# 要求

**首先为何要打牌?**

我们原本需要学会做到这三步，以用硬件虚拟化技术 控制网络设备。

这里面的第一步，要求熟悉网络中的API的使用方法，于是引入了这个打牌的游戏作为homework，不断请求并从服务器获取结果，来理解网络中API的使用方法，为Step 2，Step 3打基础、

1. Step 1: Learn the basics of the APIC-EM REST API
2. Step 2: Generate and use a new service ticket
3. Step 3: Use network device-related APIs

# Poker-with-API

Implement the code given in http://thinkpython2.com/code (Card.py and PokerHand.py) using https://deckofcardsapi.com/.

## Todos

- Download the requried files or create them in this repo.
- Comment on the design in this readmefile.
- Rewrite the basic classes in Card.py using https://deckofcardsapi.com/
- Modifiy PokerHand.py to accommodate the changes you made
- https://greenteapress.com/thinkpython2/html/thinkpython2019.html has two exercise 2 and 3 for PokerHand.py,
- Can you make a playable game of poker with a automated dealer and few players?

# Prerequisites

- Familiarity with REST APIs and JSON.
- Basic proficiency with Python or a similar high-level programming language.

To study these topics, complete the Coding 101: REST API Basics Lab.

You will make API calls to an APIC-EM controller. By default, these labs use the Cisco DevNet Sandbox's APIC-EM controller at https://sandboxapicem.cisco.com/.

# Python API Note-INWK

Lianda Duan

2020

- duanlianda.blog.csdn.net

# 几个重要的连接

- 测试代码用到的网站 Deck of Cards------官方网站：https://deckofcardsapi.com/
- 增加JSON结果的可读性 https://codebeautify.org/jsonviewer

# 赌徒的基本素养： Deck of Cards API**

## 洗牌 Shuffle the Cards

Add **deck_count** as a **GET or POST** parameter to define the number of Decks you want to use. Blackjack typically uses 6 decks. The default is 1.

使用者需要将deck_count作为发送GET或者POST协议的其中一个参数，这个参数在拼接http报文的时候会被放在连接的最后，这里给出参数为1的时候的拼接结果

```
https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1
```

此时可以得到一个结果：

```
{"success": true, "deck_id": "0y05kmosn3gn", "remaining": 52, "shuffled": true}
```

# 画出这张牌 Draw a Card：

The count variable defines how many cards to draw from the deck. Be sure to replace deck_id with a valid deck_id. We use the deck_id as an identifier so we know who is playing with what deck. After two weeks, if no actions have been made on the deck then we throw it away.

要想画出一张牌，需要下面这个链接：

```
https://deckofcardsapi.com/api/deck/<<deck_id>>/draw/?count=2
```

```
count：想画出几张牌
<<deck_id>>：这个地方是这副牌在服务器上的身份，如果连续两周没有活动，这个身份会失效
使用用一个身份的玩家可以操作同一副牌。
```

TIP: replace <<deck_id>> with "new" to create a shuffled deck and draw cards from that deck in the same request.

我们可以通过将参数**<<deck_id>>**替换为**new**来得到一副新的牌。

以上代码的返回结果*（已经替换了<<deck_id>>）


```
{"success": true, "deck_id": "0y05kmosn3gn", "cards": [{"code": "KH", "image":
"https://deckofcardsapi.com/static/img/KH.png", "images": {"svg":
"https://deckofcardsapi.com/static/img/KH.svg", "png":
"https://deckofcardsapi.com/static/img/KH.png"}, "value": "KING", "suit":
"HEARTS"}, {"code": "8C", "image":
"https://deckofcardsapi.com/static/img/8C.png", "images": {"svg":
"https://deckofcardsapi.com/static/img/8C.svg", "png":
"https://deckofcardsapi.com/static/img/8C.png"}, "value": "8", "suit":
"CLUBS"}], "remaining": 50}
```

# 重新洗牌 Reshuffle the Cards

Don't throw away a deck when all you want to do is shuffle. Include the deck_id on your call to shuffle your cards. Don't worry about reminding us how many decks you are playing with.

