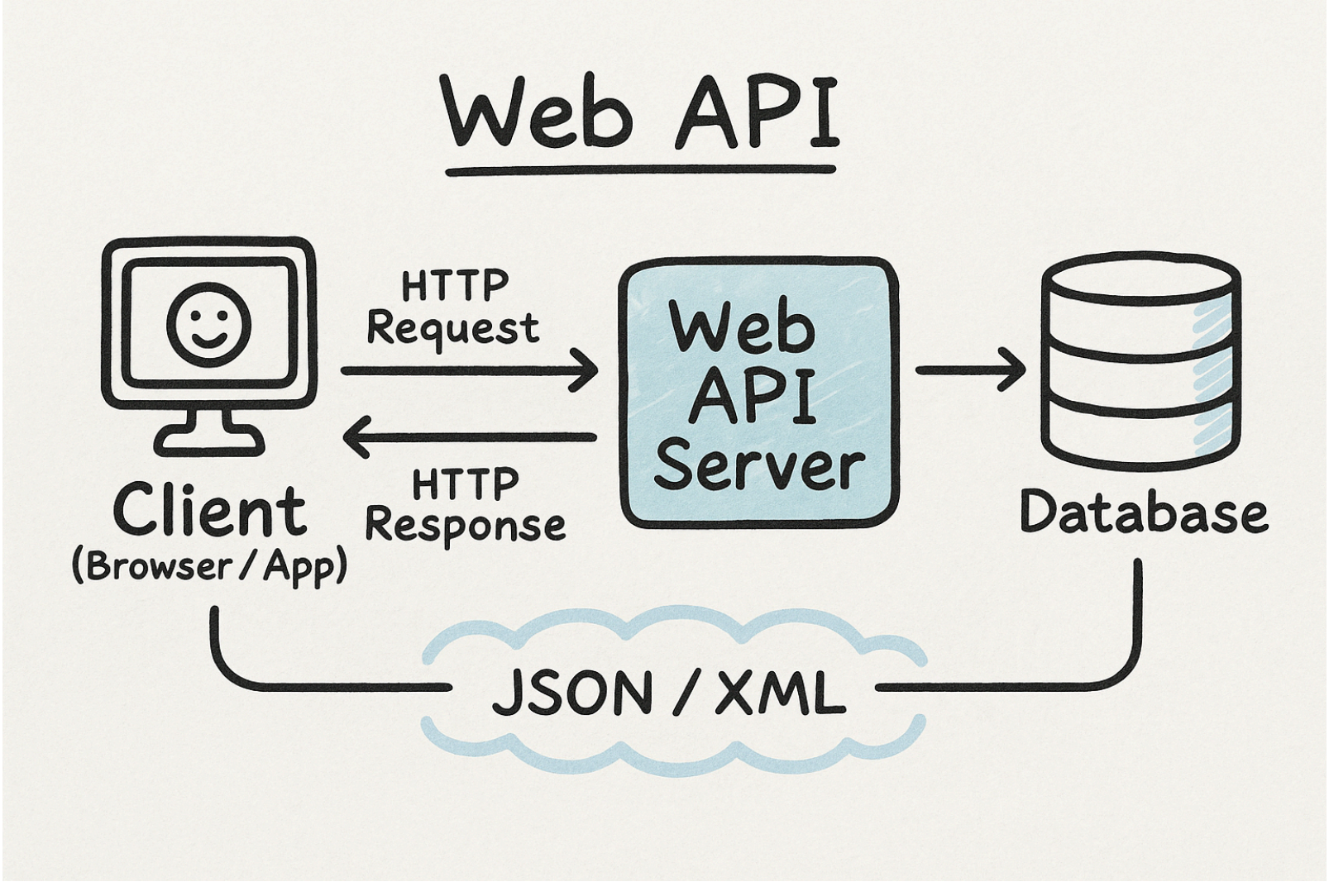
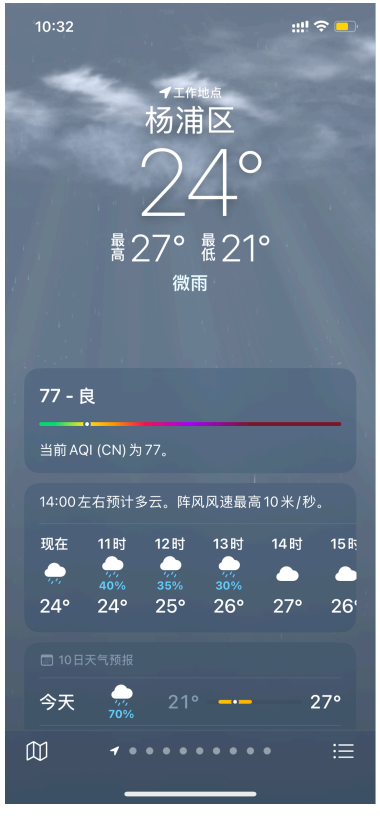
Python Programming

