《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 | 龙志强 |
| 学号 | B20190304321 |
| 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 | 龙志强 |
| 答题人班级 | 19软件03班 |
| 答题人学号 | B20190304321 |
| 互评分数 | |
| 总分 | 150 |
| 评语 | 功能按实验要求完成，代码用git版本控制，语法规范，完成了附加题，编写了单元测试。 |
| 互评人信息 | |
| 互评人姓名 | 康泽 |
| 互评人学号 | B20190304316 |
| 互评人班级 | 19软件03班 |

## 第1章 需求分析

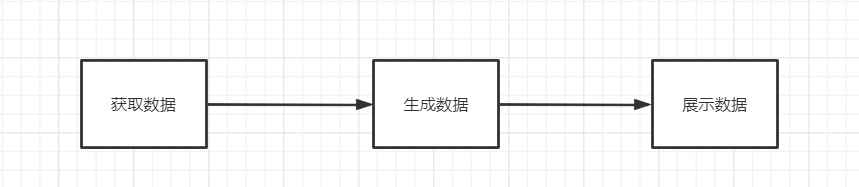
本系统需要实现了对数据的可视化操作，包括制作折线图、散点图以及直方图等等，需要通过用户所提供的数据可以制作相应的可视化图形，从而解决用户的需求，主要的需求就是能够绘制模拟掷骰子、天气数据、全球地震散点图以及使用Web API获取数据并绘制相应的图形。

# 第2章 设计与实现

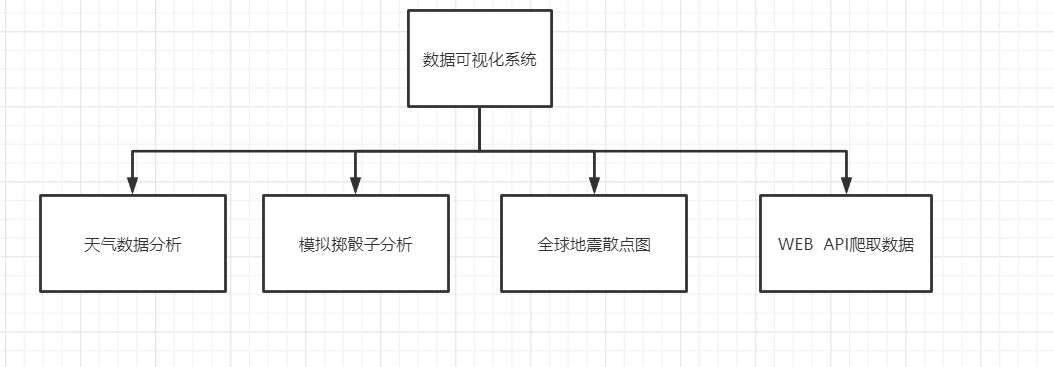
**系统架构：**

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：**



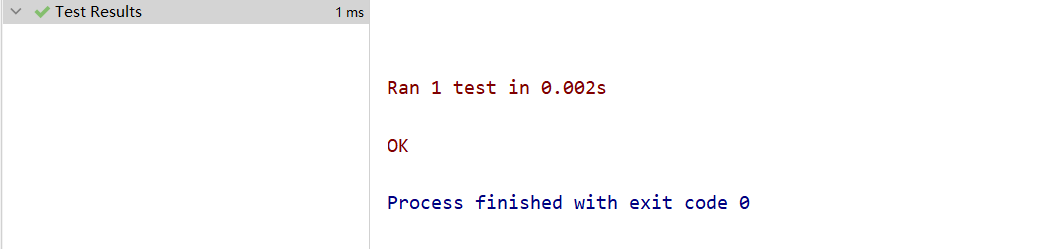
**系统模块：**



# 第3章 单元测试

* 对随机生成骰子的测试：

import unittest  
  
from rolling\_dice.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()



