效果：

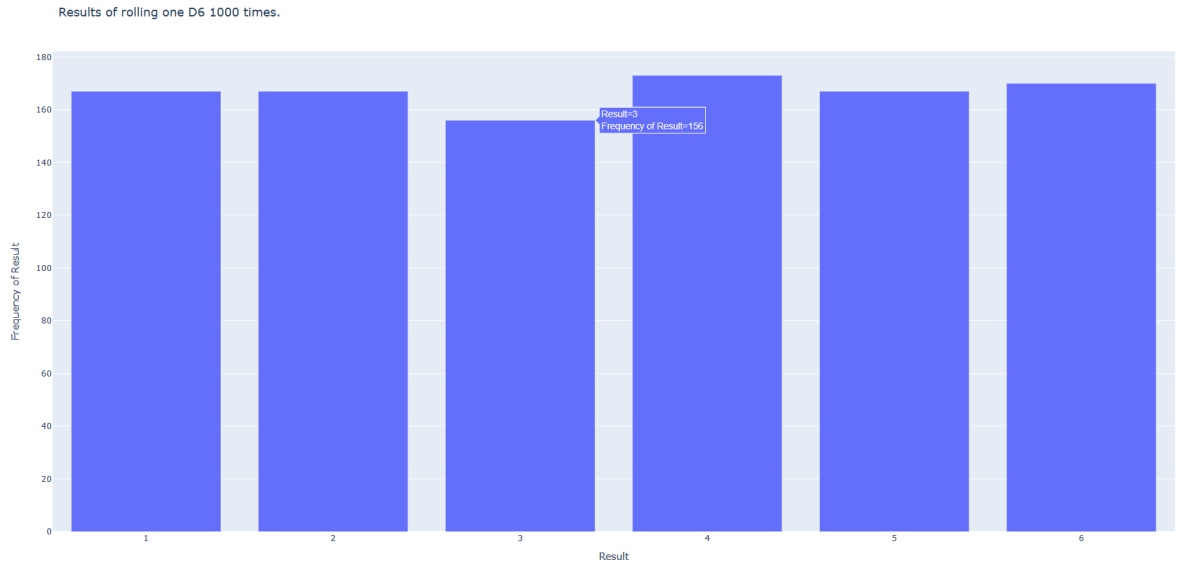
1. 教材15章：使用`Plotly`模拟投掷骰子。

代码：

```python
# 1
from random import randint

class Die():
    '''表示一个骰子的类'''

    def __init__(self, num_sides=6):
        '''骰子默认为6面的'''
        self.num_sides = num_sides
    def roll(self):
        '''返回一个介于1与骰子面数之间的随机值'''
        return randint(1, self.num_sides)

'''
# 创建一个D6
die = Die()

# 掷几次骰子并将结果存储在一个列表中
```

```
results = []
for roll_num in range(1000):
    result = die.roll()
    results.append(result)

# print(results)

# 分析结果
frequencies = []
poss_results = range(1,die.num_sides+1)
for value in poss_results:
    frequency = results.count(value)
    frequencies.append(frequency)

print(frequencies)
'''


import plotly.express as px

# from die import Die

# 创建一个D6
die = Die()

# 掷几次骰子并将结果存储在一个列表中
results = []
for roll_num in range(1000):
    result = die.roll()
    results.append(result)

# 分析结果
frequencies = []
poss_results = range(1,die.num_sides+1)
for value in poss_results:
    frequency = results.count(value)
    frequencies.append(frequency)

# 对结果进行可视化
title = "Results of rolling one D6 1000 times."
labels = {'x':'Result','y':'Frequency of Result'}
fig = px.bar(x=poss_results, y=frequencies,title = title,labels = labels) # px.bar()直方图
# fig = px.scatter(x=poss_results, y=frequencies) # px.scatter()散点图
# fig = px.line(x=poss_results, y=frequencies) # px.line()折线图

fig.show()

```

效果图:

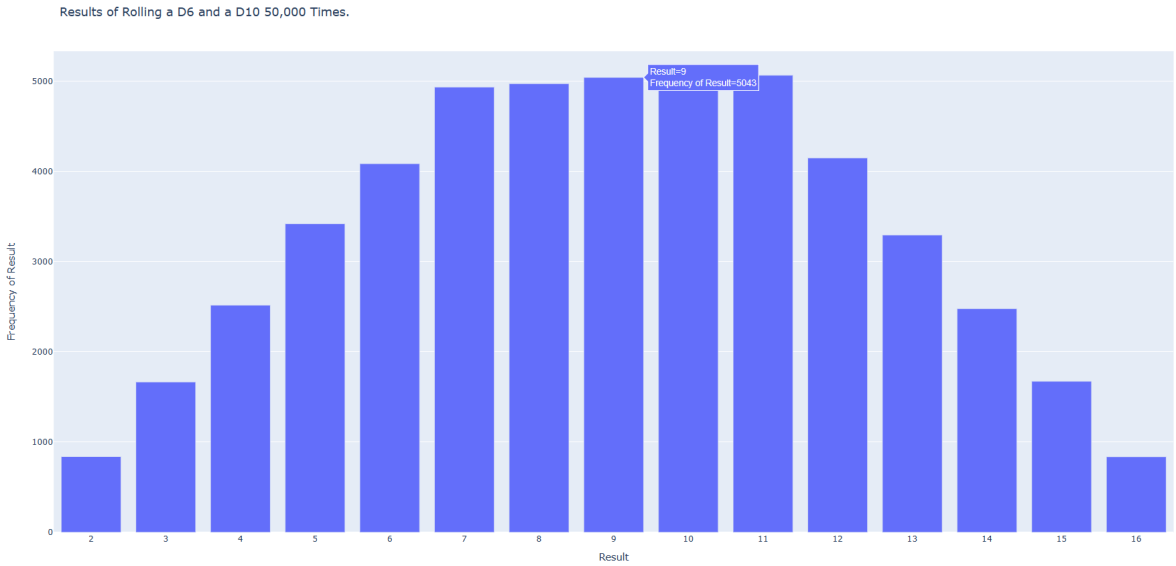


代码：

```python
#2 两个骰子
from random import randint

class Die():
    '''表示一个骰子的类'''

    def __init__(self, num_sides=6):
        '''骰子默认为6面的'''
        self.num_sides = num_sides
    def roll(self):
        '''返回一个介于1与骰子面数之间的随机值'''
        return randint(1, self.num_sides)


import plotly.express as px
# from die import Die

# 创建两个骰子
# die_1 = Die()
# die_2 = Die()

# 创建一个D6和一个D10
die_1 = Die()
die_2 = Die(10)

# 掷骰子多次，并将结果存储到一个列表中
results = []
for roll_num in range(50_000):
    result = die_1.roll() + die_2.roll()
    results.append(result)

# 分析结果
frequencies = []
max_result = die_1.num_sides + die_2.num_sides
poss_results = range(2, max_result+1)
for value in poss_results:
    frequency = results.count(value)
```

```
    frequencies.append(frequency)

# 可视化结果
# title = "Results of Rolling Two D6 Dice 1,000 Times."
title = "Results of Rolling a D6 and a D10 50,000 Times."
labels = {'x':'Result','y':'Frequency of Result'}
fig = px.bar(x=poss_results,y=frequencies,title=title,labels=labels)

# 进一步定制图形
# xaxis_dtick指定x轴上刻度标记的间距，此处设置为1
fig.update_layout(xaxis_dtick=1)

# fig.show()
fig.write_html('dice_visual_d6d10.html')