Lecture 13 Web API

13.1 Web Api


Web API
HTTP Request
HTTP Response
Web API Server
Client
(Browser / App)
Database
JSON / XML

Weather Data



中国气象台大数据接口

```python
import requests

url = "http://t.weather.itboy.net/api/weather/city/101020100"
r = requests.get(url)
print(r.status_code)
response_dict = r.json()
```

```python
f = response_dict['data']
ff = f['forecast']
ff_today = ff[0]
ff_1 = ff[1]
ff_2 = ff[2]

def show(day):
    for x in day:
        print(x+': '+str(day[x]))
    print()
show(ff_today)
show(ff_1)
show(ff_2)
```

Deepseek API开放平台

API说明文档

```python
# Please install OpenAI SDK first: `pip3 install openai`
from openai import OpenAI

client = OpenAI(api_key="your api_key", base_url="https://api.deepseek.com")

response = client.chat.completions.create(
    model="deepseek-chat",
    messages=[
        {"role": "system", "content": "You are a helpful assistant"},
        {"role": "user", "content": "鲁迅暴打周树人"},
    ],
    stream=False
)

print(response.choices[0].message.content)
```

Where to find Web Api? Public APIs, 聚合数据, IMDB-API

TMDB-API, OMDB-API

```
import requests

api_access = 'Your API Key'
```

```
page = 1
url = f"https://api.tmdb.org/3/movie/\
top_rated?language=en-US&page={page}"

headers = {
    "accept": "application/json",
    "Authorization": f"Bearer {api_access}"
}
response = requests.get(url, headers=headers)
response_dict = response.json()
# print(response_dict)
```

```
movies=response_dict["results"]
print(len(movies))

for key, value in movies[0].items():
    print(f"{key}: {value}")
```

```
adult: False
backdrop_path: /zfbjgQE1uSd9wiPTX4VzsLi0rGG.jpg
genre_ids: [18, 80]
id: 278
original_language: en
original_title: The Shawshank Redemption
overview: Imprisoned in the 1940s for the double murder of his wife and her lover,
upstanding banker Andy Dufresne begins a new life at the Shawshank prison,
where he puts his accounting skills to work for an amoral warden.
During his long stretch in prison, Dufresne comes to be admired by the other
inmates -- including an older prisoner named Red --
for his integrity and unquenchable sense of hope.
popularity: 115.576
poster_path: /9cqNxx0GxF0bflZmeSMuL5tnGzr.jpg
release_date: 1994-09-23
title: The Shawshank Redemption
video: False
vote_average: 8.705
vote_count: 26204
```

Downloading Images

```
from pathlib import Path

poster = movies[0]['poster_path']
title = movies[0]['title']
img_url = f"https://image.tmdb.org/t/p/w500{poster}"
r = requests.get(img_url, headers=headers)

if r.status_code == 200:
    save_path = Path(f"{title}.jpg")
    save_path.write_bytes(r.content)
else:
    print("download failed")
```