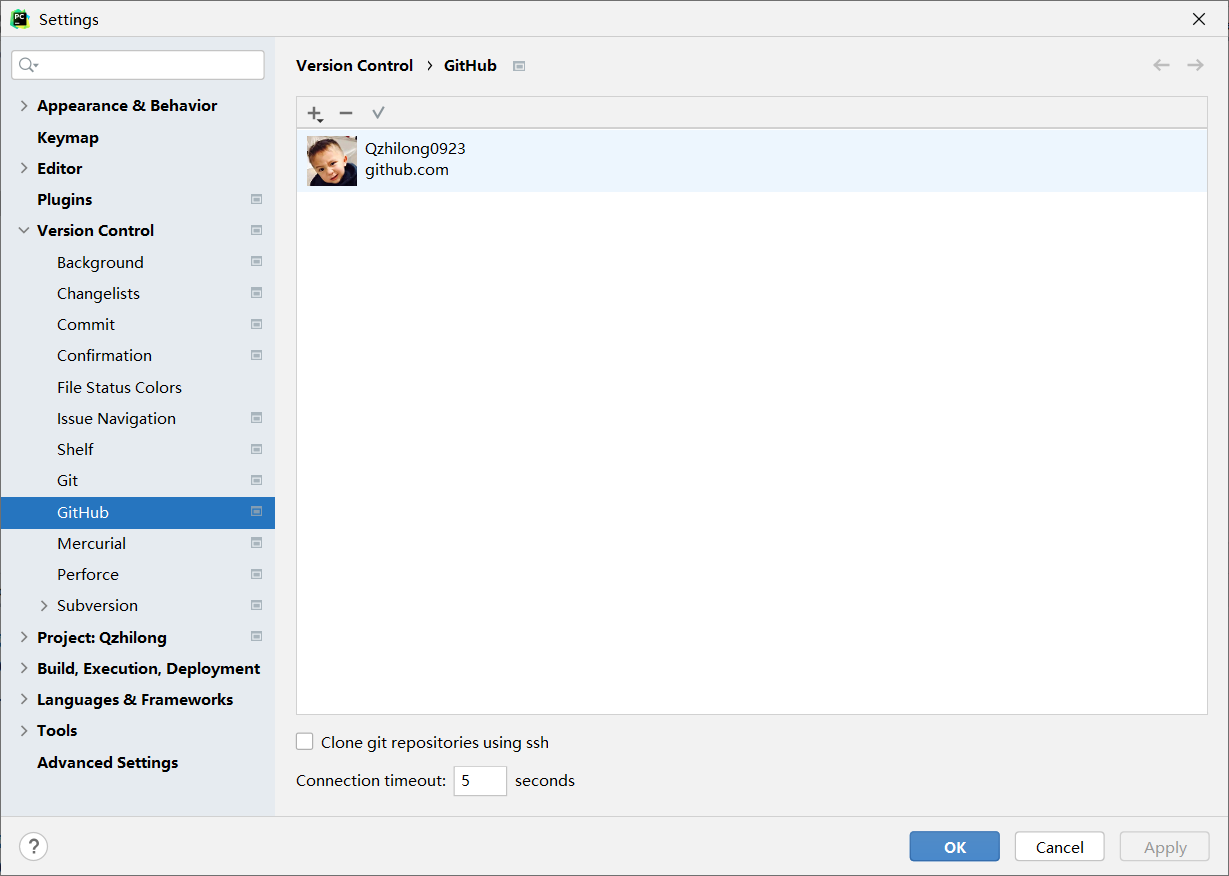
## 第4章 Git版本控制管理

1. 安装git
2. 配置git

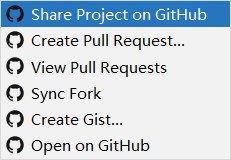
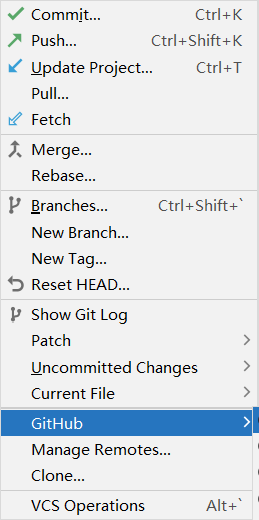
git config -global user.name "longzq"

git config -global user.email ["1299166485@qq.com"](mailto:\"1299166485@qq.com\")