```

效果图：



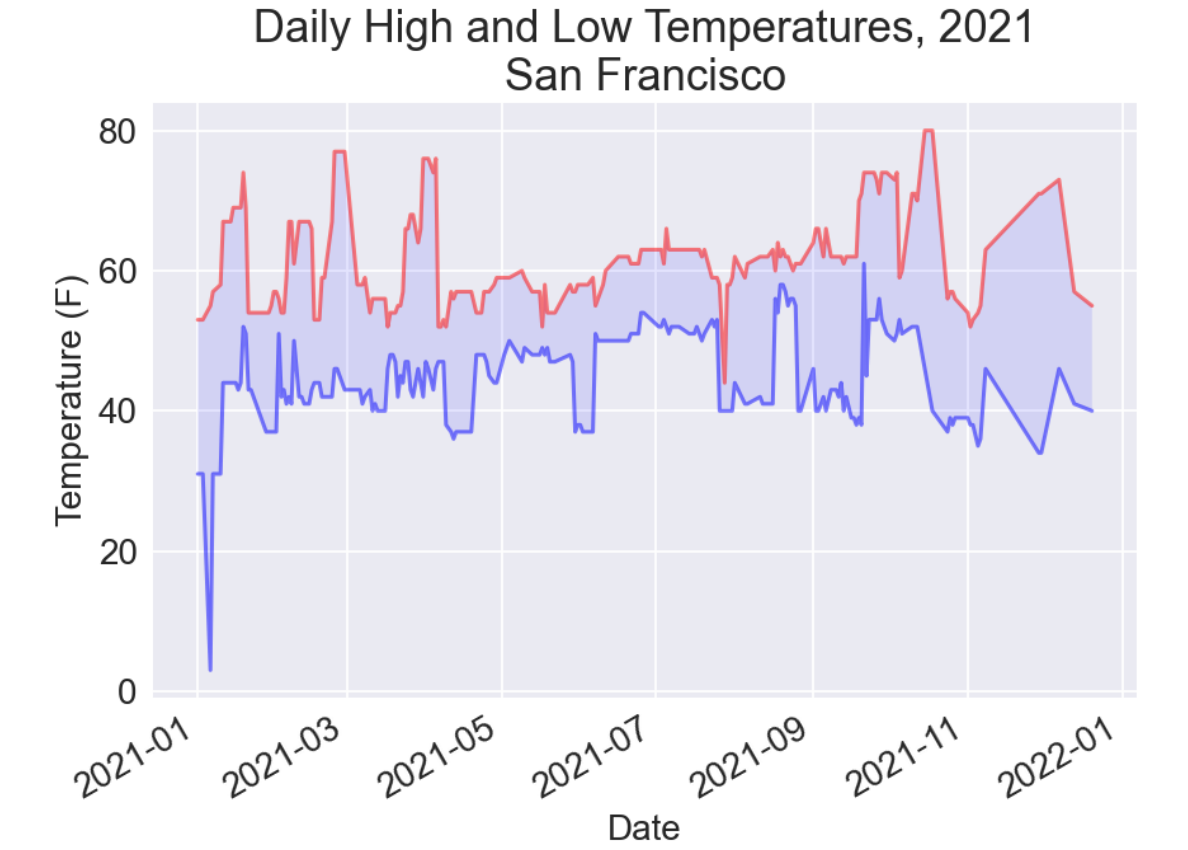
2. 教材16章：16.1 `CSV`文件格式，绘制天气数据的折线图。

代码：

```
"""
CSV文件格式，绘制天气数据的折线图
"""
from pathlib import Path
import csv
import matplotlib.pyplot as plt
from datetime import datetime

path = Path('16\san_francisco_2021_simple.csv')
lines = path.read_text().splitlines()

reader = csv.reader(lines)
header_row = next(reader)

# 提取日期和最高温度
dates,highs,lows = [],[],[]
for row in reader:

```

```
    current_date = datetime.strptime(row[1],'%Y-%m-%d')
    try:
        high = int(row[2])
        low = int(row[3])
    except ValueError:
        print(f"Missing data of {current_date}")
    else:
        dates.append(current_date)
        highs.append(high)
        lows.append(low)

# 根据最高温度绘图
plt.style.use('seaborn-v0_8')
fig,ax = plt.subplots()
ax.plot(dates,highs,color='red',alpha = 0.5)
ax.plot(dates,lows,color='blue',alpha = 0.5)
ax.fill_between(dates,highs,lows,facecolor='blue',alpha=0.1)

# 设置绘图格式
title = "Daily High and Low Temperatures, 2021\nSan Francisco"
ax.set_title(title,fontsize=20)
ax.set_xlabel('Date',fontsize=16)
fig.autofmt_xdate()
ax.set_ylabel("Temperature (F)",fontsize=16)
ax.tick_params(labelsize=16)

plt.show()

```

效果图:



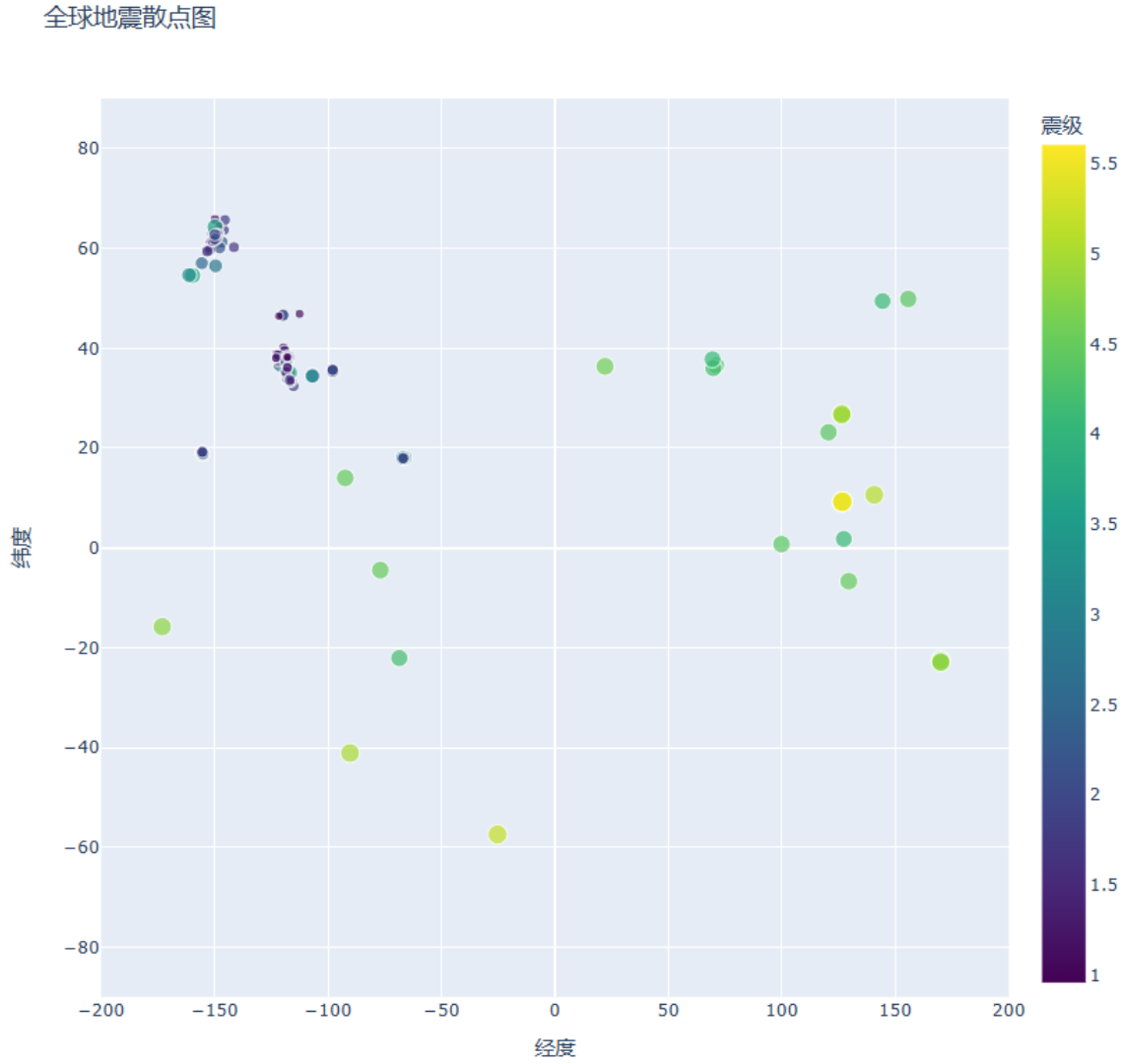
3. 教材16章：16.2 制作全球地震散点图。

代码：

```
"""
16.2 制作全球地震散点图
"""
from pathlib import Path
import json
import plotly.express as px

# 将数据作为字符串读取并转换为Python对象
path = Path('16\eq_data_1_day_m1.geojson')
contents = path.read_text()
all_eq_data = json.loads(contents)

# 将数据文件转换为更易于阅读的版本
path = Path('16\eq_data_1_day_m1.geojson')
readable_contents = json.dumps(all_eq_data, indent=4)
path.write_text(readable_contents)

# 查看数据集中的所有地震
all_eq_dicts = all_eq_data['features']
# print(len(all_eq_dicts))

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]
    mags.append(mag)
    titles.append(title)
    lons.append(lon)
    lats.append(lat)

print(mags[:10])  # 震级
print(titles[:2])  # 标题
print(lons[:5])  # 经度
print(lats[:5])  # 纬度


# 以300dpi为例，就是2.54cm*2.54cm的单位面积上有300*300个像素点，
# 把像素当成一个个小点。像素点在图片上是均匀分布的，
# 一寸照的尺寸为2.5cm*3.5cm，
# 在宽度上能放下300*2.5/2.54=295px个像素点，
# 那么长度上像素是300*3.5/2.54=413px，则一寸照的像素是413px*295px。

# 创建fig实例
'''
fig = px.scatter(
    x=lons,   # 经度
    y=lats,   # 纬度
    labels={'x':'经度','y':'纬度'},
    range_x=[-200,200],  # 扩大空间，完整显示+-180附近的地震点
    range_y=[-90,90],
```

```
    width=800,  # 宽度800像素
    height=800,  # 高度800像素
    title='全球地震散点图',  # 标题
)
fig.write_html('global_earthquakes.html')
fig.show()
'''

import pandas as pd
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],  # 扩大空间，完整显示+-180附近的地震点
    range_y=[-90,90],
    width=800,  # 宽度800像素
    height=800,  # 高度800像素
    title='全球地震散点图',  # 标题
    size='震级',  # data中的'震级'字段提供给size参数指定散点图中每个标记的尺寸
    size_max=10,  # 最大显示尺寸
    color='震级',
    color_continuous_scale='Viridis',  # 设置不同震级的渐变颜色
    hover_name='位置',
)
fig.write_html('global_earthquakes.html')
fig.show()

```

效果图:



4. 教材17章：使用`Web API`获取`Github`的数据。

代码：

```python
"""
使用Web API获取Github的数据
"""
import time
import requests

# 执行API调用并查看响应
url = "https://api.github.com/search/repositories" # url的主要部分
url += "?q=language:python+sort:stars+stars:>10000"  # 查询字符串

headers = {"Accept": "application/vnd.github.v3+json"}
r = requests.get(url, headers=headers)
print(f"Status code: {r.status_code}")  # 状态码200表示请求成功

# 将响应转换为字典
response_dict = r.json()
print(f"Total repositories:{response_dict['total_count']}")
print(f"Complete results:{not response_dict['incomplete_results']}")

# 探索有关仓库的信息
repo_dicts = response_dict["items"]
```

```python
print(f"Repositories returned: {len(repo_dicts)}")

# 研究第一个仓库
repo_dict = repo_dicts[0]
# print(f"\nKeys: {len(repo_dict)}")
# for key in sorted(repo_dict.keys()):
#     print(key)

# 研究第一个仓库
# print("\nSelected information about first repository:")
# print(f"Name: {repo_dict['name']}")
# print(f"Owner: {repo_dict['owner']['login']}")
# print(f"Stars: {repo_dict['stargazers_count']}")
# print(f"Repository: {repo_dict['html_url']}")
# print(f"Created: {repo_dict['created_at']}")
# print(f"Updated: {repo_dict['updated_at']}")
# print(f"Description: {repo_dict['description']}")


# 探索仓库包含的信息
print("\nSelected information about each repository:")
for repo_dict in repo_dicts:
    print(f"Name: {repo_dict['name']}")
    print(f"Owner: {repo_dict['owner']['login']}")
    print(f"Stars: {repo_dict['stargazers_count']}")
    print(f"Repository: {repo_dict['html_url']}")
    print(f"Created: {repo_dict['created_at']}")
    print(f"Updated: {repo_dict['updated_at']}")
    print(f"Description: {repo_dict['description']}")
    time.sleep(6)


# 处理结果
# print(response_dict.keys())

```

部分运行截图：

```
PS D:\python项目二 数据可视化\study> python -u "d:\python项目二 数据可视化\study\17\webapi17_3.py"
Status code: 200
Total repositories:425
Complete results:True
Repositories returned: 30

Selected information about each repository:
Name: public-apis
Owner: public-apis
Stars: 270356
Repository: https://github.com/public-apis/public-apis
Created: 2016-03-20T23:49:42Z
Updated: 2023-12-03T11:20:23Z
Description: A collective list of free APIs
Name: system-design-primer
Owner: donnemartin
Stars: 236790
Repository: https://github.com/donnemartin/system-design-primer
Created: 2017-02-26T16:15:28Z
Updated: 2023-12-03T10:03:08Z
Description: Learn how to design large-scale systems. Prep for the system design interview.  Includes Anki
flashcards.
Name: awesome-python
Owner: vinta
Stars: 188599
Repository: https://github.com/vinta/awesome-python
Created: 2014-06-27T21:00:06Z
Updated: 2023-12-03T11:30:39Z
```