Top10

```
import requests
from pathlib import Path
api_access = 'Your API Key'
```

```
page = 1
url = f"https://api.tmdb.org/3/movie/\
top_rated?language=en-US&page={page}"
headers = {
    "accept": "application/json",
    "Authorization": f"Bearer {api_access}"
}
response = requests.get(url, headers=headers)
response_dict = response.json()

movies=response_dict["results"]
top10 = movies[:10]
```

```
for movie in top10:
    poster = movie['poster_path']
    title = movie['title']
    img_url = f"https://image.tmdb.org/t/p/w500{poster}"
    r = requests.get(img_url, headers=headers)

    if r.status_code == 200:
        save_path = Path(f"{title}.jpg")
        save_path.write_bytes(r.content)
    else:
        print("download failed")
```

Now Playing

```
import requests
from pathlib import Path
api_access = 'Your API Key'
```

```
page = 1
url = f"https://api.tmdb.org/3/movie/\
now_playing?language=en-US&page={page}"
headers = {
    "accept": "application/json",
    "Authorization": f"Bearer {api_access}"
}
response = requests.get(url, headers=headers)
response_dict = response.json()

movies=response_dict["results"]
top10 = movies[:10]
```

```
for movie in top10:
    poster = movie['poster_path']
    title = movie['title']
    img_url = f"https://image.tmdb.org/t/p/w500{poster}"
    r = requests.get(img_url, headers=headers)

    if r.status_code == 200:
        save_path = Path(f"{title}.jpg")
        save_path.write_bytes(r.content)
    else:
        print("download failed")
```

