**2022**

**2/11/22**

1. **Prepare interview stories and add to resume.**
2. **Get to interview ready by 2/28**
3. **Check progress once every week.; back on 2/20**

**1/17**/22

1. Finish all the old stuff by 1/21/22
2. Prepare new era, starting 1/24
3. Targeting 2/1 starting interview process.

**9/14/21**

* API stories
* Master sheets:
* Python
* Selenium
* API request
* Pytest
* CI/CD Jenkins
* Linux script/command
* Private cloud vs public cloud

**9/1/21, 9/6, 9/7**

**尽快把自己已掌握的分类，巩固**

* Json parse and how to generate json or excel/cvs test data.
* API test 如何处理抛出异常
* 集中在，文件传测试参数
* Fixture 处理（what to setup and what to teardown）
* pytest测试用例编写
* pytest重点就是fixture传参，数据和script 分离。

**8/26/2021**

* Selenium automation how to handle exceptions
* Excel. File processing: xlrd
* Selenium POM and framework-pytest
* 测试数据初始化scripting
* 如何提高 selenium 脚本的自动化执行效率
* Pytest+Jenkins+git+logging+report Framework
* Cross-browser testing
* Multi-page continuous ops
* Pytest, assert , how to handle unexpected exception so following tests will still continue without being impacted.

**8/23/21**

* Focused on Linux
* Jenkins under Linux
* Database brush through
* API, UI pickup
* CDN应用和原理

**目的是， 赶紧面试进入READY**

**8/17/2021**

1. Logging
2. Learn HTML: login
3. ActionChains()

**8/4/21**

7/7/21

**Solve problems**

**Write stories**

**The aim is to get to a stage: ready for interview anytime!**

1. UI and API
2. What have done In General!
3. Give specific examples what have been done!

* **API test / UI test project for interview story: specific work**
* **And general daily routine work**

**To review:**

* **Database interview**
* **Concept: categorize and memorize.**
* **Solid examples to back up concept**

**6/18/2021, 6/22,6/24**

<https://www.cnblogs.com/>

<https://blog.csdn.net/>

* **API: requests, token/security/performance/debug**
* Regular expression: re 🡪 Next step: example studies, done(6/25)
* Fiddler, pokemon
* 谷歌浏览器的开发者者工具分析
* Python json parse
* **Page object model: focus**
* **Linux commando**
* Jenkins commando
* Allure
* **Javascript study**
* **Map,zip, filter**
* **Web test with database errors**
* **前端和后台的接口文件**

# 在网页上点击一个按钮到服务器的整个流程

# 接口测试面试题

* [**https://blog.csdn.net/dubinglin/article/details/78553242?utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-2.control&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-2.control**](https://blog.csdn.net/dubinglin/article/details/78553242?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-2.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-2.control)

# 接口测试点及常见bug

https://blog.csdn.net/fantian\_/article/details/113943753?utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-3.control&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-3.control

# 接口自动化测试面试题（1）

* [**https://blog.csdn.net/candy\_tse\_1/article/details/99452378?utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-4.control&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-4.control**](https://blog.csdn.net/candy_tse_1/article/details/99452378?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-4.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-4.control)

**6/17/2021**

* 每天花1hrs 练基本功
* UI: POM and methods/debug
* API: requests, token/security/performance/debug
* Projects stories
* Linux
* Database: pytest init and basic ops
* Exceptions

6/13/21

[**https://cloud.tencent.com/developer/article/1730701**](https://cloud.tencent.com/developer/article/1730701)

**6/7/21**

1. Start to use hackerank to practice on-line coding test
2. Chrome to debug

6/6/21

1. Pytest + requests
2. Seleniume: Page Object Model
3. Python selenium 多线程 TEST
4. **Python+selenium 分布式测试工具Grid环境搭建**
5. Git

<https://www.liaoxuefeng.com/wiki/896043488029600/896067008724000>

**12/11/20**

12/18/20

**1/30/2021**

API:

Requests: SSL, cookie, Security with token

* API authentication
* Stateless HTTP authentication
* Stateful HTTP authentication.
* HTTP security
* Cookie security
* **DataBase**

**5/22/21**

**Pytest: request, fixture, do setup, and tear\_down**

**How to use pytest to do selenium and rest API request test.**

**Use real company API to back up story points.**

**8/31/20**

1. API key vs token
2. Http Header’s fields
3. How to create json file, and how to parse json file.
4. Sniff HTTP requests:

* <https://www.telerik.com/fiddler>

1. **HTTP Sniffer and Protocol Analyzer**
2. HTTP cache (browser vs server)
3. REST API cache: Everything needs to be done on client side since no browser help for client request.

My son was using epic through school code scu9955. Since Epic school puts a limit on how much time my son can read with the free school version, it offers an annual subscription for $71.99. I made the subscription. But I just realized I can't manage my subscription since I got in the epic through school code.