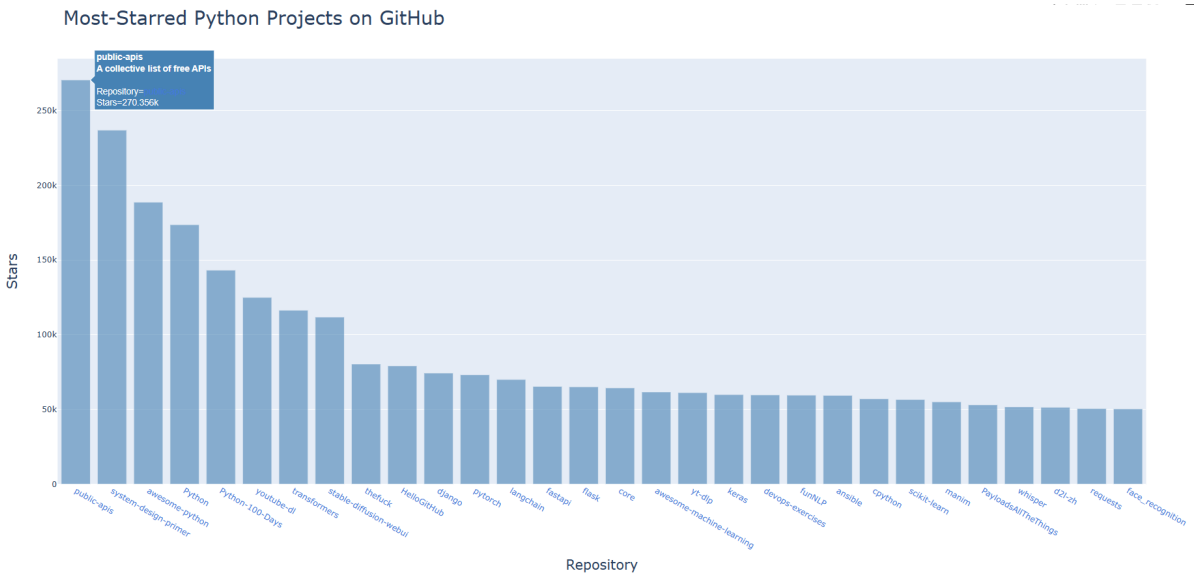
5. 教材17章：使用`Plotly`可视化仓库。

代码：

```python
"""
使用Plotly可视化仓库
"""
import requests
import plotly.express as px

# 执行API调用并查看响应
url = "https://api.github.com/search/repositories" # url的主要部分
url += "?q=language:python+sort:stars+stars:>10000"  # 查询字符串

headers = {"Accept": "application/vnd.github.v3+json"}
r = requests.get(url, headers=headers)
print(f"Status code: {r.status_code}")  # 状态码200表示请求成功

# 处理结果
response_dict = r.json()
print(f"Complete results:{not response_dict['incomplete_results']}")

# 处理有关仓库的信息
repo_dicts = response_dict["items"]
repo_links, stars,hover_texts = [], [], []
for repo_dict in repo_dicts:
    # 将仓库名转换为链接
    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"])

    # 创建悬停文本，在for循环中
    owner = repo_dict['owner']['login']
```

```
    description = repo_dict['description']
    hover_text = f"{owner}<br />{description}"
    hover_texts.append(hover_text)

# 可视化
# fig = px.bar(x=repo_names, y=stars)  # px.bar()函数创建一个条形图
# fig.show()

# 可视化
title = "Most-Starred Python Projects on GitHub"
labels = {'x': 'Repository', 'y': 'Stars'}
fig = px.bar(x=repo_links, y=stars, title=title, labels=labels,
hover_name=hover_texts)

fig.update_layout(title_font_size=28,
xaxis_title_font_size=20,yaxis_title_font_size=20)

fig.update_traces(marker_color='SteelBlue',marker_opacity=0.6)

fig.show()


def get_repos_info():
    # 执行API调用并查看响应
    url = "https://api.github.com/search/repositories" # url的主要部分
    url += "?q=language:python+sort:stars+stars:>10000"  # 查询字符串

    headers = {"Accept": "application/vnd.github.v3+json"}
    r = requests.get(url, headers=headers)
    print(f"Status code: {r.status_code}")  # 状态码200表示请求成功

    return r

def get_response_dict(response):
    # 处理结果
    response_dict = response.json()
    # print(f"Complete results:{not response_dict['incomplete_results']}")
    return response_dict

```

效果图:



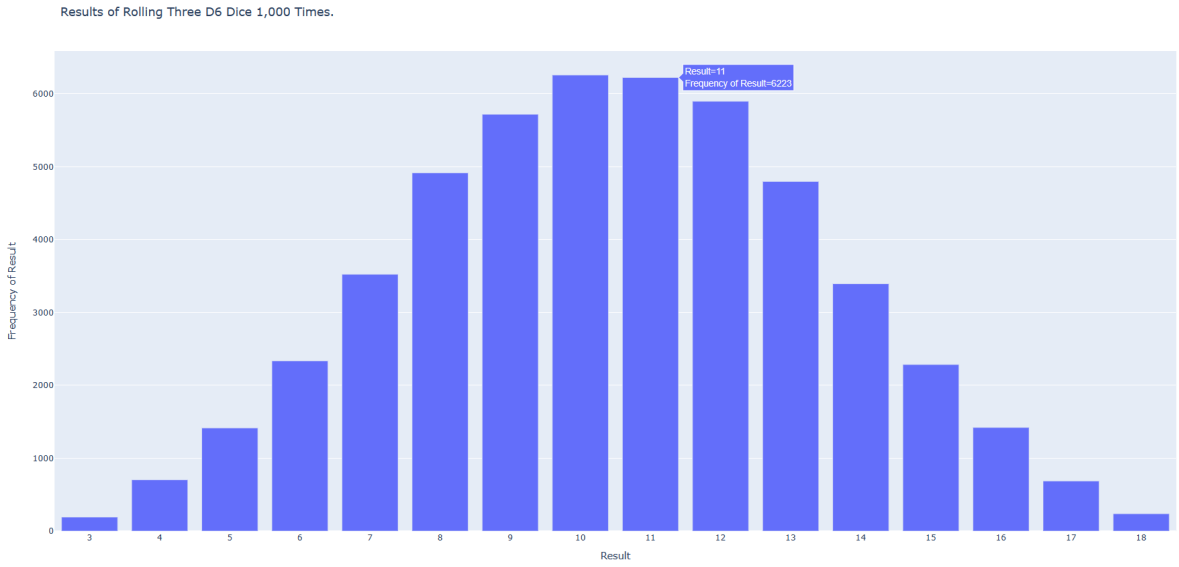
6. 练习15-7 同时投掷三个骰子。

代码：

```
"""
    练习15.7 同时掷三个骰子
"""
from random import randint

class Die():
    '''表示一个骰子的类'''

    def __init__(self, num_sides=6):
        '''骰子默认为6面的'''
        self.num_sides = num_sides
    def roll(self):
        '''返回一个介于1与骰子面数之间的随机值'''
        return randint(1, self.num_sides)


import plotly.express as px

# 创建三个骰子
die_1 = Die()
die_2 = Die()
die_3 = Die()

# 掷骰子多次，并将结果存储到一个列表中
results = []
for roll_num in range(50_000):
    result = die_1.roll() + die_2.roll() + die_3.roll()
    results.append(result)

# 分析结果
frequencies = []
max_result = die_1.num_sides + die_2.num_sides + die_3.num_sides
poss_results = range(3, max_result+1)
for value in poss_results:
    frequency = results.count(value)
```

```
    frequencies.append(frequency)

# 可视化结果
title = "Results of Rolling Three D6 Dice 1,000 Times."
labels = {'x':'Result','y':'Frequency of Result'}
fig = px.bar(x=poss_results,y=frequencies,title=title,labels=labels)

# 进一步定制图形
# xaxis_dtick指定x轴上刻度标记的间距，此处设置为1
fig.update_layout(xaxis_dtick=1)

fig.show()

# 可截图保存
```