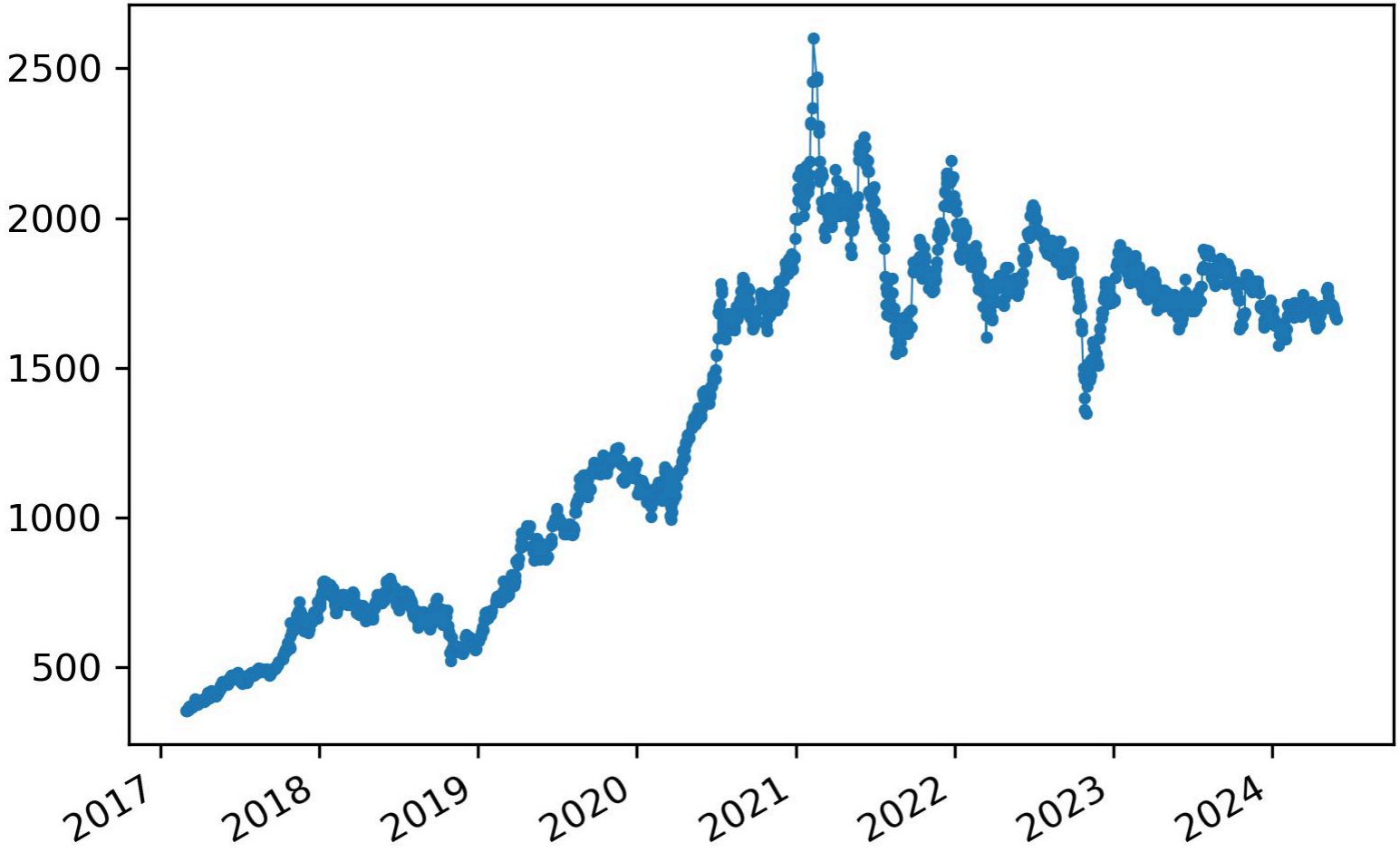
量化投资

Tushare, JoinQuant, AKshare

```python
import akshare as ak

stock_zh_a_hist_df = ak.stock_zh_a_hist(symbol="600519", period="daily",\
start_date="20170301", end_date='20240528', adjust="")
print(stock_zh_a_hist_df)
```

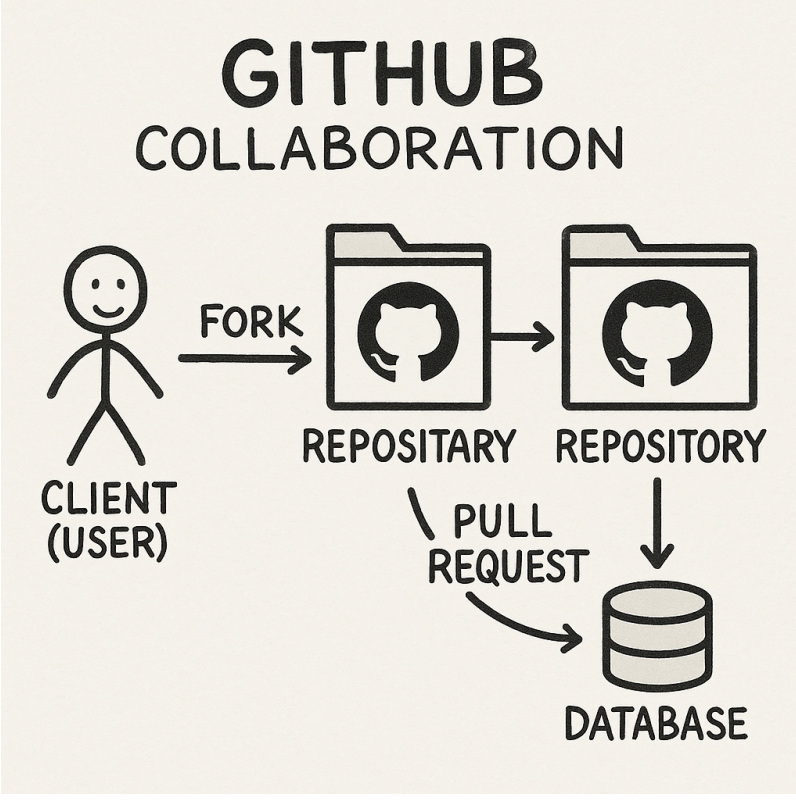
- symbol为股票代码，adjust为是否复权
- 生成的格式为pandas包中的DataFrame格式
- “贵州茅台”收盘价绘图（2017-2024）

```python
import matplotlib.pyplot as plt
from datetime import datetime

fig, ax = plt.subplots()
ax.plot(stock_zh_a_hist_df["日期"],stock_zh_a_hist_df["收盘"], linewidth=0.5)
ax.scatter(stock_zh_a_hist_df["日期"],stock_zh_a_hist_df["收盘"], s=5)
fig.autofmt_xdate()
plt.savefig('maotai.jpg',dpi=300)
```