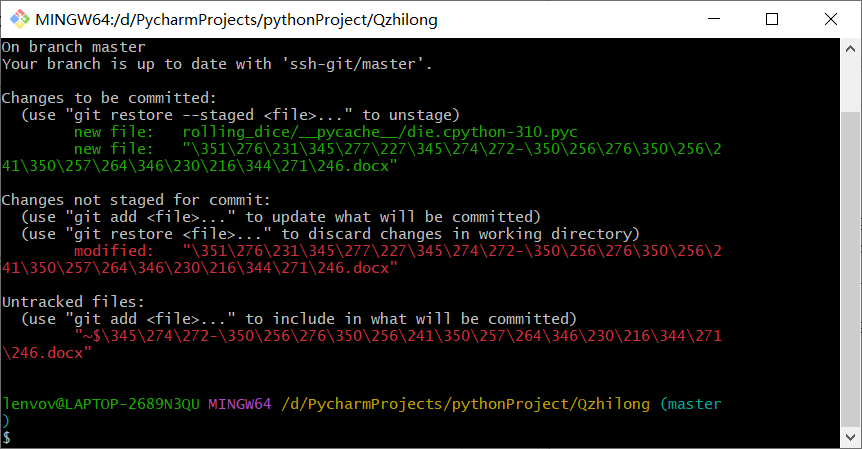
1. IDEA集成GitHub



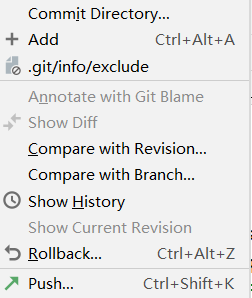
1. 初始化仓库



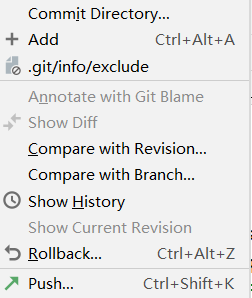
1. 检查状态



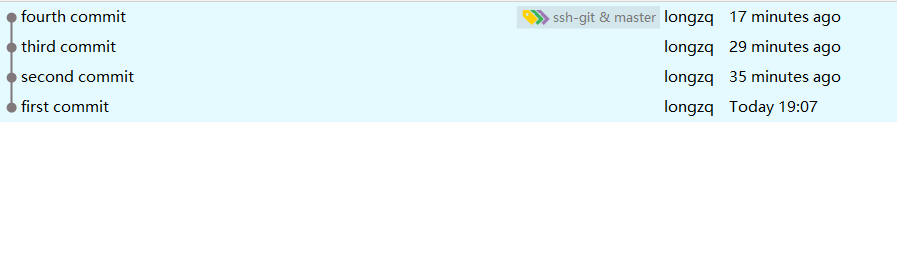
1. 添加文件



1. 提交



1. 查看提交历史



# 总 结

本次实训的内容是对数据的可视化，主要学会了如何能够绘制模拟掷骰子、CSV文件格式，绘制天气数据的折线图，、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：

* **掷骰子：**

die.py

from random import randint  
  
class Die:  
 *"""A class representing a single die."""* def \_\_init\_\_(self, num\_sides=6):  
 *"""Assume a six-sided die."""* self.num\_sides = num\_sides  
   
 def roll(self):  
 *""""Return a random value between 1 and number of sides."""* return randint(1, self.num\_sides)

die\_visual.py

from plotly.graph\_objs import Bar, Layout  
from plotly import offline  
  
from die import Die  
  
*# Create a D6.*die = Die()  
  
*# Make some rolls, and store results in a list.*results = []  
for roll\_num in range(1000):  
 result = die.roll()  
 results.append(result)  
   
*# Analyze the results.*frequencies = []  
for value in range(1, die.num\_sides+1):  
 frequency = results.count(value)  
 frequencies.append(frequency)  
   
*# Visualize the results.*x\_values = list(range(1, die.num\_sides+1))  
data = [Bar(x=x\_values, y=frequencies)]  
  
x\_axis\_config = {'title': 'Result'}  
y\_axis\_config = {'title': 'Frequency of Result'}  
my\_layout = Layout(title='Results of rolling one D6 1000 times',  
 xaxis=x\_axis\_config, yaxis=y\_axis\_config)  
offline.plot({'data': data, 'layout': my\_layout}, filename='d6.html')

dice\_visual.py

from plotly.graph\_objs import Bar, Layout  
from plotly import offline  
  
from die import Die  
  
*# Create two D6 dice.*die\_1 = Die()  
die\_2 = Die()  
  
*# Make some rolls, and store results in a list.*results = []  
for roll\_num in range(1000):  
 result = die\_1.roll() + die\_2.roll()  
 results.append(result)  
   
*# Analyze the results.*frequencies = []  
max\_result = die\_1.num\_sides + die\_2.num\_sides  
for value in range(2, max\_result+1):  
 frequency = results.count(value)  
 frequencies.append(frequency)  
   
*# Visualize the results.*x\_values = list(range(2, max\_result+1))  
data = [Bar(x=x\_values, y=frequencies)]  
  
x\_axis\_config = {'title': 'Result', 'dtick': 1}  
y\_axis\_config = {'title': 'Frequency of Result'}  
my\_layout = Layout(title='Results of rolling two D6 dice 1000 times',  
 xaxis=x\_axis\_config, yaxis=y\_axis\_config)  
offline.plot({'data': data, 'layout': my\_layout}, filename='d6\_d6.html')

**dice\_visual\_different\_sizes.py**

from plotly.graph\_objs import Bar, Layout  
from plotly import offline  
  
from die import Die  
  
*# Create a D6 and a D10.*die\_1 = Die()  
die\_2 = Die(10)  
  