效果图：



7. 练习16-3 对旧金山的天气数据进行研究分并绘制图表。

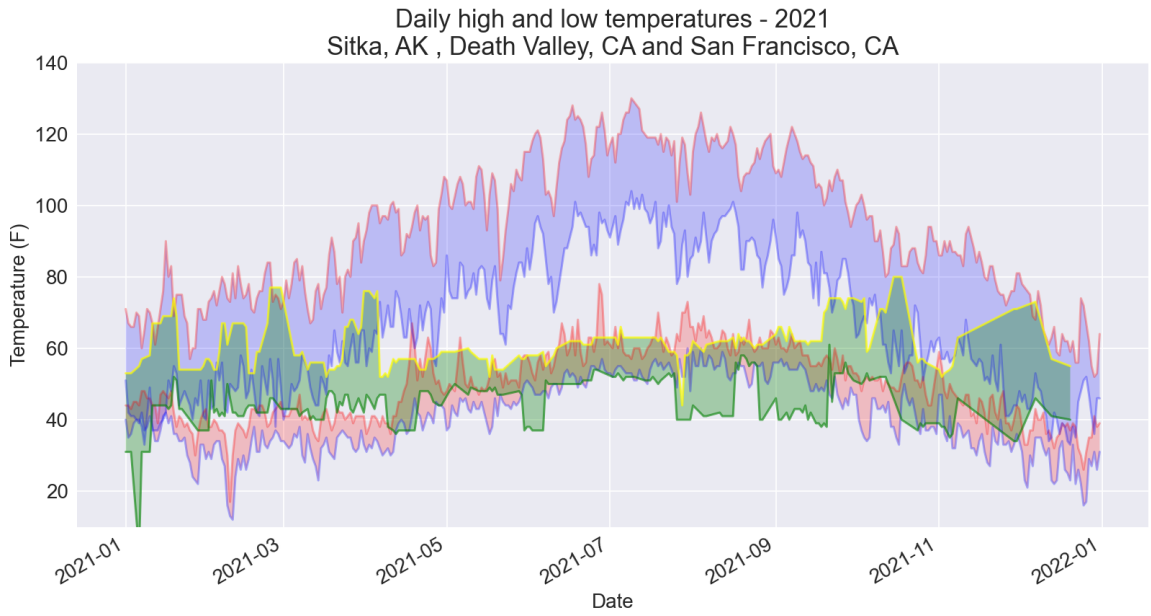
功能代码：

```
"""
    练习16.3 旧金山
    San Francisco Comparison
"""
from pathlib import Path
import csv
from datetime import datetime

import matplotlib.pyplot as plt


def get_weather_data(path, dates, highs, lows, date_index, high_index,
        low_index):
    """得到文件中最低温度和最高温度"""
    lines = path.read_text().splitlines()
    reader = csv.reader(lines)
    header_row = next(reader)
```

```
    # 提取日期，最高温和最低温
    for row in reader:
        current_date = datetime.strptime(row[date_index], '%Y-%m-%d')
        try:
            high = int(row[high_index])
            low = int(row[low_index])
        except ValueError:
            print(f"Missing data for {current_date}")
        else:
            dates.append(current_date)
            highs.append(high)
            lows.append(low)

# 获取锡特卡的天气数据。
path = Path('16\sitka_weather_2021_simple.csv')
dates, highs, lows = [], [], []
get_weather_data(path, dates, highs, lows, date_index=2, high_index=4,
        low_index=5)

# 绘制锡特卡的天气数据。
plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots()
ax.plot(dates, highs, color='red', alpha=0.3)
ax.plot(dates, lows, color='blue', alpha=0.3)
ax.fill_between(dates, highs, lows, facecolor='red', alpha=0.2)

# 获取死亡谷的天气数据。
path = Path('16\death_valley_2021_simple.csv')
dates, highs, lows = [], [], []
get_weather_data(path, dates, highs, lows, date_index=2, high_index=3,
        low_index=4)

# 将死亡谷天气数据添加到当前绘图中。
ax.plot(dates, highs, color='red', alpha=0.3)
ax.plot(dates, lows, color='blue', alpha=0.3)
ax.fill_between(dates, highs, lows, facecolor='blue', alpha=0.2)

# 获取旧金山的天气数据。
path = Path('16\san_francisco_2021_simple.csv')
dates, highs, lows = [], [], []
get_weather_data(path, dates, highs, lows, date_index=1, high_index=2,
        low_index=3)

# 将旧金山天气数据添加到当前绘图中。
ax.plot(dates, highs, color='yellow', alpha=0.7)
ax.plot(dates, lows, color='green', alpha=0.6)
ax.fill_between(dates, highs, lows, facecolor='green', alpha=0.3)

# 设置绘图格式。
title = "Daily high and low temperatures - 2021"
title += "\nSitka, AK , Death Valley, CA and San Francisco, CA"
ax.set_title(title, fontsize=20)
ax.set_xlabel('Date', fontsize=16)
fig.autofmt_xdate()
ax.set_ylabel("Temperature (F)", fontsize=16)
ax.tick_params(labelsize=16)
ax.set_ylim(10, 140)
```

```
76
77  plt.show()
```

效果图：



8. 练习17-1 其他语言。

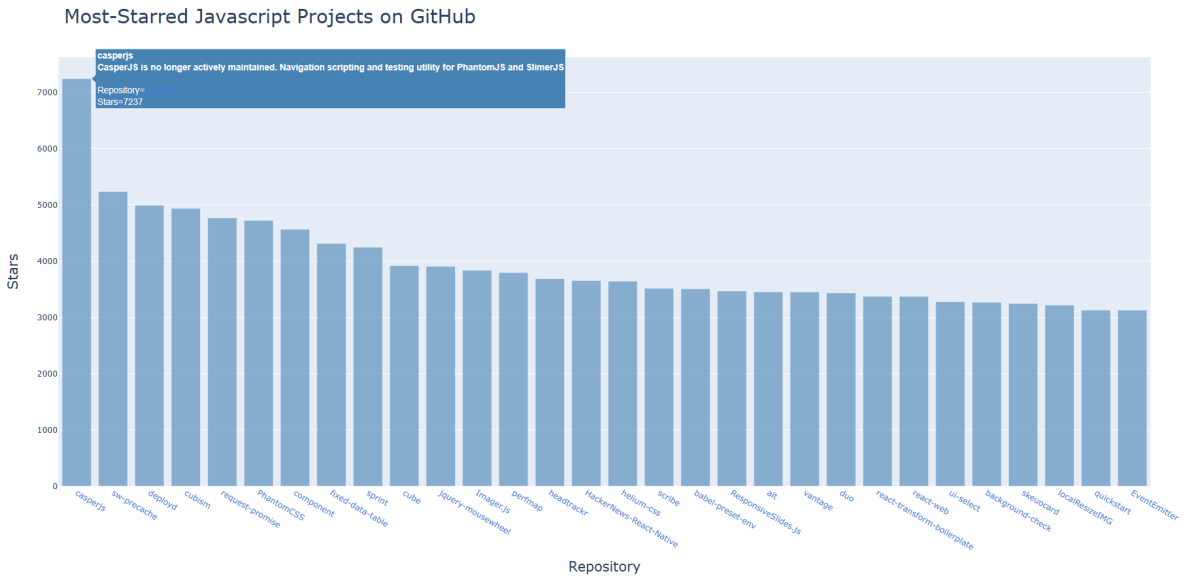
代码：

```
1   """
2   练习17.1 其他语言
3   """
4   import time
5   import requests
6   import plotly.express as px
7
8   # 执行API调用并查看响应
9   languages = ['javascript', 'ruby', 'c', 'java', 'perl', 'haskell', 'go']
10  for language in languages:
11      url = "https://api.github.com/search/repositories" # url的主要部分
12      # url += "?q=language:python+sort:stars+stars:>10000"  # 查询字符串
13      url += "?q=language:{}+sort:stars".format(language)
14
15      headers = {"Accept": "application/vnd.github.v3+json"}
16      r = requests.get(url, headers=headers)
17      print(f"Status code: {r.status_code}")  # 状态码200表示请求成功
18      time.sleep(6)
19
20      # 处理结果
21      response_dict = r.json()
22      try:
23          print(f"Complete results:{not response_dict['incomplete_results']}")
24      except KeyError:
25          print('Key not found')
26
27      # 处理有关仓库的信息
```

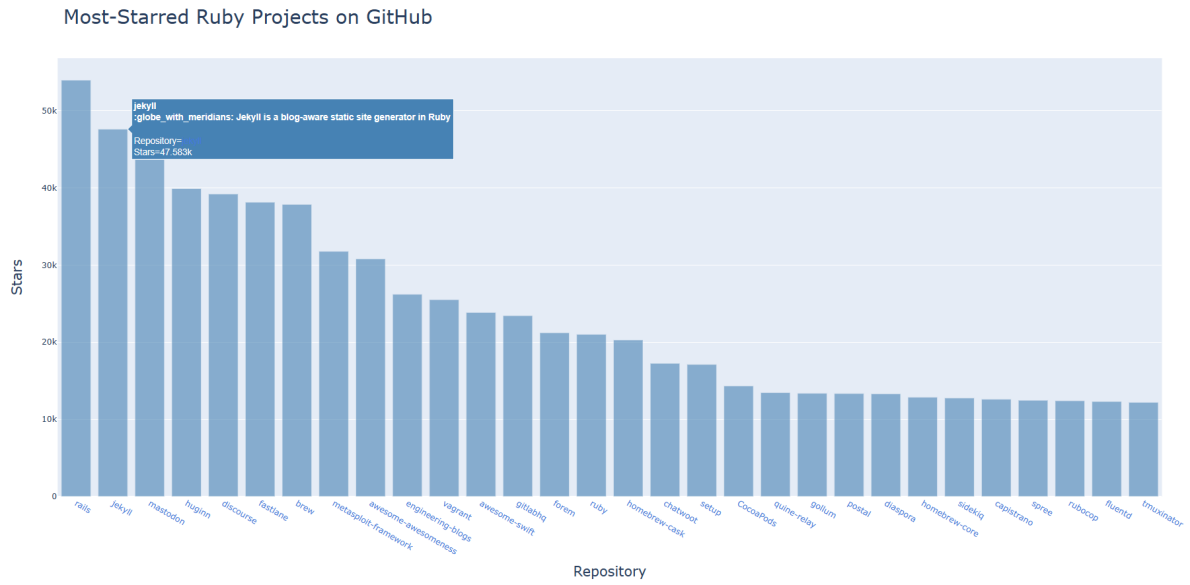
```
    repo_dicts = response_dict["items"]
    repo_links, stars,hover_texts = [], [], []
    for repo_dict in repo_dicts:
        # 将仓库名转换为链接
        repo_name = repo_dict['name']
        repo_url = repo_dict['html_url']
        repo_link = f"<a href='{repo_url}'>{repo_name}</a>"
        repo_links.append(repo_link)

        stars.append(repo_dict["stargazers_count"])

        # 创建悬停文本，在for循环中
        owner = repo_dict['owner']['login']
        description = repo_dict['description']
        hover_text = f"{owner}<br />{description}"
        hover_texts.append(hover_text)

    # 可视化
    title = f"Most-Starred {language.title()} Projects on GitHub"
    labels = {'x': 'Repository', 'y': 'Stars'}
    fig = px.bar(x=repo_links, y=stars, title=title, labels=labels,
hover_name=hover_texts)

    fig.update_layout(title_font_size=28,
xaxis_title_font_size=20,yaxis_title_font_size=20)

    fig.update_traces(marker_color='SteelBlue',marker_opacity=0.6)

    fig.show()