2500
2000
1500
1000
500
2017
2018
2019
2020
2021
2022
2023
2024

13.2 Github & AI辅助编程

Github教程



GitHub 是一个基于 Git 版本控制系统 的 代码托管平台，由 GitHub 公司（现为 Microsoft 子公司）于 2008 年推出。它是全球最受欢迎的开源协作平台，广泛应用于软件开发、版本管理、团队协作和项目发布。GitHub官网地址：https://github.com

👩‍💻 适用人群

- 开发者：托管代码、开源项目、构建团队协作流程
- 学生：学习编程、参与开源项目、积累实践经验
- 教师：共享教学代码、布置项目作业
- 企业/组织：进行私有项目开发、部署自动化流程

Github的API接口（关于Python的项目）

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

```python
import requests

url = 'https://api.github.com/search/\
repositories?q=language:python&sort=stars'
r = requests.get(url)
print("Status code:", r.status_code)
response_dict = r.json()

for keys in response_dict.keys():
    print(keys)
```

```
Status code: 200
total_count
incomplete_results
items
```

- Working with the Response Dictionary

```
print("Total repositories:", response_dict['total_count'])

repo_dicts = response_dict['items']
print("Repositories returned:", len(repo_dicts))

repo_dict = repo_dicts[0]
print("\nKeys:", len(repo_dict))
```

```
Total repositories: 20566130
Repositories returned: 30

Keys: 80
```

```python
print("\nSelected information about first repository:")
print('Name:', repo_dict['name'])
print('Owner:', repo_dict['owner']['login'])
print('Stars:', repo_dict['stargazers_count'])
print('Repository:', repo_dict['html_url'])
print('Created:', repo_dict['created_at'])
print('Updated:', repo_dict['updated_at'])
print('Description:', repo_dict['description'])
```

```
Selected information about first repository:
Name: public-apis
Owner: public-apis
Stars: 339656
Repository: https://github.com/public-apis/public-apis
Created: 2016-03-20T23:49:42Z
Updated: 2025-05-16T06:51:23Z
Description: A collective list of free APIs
```

- Visualizing Repositories Using Plotly

```
import requests
import plotly.express as px

URL = 'https://api.github.com/search/repositories?q=language:python&sort=star'
r = requests.get(URL)
print("Status code:", r.status_code)
response_dict = r.json()
```

```
repo_dicts = response_dict['items']
names, stars = [], []
for repo_dict in repo_dicts:
    names.append(repo_dict['name'])
    stars.append(repo_dict['stargazers_count'])
```

```
title = "Most-Starred Python Projects on GitHub"
labels = {'x': 'Repository', 'y': 'Stars'}
fig = px.bar(x = names, y=stars, title = title, labels=labels)
fig.write_html('python_repos.html')
```

Most-Starred Python Projects

GitHub Copilot官网

尽管 ChatGPT或者Deepseek 可以编写完整的代码，但是要与集成开发环境（Integrated DevelopmentEnvironment,IDE）无缝对按，使用ChatGPT 就不太方便了，尤其是在生成片段代码时，如补全一个函数的定义、补全某个语句等，在这种情况下，使用GitHub Copilot 是一个非常好的选择。当然，最好是将 ChatGPT 与 GitHub Copilot一起使用：使用 ChatGPT 生成一个完整的解决方案，并使用 GitHub Copilot 对这个解决方案进行微调。