*# Make some rolls, and store results in a list.*results = []  
for roll\_num in range(50\_000):  
 result = die\_1.roll() + die\_2.roll()  
 results.append(result)  
   
*# Analyze the results.*frequencies = []  
max\_result = die\_1.num\_sides + die\_2.num\_sides  
for value in range(2, max\_result+1):  
 frequency = results.count(value)  
 frequencies.append(frequency)  
   
*# Visualize the results.*x\_values = list(range(2, max\_result+1))  
data = [Bar(x=x\_values, y=frequencies)]  
  
x\_axis\_config = {'title': 'Result', 'dtick': 1}  
y\_axis\_config = {'title': 'Frequency of Result'}  
my\_layout = Layout(title='Results of rolling a D6 and a D10 50000 times',  
 xaxis=x\_axis\_config, yaxis=y\_axis\_config)  
offline.plot({'data': data, 'layout': my\_layout}, filename='d6\_d10.html')

**dice\_visual\_three.py**

from plotly import offline  
from die import Die  
from plotly.graph\_objs import Bar, Layout  
*# 创建三个 D6 骰子。*die\_1 = Die()  
die\_2 = Die()  
die\_3 = Die()  
  
*# 掷骰子多次，并将结果存储在一个列表中。*results = [die\_1.roll() + die\_2.roll() + die\_3.roll() for roll\_num in range(1\_000\_000)]  
  
*# 分析结果。*max\_result = die\_1.num\_sides + die\_2.num\_sides + die\_3.num\_sides  
frequencies = [results.count(value) for value in range(3, max\_result+1)]  
  
*# 可视化结果。*x\_values = list(range(3, max\_result+1))  
data = [Bar(x=x\_values, y=frequencies)]  
x\_axis\_config = {'title': 'Result', 'dtick': 1}  
y\_axis\_config = {'title': 'Frequency of Result'}  
my\_layout = Layout(title='Results of rolling three D6 dice 1,000,000 times',  
xaxis=x\_axis\_config, yaxis=y\_axis\_config)  
offline.plot({'data': data, 'layout': my\_layout}, filename='3d6.html')

* **CSV文件格式，绘制天气数据的折线图:**

**sitka\_highs.py**

import csv  
from datetime import datetime  
  
from matplotlib import pyplot as plt  
  
filename = 'data/sitka\_weather\_2018\_simple.csv'  
with open(filename) as f:  
 reader = csv.reader(f)  
 header\_row = next(reader)  
  
 *# Get dates and high temperatures from this file.* dates, highs = [], []  
 for row in reader:  
 current\_date = datetime.strptime(row[2], '%Y-%m-%d')  
 dates.append(current\_date)  
 high = int(row[5])  
 highs.append(high)  
  
*# Plot the high temperatures.*plt.style.use('seaborn')  
fig, ax = plt.subplots()  
ax.plot(dates, highs, c='red')  
  
*# Format plot.*plt.title("Daily high temperatures - 2018", fontsize=24)  
plt.xlabel('', fontsize=16)  
fig.autofmt\_xdate()  
plt.ylabel("Temperature (F)", fontsize=16)  
plt.tick\_params(axis='both', which='major', labelsize=16)  
  
plt.show()

sitka\_highs\_lows.py

import csv  
from datetime import datetime  
  
from matplotlib import pyplot as plt  
  
filename = 'data/sitka\_weather\_2018\_simple.csv'  
with open(filename) as f:  
 reader = csv.reader(f)  
 header\_row = next(reader)  
  
 *# Get dates, and high and low temperatures from this file.* dates, highs, lows = [], [], []  
 for row in reader:  
 current\_date = datetime.strptime(row[2], '%Y-%m-%d')  
 high = int(row[5])  
 low = int(row[6])  
 dates.append(current\_date)  
 highs.append(high)  
 lows.append(low)  
  
*# Plot the high and low temperatures.*plt.style.use('seaborn')  
fig, ax = plt.subplots()  
ax.plot(dates, highs, c='red', alpha=0.5)  
ax.plot(dates, lows, c='blue', alpha=0.5)  
plt.fill\_between(dates, highs, lows, facecolor='blue', alpha=0.1)  
  
*# Format plot.*plt.title("Daily high and low temperatures - 2018", fontsize=24)  
plt.xlabel('', fontsize=16)  
fig.autofmt\_xdate()  
plt.ylabel("Temperature (F)", fontsize=16)  
plt.tick\_params(axis='both', which='major', labelsize=16)  
  
plt.show()

death\_valley\_highs\_lows.py

import csv  
from datetime import datetime  
  
from matplotlib import pyplot as plt  
  
filename = 'data/death\_valley\_2018\_simple.csv'  
with open(filename) as f:  
 reader = csv.reader(f)  
 header\_row = next(reader)  
  