```

效果图:

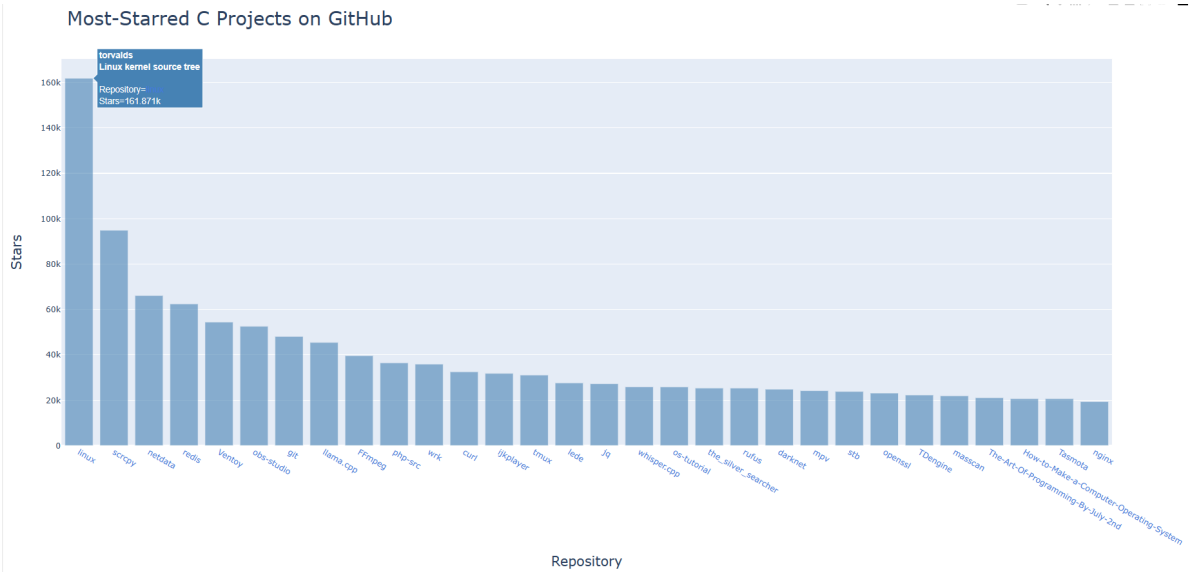
Javascript:



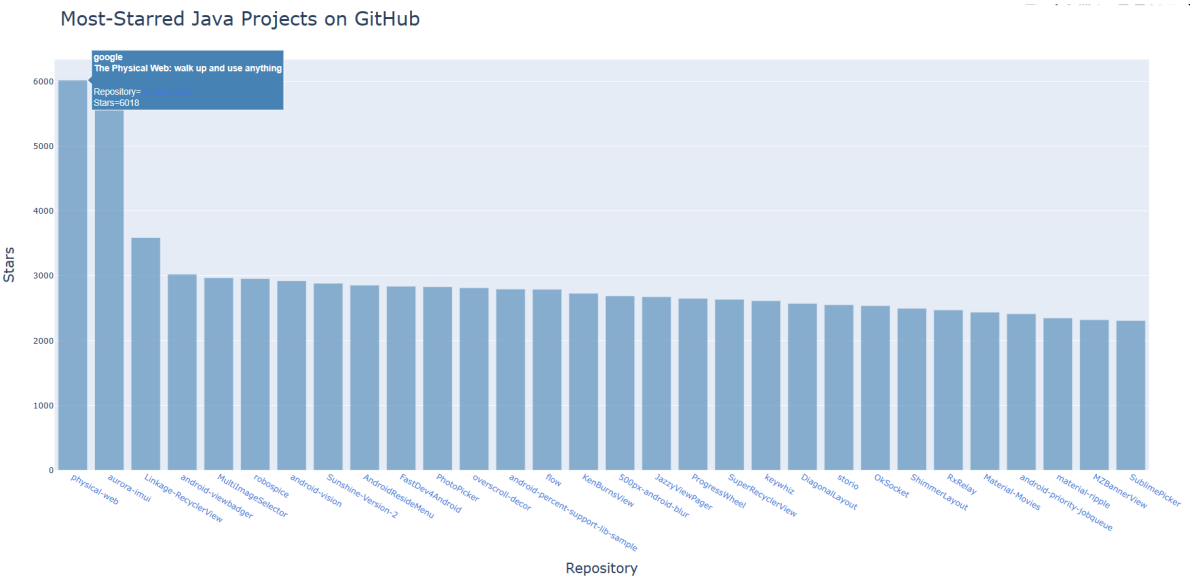
Ruby:


Most-Starred Ruby Projects on GitHub
jekyll
:globe_with_meridians: Jekyll is a blog-aware static site generator in Ruby
Repository=
Stars=47.583k
Stars
Repository
rails
jekyll
mastodon
huginn
discourse
fastlane
brew
metasploit-framework
awesome-awesomeness
engineering-blogs
vagrant
awesome-swift
gitlabhq
fpm
ruby
homebrew-cask
chatwoot
setup
CocoaPods
quine-relay
gollum
postal
diaspora
homebrew-core
sidekiq
capistrano
spree
rubocop
fluentd
tmuxinator

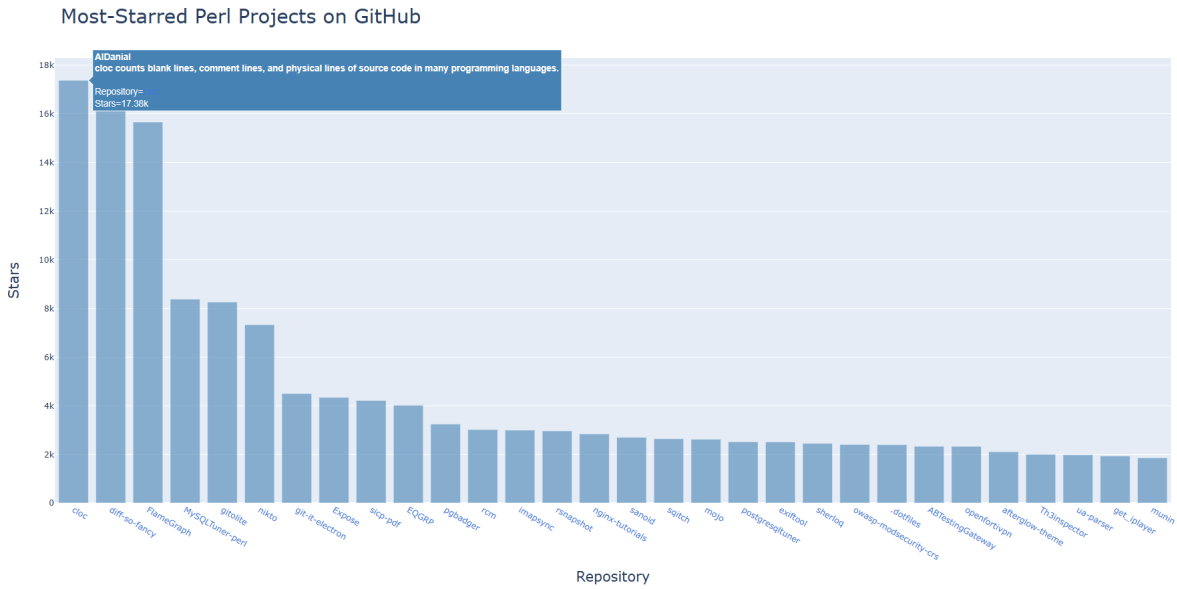
C:


Most-Starred C Projects on GitHub
torvalds
Linux kernel source tree
Repository=
Stars=161.871k
Stars
Repository
linux
scrcpy
netdata
redis
Ventoy
obs-studio
git
llama.cpp
FFmpeg
php-src
wrk
curl
ljkplayer
tmux
ledte
jq
whisper.cpp
os-tutorial
the_silver_searcher
rufus
darknet
mpv
stb
openssl
TDengine
masscan
The-Art-Of-Programming-By-July-2nd
How-to-Make-a-Computer-Operating-System
Tasmota
nginx

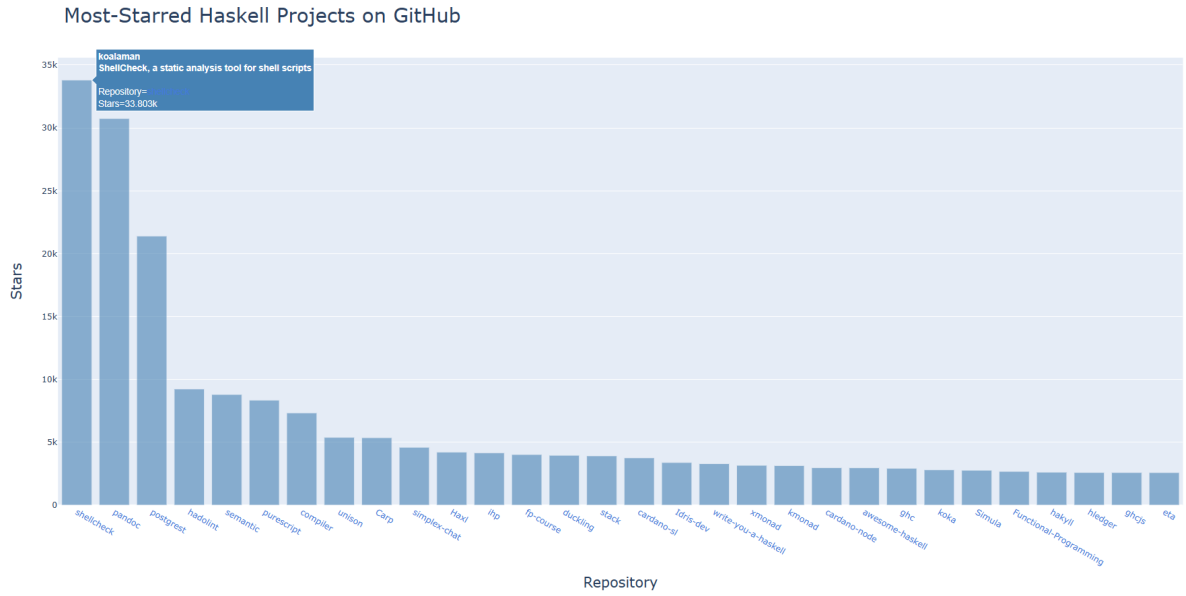
Java:


Most-Starred Java Projects on GitHub
google
The Physical Web: walk up and use anything
Repository=
Stars=6018
Stars
Repository
physical-web
aurora-imui
Linkage-RecyclerView
android-viewbadger
MultiImageSelector
robospice
android-vision
Sunshine-Version-2
AndroidResideMenu
FastDev4Android
PhotoPicker
overscroll-decor
android-percent-support-lib-sample
flow
KenBurnsView
500px-android-blur
JazzyViewPager
ProgressWheel
SuperRecyclerView
keywhiz
DiagonalLayout
storio
OkSocket
ShimmerLayout
RxRelay
Material-Movies
android-priority-jobqueue
material-ripple
MZBannerView
SublimePicker

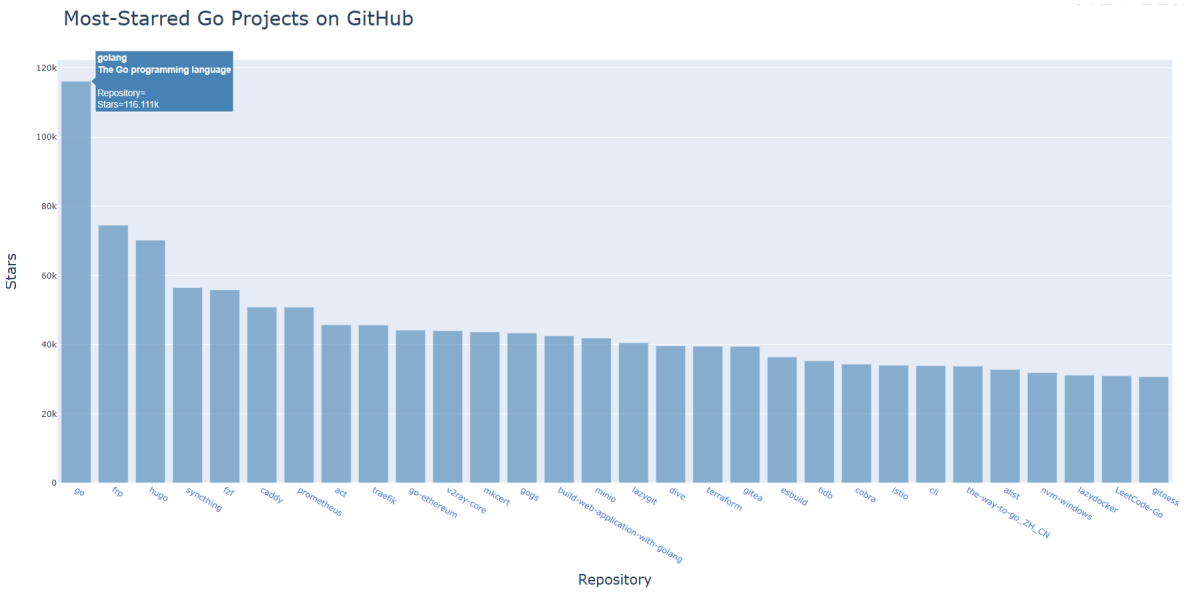
Perl:



Haskell:



Go:



第3章 软件测试

单元测试用例

#	测试目标	输入	预期结果	测试结果
1	Die类	6	6	6
2	get_repos_info函数	无	r.status_code=200	r.status_code=200
3	get_response_dict函数	url	total_count > 190 && complete_results=True	total_count > 190 && complete_results=True

测试目标`Die`:

测试类:

```
from random import randint

class Die():
    '''表示一个骰子的类'''

    def __init__(self, num_sides=6):
        '''骰子默认为6面的'''
        self.num_sides = num_sides
    def roll(self):
        '''返回一个介于1与骰子面数之间的随机值'''
        return randint(1, self.num_sides)
```

测试代码:

```
"""
test_exercise15_7
"""
from exercise15_7 import Die
from random import randint
import pytest

# 测试方法一：不使用夹具
def test_roll_within_range_6_sides():
    die = Die()
    result = die.roll()
    assert 1 <= result <= 6

# 测试方法二：使用夹具
@pytest.fixture
def die():
    '''创建一个骰子实例的夹具'''
    return Die()

def test_roll(die):
    '''测试掷骰子的结果是否在1到骰子面数之间'''
    result = die.roll()
    assert 1 <= result <= die.num_sides
```

```
24
```