1. 安装VScode参考配置 参考配置（VScode是什么？常用IDE有哪些？）

2. 注册GitHub账户；3. 在VScode里面安装GitHub Copilot插件

功能介绍

1. 自动补全注释（按Tab键）

```
# 编写一个程序，读取文件夹中的文件
```

2. 根据函数名自动生成代码（按Tab键）

```
def bubblesort()
```

3. 生成测试用例（在bubblesort函数下方输入）

```
#测试bubblesort函数
```

4. 逐步代码生成

```
#定义5个列表变量，每个列表包含2到10个元素
#将5个列表合并，再调用bubblesort函数对列表排序
```

5. 自动生成语句架构

```
for i
```

```
-if
```

6. 生成多个候选解决方案

如果使用Tab键生成解决方案，那么对于一些复杂的代码，可能需要一行一行地生成（需 要不断地按 Enter 键和Tab 键），比较麻烦。GitHub Copilot 提供了生成多个候选解决方案的功能，具体的做法就是在注释中按 Ctrl + Enter 组合键，这将显示一个新的选项卡，默认会自动生成10个解决方案。(有时候还不如直接ChatGPT或者Deepseek)

```
#用Flask实现一个服务端程序，只支持GET请求，请求的参数是一个字符串，返回一个字符串。
```

7. 检查代码漏洞

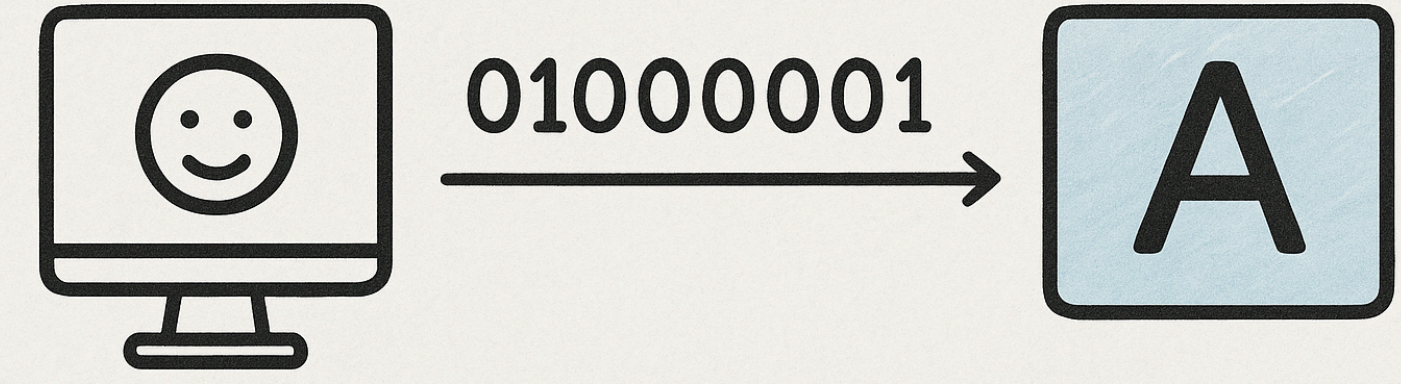
```
# 检查上面的代码是否有漏洞
```

一句话就是：写注释，然后enter和tab来回的按。

GitHub Copilot的免费平替CodeGeeX

13.3 Encoding（计算机编码）

Computer Encoding



Encoding

```python
from pathlib import Path

path = Path('alice.txt')
contents = path.read_text(encoding='utf-8')
```

Character Encoding: ASCII, Unicode, UTF-8, GBK

ASCII 字符代码表 一

高四位 / 低四位		ASCII非打印控制字符										ASCII 打印字符													
		0000					0001					0010		0011		0100		0101		0110		0111			
		0					1					2		3		4		5		6		7			
		+进制	字符	ctrl	代码	字符解释	+进制	字符	ctrl	代码	字符解释	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	ctrl
0000	0	0	BLANK NULL	^@	NUL	空	16	▶	^P	DLE	数据链路转意	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH	头标开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☻	^B	STX	正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX	正文结束	19	!!	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	◆	^D	EOT	传输结束	20	¶	^T	DC4	设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ	查询	21	§	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK	确认	22	■	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	震铃	23	↨	^W	ETB	传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	◘	^H	BS	退格	24	↑	^X	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	TAB	水平制表符	25	↓	^Y	EM	媒体结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◙	^J	LF	换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	竖直制表符	27	←	^[	ESC	转意	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	换页/新页	28	∟	^\	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|	
1101	D	13	♪	^M	CR	回车	29	↔	^]	GS	组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	♫	^N	SO	移出	30	▲	^6	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI	移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ	^Back space