Please either cancel my subscription or offer a way to manage my account.

Thanks,

Jian

1. How files are uploaded in http

<form enctype="multipart/form-data" action="http://localhost:3000/upload?upload\_progress\_id=12344" method="POST">

<input type="hidden" **name="MAX\_FILE\_SIZE" value="100000**" />

Choose a file to upload: <input name="uploadedfile" type="file" /><br />

<input type="submit" value="Upload File" />

</form>

Instead of URL encoding the form parameters, the form parameters (including the file data) are sent as sections in a multipart document **in the body of the request**.

In the example above, you can see the input MAX\_FILE\_SIZE with the value set in the form, as well as a section containing the file data. The file name is part of the **Content-Disposition header**.

**Host**: localhost:3000

**Content-Length**: 1325

Origin: http://localhost:3000

... other headers ...

**Content-Type**: multipart/form-data; boundary=----WebKitFormBoundaryePkpFF7tjBAqx29L

**------WebKitFormBoundaryePkpFF7tjBAqx29L**

Content-Disposition: form-data; name="MAX\_FILE\_SIZE"

100000

**------WebKitFormBoundaryePkpFF7tjBAqx29L**

**Content-Disposition**: form-data; name="uploadedfile"; filename="hello.o"

Content-Type: application/x-object

**... contents of file goes here ...**

------WebKitFormBoundaryePkpFF7tjBAqx29L—

9/14/21

**10/18/20** 🡪 5/2021

1. import re

1.1 exceptions: what kind of exceptions and how to handle

1.2 Python assert

2. itertool

3. json files ops

4. glob

5. logging

6. from pywinauto.application import Application

7. panda

8. IOError, OSError

9. parse large json file

11.simplejson

12. jsonpath

13. generator

14. functools

# Python Name convention

# Variable/function names are lower\_caseandseparated\_with\_underscores

# Named constants are in ALL\_CAPITAL\_LETTERS

# Classes are in CamelCase

# topological sort

# bucket sort

# Block bucket: 307. Range Sum Query - Mutable

# all, any, sum

# itertools

# itertools.izip\_longest

# functools.cmp\_to\_key(func)

**11/17/20**

pytest raise

pytest monkeypatch

**4/1/21**

1. conversion between csv and json

2. ijson, Pandas

3. how to deal with large json file

4. Using linux command: jq, split, sed ,awk

jq, in its own words, is "a lightweight and flexible command-line JSON processor.

You can use it to slice and filter and map and transform structured data

with the same ease that sed, awk, grep.

5. pickle

6. python filter, any, all

7. dict comprehension

output\_dict = [{k:v for k,v in x.items() if k in ["Item", "Price"]} for x in input\_dict]

8. -----

Windows Installation

**convert json to csv files, then parse csv file line by line**

The issue here is that JSON, as a format, is generally parsed in full and then handled in-memory,

which for such a large amount of data is clearly problematic.

The solution to this is to work with the data as a stream - reading part of the file, working with it, and then repeating.

The best option appears to be using something like ijson - a module that will work with JSON as a stream, rather than as a block file

https://www.dataquest.io/blog/python-json-tutorial/

https://blog.softhints.com/python-read-huge-json-file-pandas/

https://blog.softhints.com/python-convert-json-to-json-lines/

\_\_\_\_

how to create large json file python

tools to process json data

\_\_\_\_\_

https://lpetr.org/author/lpetrazickis/page/2/

https://www.aylakhan.tech/?p=27

python memory profiler

https://sodocumentation.net/python/topic/2475/-args-and---kwargs

https://www.dataquest.io/blog/python-api-tutorial/

++++++++++++++++

9. pandas vs csv

pandas is for processing data; pandas has more power than python csv module

pandas will export processed data to be saved in csv

10. logging module

python logging level to set

11. tqdm

Instantly make your loops show a smart progress meter - just wrap any iterable with tqdm(iterable)

from tqdm import tqdm

for i in tqdm(range(100000000)):

i = 1

12. Requests-Cache

13. time

# time process

from datetime import datetime

import time

now = datetime.now()

print(now.strftime("%Y-%m-%d, %H:%M:%S"))

14. pip install requests-cache

import requests

import requests\_cache

url = "http://jasonrigden.com"

requests\_cache.install\_cache("my\_sqlite\_db", expire\_after=1000) # expire in 1000 sec

# requests\_cache.disabled() # disable cache

# requests\_cache.clear() # clear cache

# requests\_cache.uninstall\_cache() # remove cache

r = requests.get(url)

print(r.from\_cache) # First time False

# 2nd time run, return True

**5/4/21**

# Virtual Machines

**5/15**

python exception

pytho regular expression