 *# Get dates, and high and low temperatures from this file.* dates, highs, 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:  
 dates.append(current\_date)  
 highs.append(high)  
 lows.append(low)  
  
*# Plot the high and low temperatures.*plt.style.use('seaborn')  
fig, ax = plt.subplots()  
ax.plot(dates, highs, c='red', alpha=0.5)  
ax.plot(dates, lows, c='blue', alpha=0.5)  
plt.fill\_between(dates, highs, lows, facecolor='blue', alpha=0.1)  
  
*# Format plot.*title = "Daily high and low temperatures - 2018\nDeath Valley, CA"  
plt.title(title, fontsize=20)  
plt.xlabel('', fontsize=16)  
fig.autofmt\_xdate()  
plt.ylabel("Temperature (F)", fontsize=16)  
plt.tick\_params(axis='both', which='major', labelsize=16)  
  
plt.show()

seattle\_highs\_lows.py

import pandas as pd  
from matplotlib import pyplot as plt  
  
filename = 'data/2016-weather-data-seattle.csv'  
*# 读文件*data = pd.read\_csv(filename,parse\_dates = ['Date'])  
  
*# 传入数据*dates = data["Date"]  
highs = data["Max\_TemperatureC"]  
lows = data["Min\_TemperatureC"]  
  
*# Plot the high and low temperatures.*plt.style.use('seaborn')  
fig, ax = plt.subplots()  
ax.plot(dates, highs, c='red', alpha=0.5)  
ax.plot(dates, lows, c='blue', alpha=0.5)  
plt.fill\_between(dates, highs, lows, facecolor='blue', alpha=0.1)  
  
*# Format plot.*plt.title("2015 seattle high and low temperatures - 2018", fontsize=24)  
plt.xlabel('', fontsize=16)  
fig.autofmt\_xdate()  
plt.ylabel("Temperature (F)", fontsize=16)  
plt.tick\_params(axis='both', which='major', labelsize=16)  
  
plt.show()

* **制作全球地震散点图:**

**eq\_plots.py**

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)  
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]  
 mags.append(mag)  
 titles.append(title)  
 lons.append(lon)  
 lats.append(lat)  
  
data = pd.DataFrame(  
 data=zip(lons, lats, titles, mags), columns=['经度', '纬度', '位置', '震级']  
)  
  
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.html')  
fig.show()

**eq\_explore\_data.py**

import json  
  
filename = 'data/eq\_data\_30\_day\_m1.json'  
with open(filename) as f:  
 all\_eq\_data = json.load(f)  
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]  
 mags.append(mag)  
 titles.append(title)  
 lons.append(lon)  
 lats.append(lat)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(mags[:5])  
 print(titles[:5])  
 print(lons[:5])  
 print(lats[:5])

**eq\_world\_map.py**

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)  
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]  
 mags.append(mag)  
 titles.append(title)  
 lons.append(lon)  
 lats.append(lat)  
  
data = pd.DataFrame(  
 data=zip(lons, lats, titles, mags), columns=['经度', '纬度', '位置', '震级']  
)  
  
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.html')  
fig.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(mags[:5])  
 print(title[:5])  
 print(lons[:5])  
 print(lats[:5])

* **WebAPI:**

**python\_repos.py**

import requests  
  
*# Make an API call and store the response.*url = 'https://api.github.com/search/repositories?q=language:python&sort=stars'  
headers = {'Accept': 'application/vnd.github.v3+json'}  
r = requests.get(url, headers=headers)  
print(f"Status code: {r.status\_code}")  
  
*# Store API response in a variable.*response\_dict = r.json()  
print(f"Total repositories: {response\_dict['total\_count']}")  
  
*# Explore information about the repositories.*repo\_dicts = response\_dict['items']  
print(f"Repositories returned: {len(repo\_dicts)}")  
  
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']}")

**python\_repos\_visual.py**

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:python&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': 'Most-Starred Python Projects on GitHub',  
 'titlefont': {'size': 28},  
 'xaxis': {  
 'title': 'Repository',  
 'titlefont': {'size': 24},  
 'tickfont': {'size': 14},  
 },  
 'yaxis': {  
 'title': 'Stars',  
 'titlefont': {'size': 24},  
 'tickfont': {'size': 14},  
 },  
  
}  
  
fig = {'data': data, 'layout': my\_layout}  
offline.plot(fig, filename='python\_repos.html')

### 运行结果图：