注：表中的ASCII字符可以用:ALT + "小键盘上的数字键" 输入

- Unicode把所有语言都统一到一套编码里。Unicode标准也在不断发展，但最常用的是用两个字节表示一个字符（如果要用到非常偏僻的字符，就需要4个字节）。现代操作系统和大多数编程语言都直接支持Unicode。
- UTF-8编码把一个Unicode字符根据不同的数字大小编码成1-6个字节，常用的英文字母被编码成1个字节，汉字通常是3个字节，只有很生僻的字符才会被编码成4-6个字节。
- 在计算机内存中，统一使用Unicode编码，当需要保存到硬盘或者需要传输的时候，就转换为UTF-8编码。

```
chinese = '你好'.encode('utf-8')
print(chinese)      # 输出：b'\xe4\xbd\xa0\xe5\xa5\xbd'
#这是字节串（是bytes，不是字符串！）用16进制是为了方便阅读，\x是16进制的前缀
print(len(chinese))  # 输出 6（每个汉字占 3 字节）
```

```
b'\xe4\xbd\xa0\xe5\xa5\xbd'.decode('utf-8') # 输出：你好
```

- Two-dimensional code, QR code

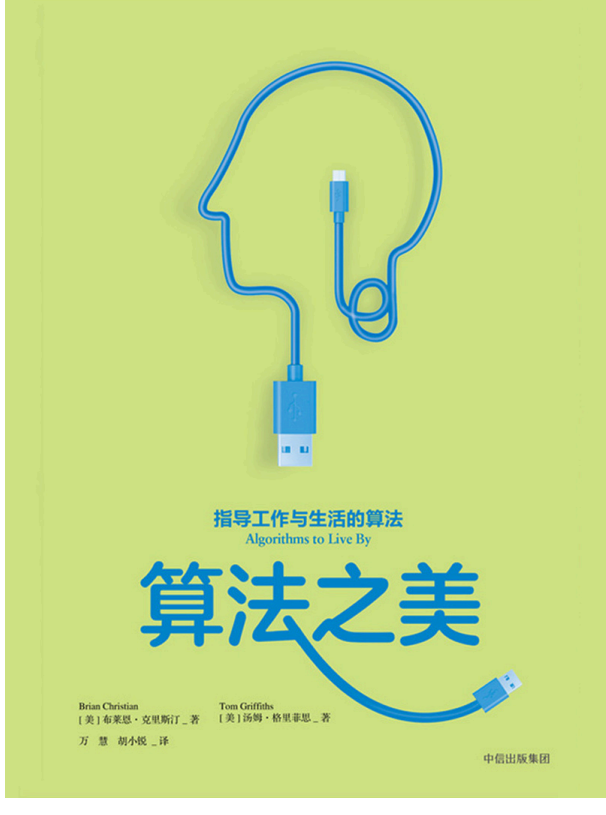
```python
import qrcode

img=qrcode.make("Hello!")
img.save("x.png")
```

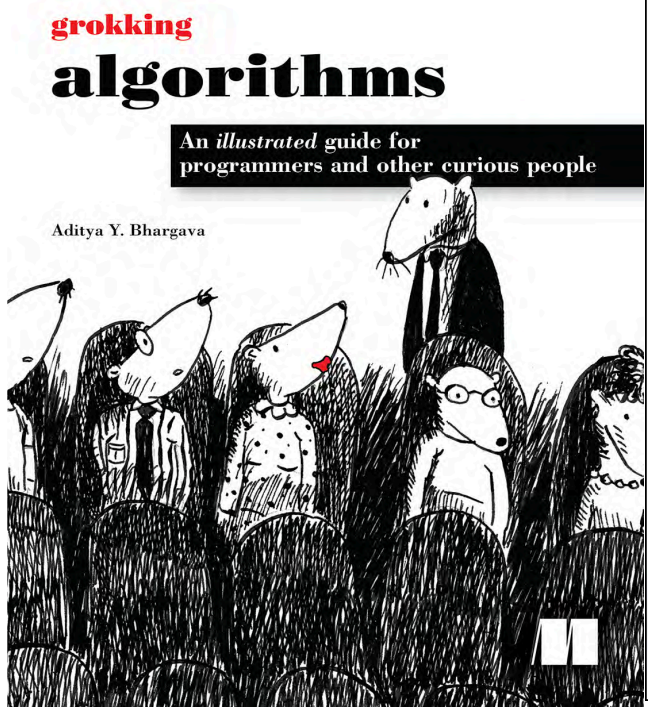
```python
import qrcode

img=qrcode.make("https://wangwanglulu.com/")
img.save("wl.png")
```