一个**<<deck_id>>**代表你正在玩一副指定的牌。如果仅仅是要洗牌，没有必要扔掉一副牌。

可以保留着一副牌，并洗牌。这样就类似避免了一个玩家要重新进入游戏才能洗牌的尴尬。

```
https://deckofcardsapi.com/api/deck/<<deck_id>>/shuffle/
```

和之前的命令一样，把http请求放进浏览器里，会得到如下的输出:

```
{
    "success": true,
    "deck_id": "3p40paa87x90",
    "shuffled": true,
    "remaining": 52
}
```

# 一副全新的牌 A Brand New Deck

要获得一副全新的牌：

```
https://deckofcardsapi.com/api/deck/new/
```

Open a brand new deck of cards. A-spades, 2-spades, 3-spades... followed by diamonds, clubs, then hearts.

NEW (Oct 2019): Add jokers_enabled=true as a GET or POST parameter to your request to include two Jokers in the deck.

```
{
    "success": true,
    "deck_id": "3p40paa87x90",
    "shuffled": false,
    "remaining": 52
}
```

# 拿到一副牌的一部分A Partial Deck

```
https://deckofcardsapi.com/api/deck/new/shuffle/?
cards=AS,2S,KS,AD,2D,KD,AC,2C,KC,AH,2H,KH
```

If you want to use a partial deck, then you can pass the card codes you want to use using the cards parameter. Separate the card codes with commas, and each card code is a just a two character case-insensitive string:

1. The value, one of A (for an ace), 2, 3, 4, 5, 6, 7, 8, 9, 0 (for a ten), J (jack), Q (queen), or K (king);
2. The suit, one of S (Spades), D (Diamonds), C (Clubs), or H (Hearts).

In this example, we are asking for a deck consisting of all the aces, twos, and kings.

在例子中，我们拿出了所有的Aces，2， 和King

Response:

```
{
    "success": true,
    "deck_id": "3p40paa87x90",
    "shuffled": true,
    "remaining": 12
}
```

# 加桩 Adding to Piles

```
https://deckofcardsapi.com/api/deck/<<deck_id>>/pile/<<pile_name>>/add/?
cards=AS,2S
```

Piles can be used for discarding, players hands, or whatever else. Piles are created on the fly, just give a pile a name and add a drawn card to the pile. If the pile didn't exist before, it does now. After a card has been drawn from the deck it can be moved from pile to pile.

Note: This will not work with multiple decks.

google 翻译：

堆可以用于丢弃，玩家的手或其他任何东西。 即时创建桩，只需给桩起一个名称并向桩中添加抽奖牌即可。 如果该桩以前不存在，那么现在就存在。 从一副牌中抽出一张卡片后，就可以将其从一堆移到另一堆。

注意：这不适用于多个卡座。

```
{
    "success": true,
    "deck_id": "3p40paa87x90",
    "remaining": 12,
    "piles": {
        "discard": {
            "remaining": 2
        }
    }
}
```


## 随机桩Shuffle Piles

Note: This will not work with multiple decks.

```
https://deckofcardsapi.com/api/deck/<<deck_id>>/pile/<<pile_name>>/shuffle/
```

### Response:


```
{
    "success": true,
    "deck_id": "3p40paa87x90",
    "remaining": 12,
    "piles": {
        "discard": {
            "remaining": 2
        }
    }
}
```


## 把桩中的牌列出 Listing Cards in Piles

```
https://deckofcardsapi.com/api/deck/<<deck_id>>/pile/<<pile_name>>/list/
```

Note: This will not work with multiple decks.

Reponse:


```
{
    "success": true,
    "deck_id": "d5x0uw65g416",
    "remaining": "42",
```

```
    "piles": {
        "player1": {
            "remaining": "3"
        },
        "player2": {
            "cards": [
                {
                    "image": "https://deckofcardsapi.com/static/img/KH.png",
                    "value": "KING",
                    "suit": "HEARTS",
                    "code": "KH"
                },
                {
                    "image": "https://deckofcardsapi.com/static/img/8C.png",
                    "value": "8",
                    "suit": "CLUBS",
                    "code": "8C"
                }
            ],
            "remaining": "2"
        }
    },
}
```


# 画出桩中的牌Drawing from Piles

```
https://deckofcardsapi.com/api/deck/<<deck_id>>/pile/<<pile_name>>/draw/?cards=AS
https://deckofcardsapi.com/api/deck/<<deck_id>>/pile/<<pile_name>>/draw/?count=2
https://deckofcardsapi.com/api/deck/<<deck_id>>/draw/bottom/
```

Specify the cards that you want to draw from the pile. The default is to just draw off the top of the pile (it's a stack). Or add the bottom parameter to the URL to draw from the bottom.