测试效果截图(两种方法):

```
PS D:\python项目二 数据可视化\study> pytest
========================================= test session starts =========================================
platform win32 -- Python 3.11.5, pytest-7.4.3, pluggy-1.3.0
rootdir: D:\python项目二 数据可视化\study
plugins: anyio-4.1.0
collected 1 item

15\test_exercise15_7.py .                                                                    [100%]

========================================== 1 passed in 1.42s ==========================================
PS D:\python项目二 数据可视化\study> pytest
========================================= test session starts =========================================
platform win32 -- Python 3.11.5, pytest-7.4.3, pluggy-1.3.0
rootdir: D:\python项目二 数据可视化\study
plugins: anyio-4.1.0
collected 1 item

15\test_exercise15_7.py .                                                                    [100%]

========================================== 1 passed in 1.42s ==========================================
```

测试目标`get_repos_info`:

测试函数:

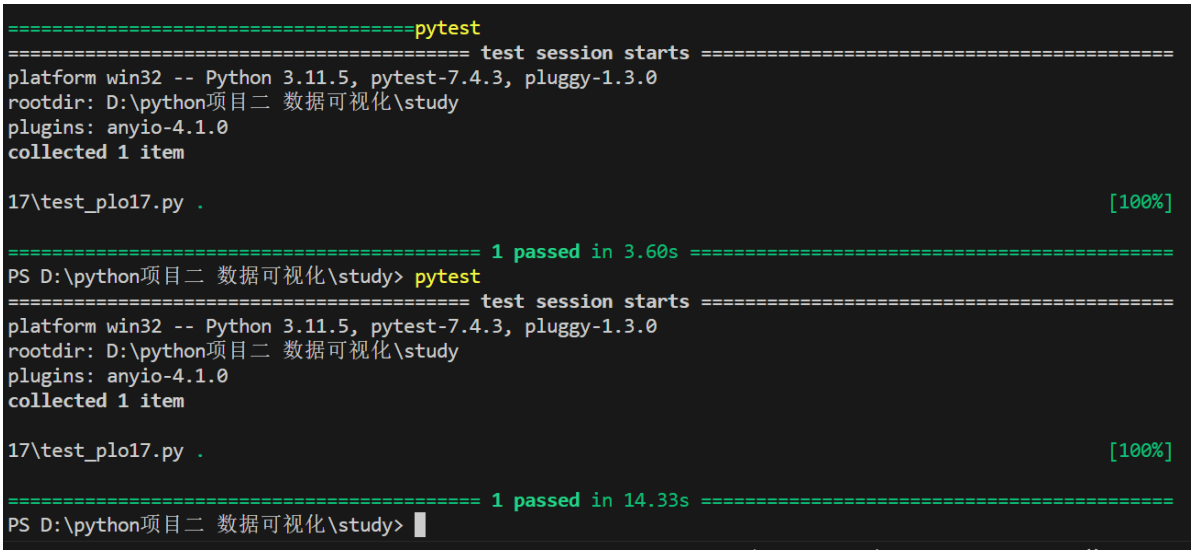
```python
def get_repos_info():
    # 执行API调用并查看响应
    url = "https://api.github.com/search/repositories" # url的主要部分
    url += "?q=language:python+sort:stars+stars:>10000"  # 查询字符串

    headers = {"Accept": "application/vnd.github.v3+json"}
    r = requests.get(url, headers=headers)
    print(f"Status code: {r.status_code}")  # 状态码200表示请求成功

    return r
```

测试代码:

```python
# """
# test_exercise15_7
# """
from exercise15_7 import Die
from random import randint
import pytest

# # 测试方法一：不使用夹具
def test_roll_within_range_6_sides():
    die = Die()
    result = die.roll()
    assert 1 <= result <= die.num_sides

# # 测试方法二：使用夹具
@pytest.fixture
def die():
    '''创建一个骰子实例的夹具'''
    return Die()

def test_roll(die):
```

```
    '''测试掷骰子的结果是否在1到骰子面数之间'''
    result = die.roll()
    assert 1 <= result <= die.num_sides

```

测试效果截图（两种方法）：



测试目标`get_response_dict`:

测试函数:

```
def get_response_dict(response):
    # 处理结果
    response_dict = response.json()
    # print(f"Complete results:{not response_dict['incomplete_results']}")
    return response_dict
```

测试代码:

```
"""
test_plo17
"""
import pytest
from plo17 import get_repos_info,get_response_dict

# 测试方法一：不使用夹具
def test_response_dict():
    '''验证是否表示了适当数量的存储库，以及结果是否完整。'''
    r = get_repos_info()
    response_dict = get_response_dict(r)

    total_count = response_dict['total_count']
    complete_results = not response_dict['incomplete_results']

    assert total_count > 190
    assert complete_results


# 测试方法二：使用夹具
@pytest.fixture
```

```python
def response():
    """获取一个响应对象。"""
    r = get_repos_info()
    return r

def test_response_dict(response):
    """验证是否表示了适当数量的存储库，以及结果是否完整。"""
    response_dict = get_response_dict(response)

    total_count = response_dict['total_count']
    complete_results = not response_dict['incomplete_results']

    assert total_count > 190
    assert complete_results
```

测试效果截图（两种方法）：

```
========================================= 1 passed in 11.71s ==========================================
PS D:\python项目二 数据可视化\study> pytest
========================================= test session starts =========================================
platform win32 -- Python 3.11.5, pytest-7.4.3, pluggy-1.3.0
rootdir: D:\python项目二 数据可视化\study
plugins: anyio-4.1.0
collected 1 item

17\test_plo17.py .                                                                           [100%]

========================================= 1 passed in 3.72s ===========================================
PS D:\python项目二 数据可视化\study> pytest
========================================= test session starts =========================================
platform win32 -- Python 3.11.5, pytest-7.4.3, pluggy-1.3.0
rootdir: D:\python项目二 数据可视化\study
plugins: anyio-4.1.0
collected 1 item

17\test_plo17.py .                                                                           [100%]

========================================= 1 passed in 4.60s ===========================================
```

结论

通过数据可视化项目，我使用 `Plotly` 模拟投掷骰子，并实现 `CSV` 文件格式、绘制天气数据的折线图、制作全球地震散点图、使用 `Web API` 获取 `Github` 的数据和使用 `Plotly` 可视化仓库等功能。此外，我还实现练习15-7的同时投掷三个骰子、练习16-3的对旧金山、死亡谷和锡斯卡的天气数据进行研究比较以及练习17-1的其他语言将七门编程语言的受欢迎仓库数据可视化。最后，我对 `Die` 类、`get_repos_info` 函数和 `get_response_dict` 函数进行测试，并将材料提交至远程仓库 `Github`。

通过实现使用 `Plotly` 模拟投掷骰子的功能，帮助用户更好地理解概率和随机事件，并且在游戏和赌博等领域有一定的应用。通过实现绘制天气数据的折线图和制作全球地震散点图的功能，帮助用户更好地了解天气和地震等自然现象，并对其进行可视化分析。通过实现使用 `Web API` 获取 `Github` 的数据和使用 `Plotly` 可视化仓库的功能，帮助用户更好地管理代码仓库，并且能够对代码质量和开发进度进行有效的监控和评估。此外，练习15-7、练习16-3和练习17-1帮助我更好地掌握Python编程知识，并提高对数据分析和可视化处理的能力。

但是，在本次项目，我实现的可视化种类较少，没有将 `Matplotlib` 等库的功能充分利用，挖掘程度不深，范围不宽。在 `API` 接口调用方面，也有爬虫知识的拓展和要求。

总而言之，本次项目让我收获巨大，我需要学习了解的地方还有很多。

参考文献

[1] [美]Eric Matthes.Python编程：从入门到实践[M].袁国忠,译.北京:人民邮电出版社,2023:273-341.