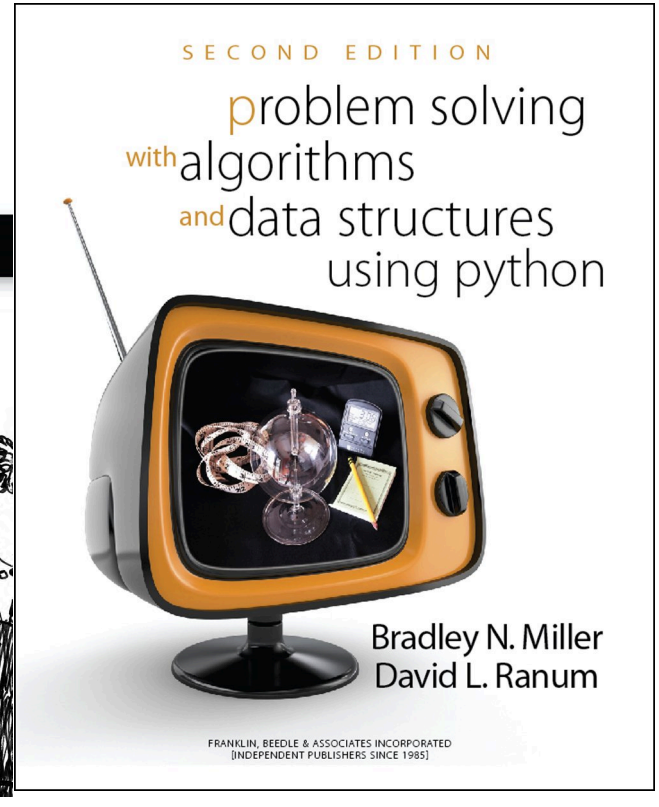
教材中我们跳过的章节：

- Python Crash Couse (Chapter 12 - 14, 18 - 20)
  - Chapter 11: Testing Your Code（测试代码）
  - Chapter 12 -14: Alien Invasion（外星人入侵）
  - Chapter 18 - 20: Django（Web应用：创建网站）
- Python for Everybody (Chapter 11 - 13, 15 - 16)
  - Chapter 11: Regular Expressions（正则表达式）
  - Chapter 12: Networked Programs 12.4 - 12.8 (urlib, BeautifulSoup)（分析网页）
  - Chapter 13: Using Web Services (XML, JSON, API)（还是web api）
  - Chapter 15: Databases and SQL（数据库）
  - Chapter 16: Visualizing data (Network, Word Cloud)（可视化：网络图，词云）

What's next?


指导工作与生活的算法
Algorithms to Live By
算法之美
Brian Christian
[美] 布莱恩·克里斯汀 _ 著
Tom Griffiths
[美] 汤姆·格里菲思 _ 著
万 慧 胡小锐 _ 译
中信出版集团


grokking
algorithms
An illustrated guide for programmers and other curious people
Aditya Y. Bhargava


SECOND EDITION
problem solving
with algorithms
and data structures
using python
Bradley N. Miller
David L. Ranum
FRANKLIN, BEEDLE & ASSOCIATES INCORPORATED
[INDEPENDENT PUBLISHERS SINCE 1985]

Summary

- Web Api
  - Reading: Python Crash Course, Chapter 17