指定要从堆中抽出的牌。 默认设置是仅从堆的顶部抽出（它是一个堆栈）。 或将底部参数添加到要从底部绘制的URL。

Reponse:


```
{
    "success": true,
    "deck_id": "3p40paa87x90",
```

```
    "remaining": 12,
    "piles": {
        "discard": {
            "remaining": 1
        }
    },
    "cards": [
        {
            "image": "https://deckofcardsapi.com/static/img/AS.png",
            "value": "ACE",
            "suit": "SPADES",
            "code": "AS"
        }
    ]
}
```


# REST & API

- ***REST*** stands for **RE**presentational **S**tate **T**ransfer
- An ***API*** is an **A**pplication **P**rogramming **I**nterface

## Objectives

This lab teaches a number of things about REST APIS:

- Understand REST API concepts.
- Learn the basics of consuming REST APIs.
- Learn the standard verbs for REST APIs.
- Understand JSON and XML Payloads.
- Practical examples using REST APIs.

## REST API status CODE

| Status Code | Status Message | Meaning |
| --- | --- | --- |
| 200 | OK | All looks good |
| 201 | Created | New resource created |
| 400 | Bad Request | Request was invalid |
| 401 | Unauthorized | Authentication missing or incorrect |
| 403 | Forbidden | Request was understood but not allowed |
| 404 | Not Found | Resource not found |
| 500 | Internal Server Error | Something wrong with the server |
| 503 | Service Unavailable | Server is unable to complete request |

注意如果是500打头的出了错误一定是服务器那边的问题

# APIC-EM APIs with Python

## 基本操作

- **POST** creates data.
- **GET** reads (retrieves) data.
- **PUT** updates data.
- **DELETE** deletes data.

## GET

To get data, a Python script issues a GET request, such as:

```
import requests
url = "https://fqdn-Or-IPofController/api/v1/api_itself"
response = requests.get(url,verify=False)
# verify=False means not verifying the SSL certificate
```

## POST

To create data, a Python script issues a POST request, such as:

```
import requests
import json

json_obj = {
"key":"value"
}
url = "https://fqdn-Or-IPofController/api/v1/api_itself"
# need to specify content type -json- for POST request #
headers = {'content-type': 'application/json'}
response = requests.post(url, json.dumps(json_obj),
headers=headers,verify=False)
```

## PUT

To modify data, a Python script issues a PUT request, such as:

```
import requests
import json

json_obj = {
"key":"value_to_change"
}
url = "https://fqdn-Or-IPofController/api/v1/api_itself"
# need to specify content type -json- for PUT request #
headers = {'content-type': 'application/json'}
response = requests.put(url, json.dumps(json_obj),
headers=headers,verify=False)
```

### DELETE

To delete data, a Python request issues a DELETE request, such as:

```
import requests
url = "https://fqdn-Or-IPofController/api/v1/api_itself"
response = requests.delete(url,verify=False)
```

# 实战笔记

## from future import print_function用法

阅读代码的时候会看到下面语句：

```
from __future__ import print_function
```

在python2.x的环境是使用下面语句，则第二句语法检查通过，第三句语法检查失败

```
1 from __future__ import print_function
2 print('you are good')
3 print 'you are good'
```

所以以后看到这个句子的时候，不用害怕，只是把下一个新版本的特性导入到当前版本！

## 未完待续

# References


[1] https://deckofcardsapi.com/ 官方说明

[2] https://developer.cisco.com/learning/tracks/devnet-beginner/network-programmability/apic-em-basic/step/1

[3] https://developer.cisco.com/learning/labs/what-are-rest-apis/step/1

[4] https://zhuanlan.zhihu.com/p/28641474