



Assignment Name: Lab 12

Names: Jesus Andres Bernal Lopez, Paul Whipp, Michael Avalos-Garcia

Summary: The lab was extremely straight forward and easy to complete. For task1 we had already turned it in completed. For task2 we found an interesting pokeAPI with many options. For task3 we found a clash royaleAPI and we messed around with it, luckily we didn't get elbowed(Torin if you're grading this you will know what we're talking about).

Task 1

Last modified	Monday, 11 March 2019, 12:52 AM
File submissions	 Lab11Updated2.0.pdf
Submission comments	 Comments (0)

Task 2

Authentication

All API requests must be accompanied by a developer key. This key is a unique identifier to an individual developer and you should not share it with anyone else.

Authenticated Requests

You must authenticate your requests using **custom headers**. Add a key named `auth` with value `<token>` to your header when you make your requests.

Additionally, we support **Bearer Token** authorization. Bearer tokens use a field named `Authorization` with value `Bearer <token>` in the header.

Set your token as an **environment variable** or save it as part of the Authorization setting in your **Postman Collection** settings so you don't need to ever type it again.

Key Management

1. Once issued, your key will be active.
2. Your developer key will remain active as long as you remain a user on our Discord server. Your key will be automatically disabled once you have left the server.
3. We reserve the right to permanently blacklist keys / developers who have been shown to abuse the API with requests.

```
"""
Contributors: Jesus Andres Bernal Lopez, Paul Whipp, Michael Avalos-Garcia
File: lab12.py
Date: 03/13/2019
Description: We found Royale API and made a PyQt Window of it that asks the user for
their player tags and returns back their chest cycle
"""

import requests
from dotenv import load_dotenv
import os
from PyQt5.QtWidgets import QWidget, QApplication, QVBoxLayout, QLabel, QPushButton,
QLineEdit
from PyQt5.QtCore import pyqtSlot
import sys

class Window(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Clash Royale Chest Cycle")

        vbox = QVBoxLayout()

        label = QLabel("Enter your player tag(without the #)")
        self.user_input = QLineEdit()
        self.my_button = QPushButton("Get Chest Cycle")
        self.my_button.clicked.connect(self.button_clicked)
        self.chest_label = QLabel()
        self.chest_label.isHidden()
```

```

vbox.addWidget(label)
vbox.addWidget(self.user_input)
vbox.addWidget(self.my_button)
vbox.addWidget(self.chest_label)

self.setLayout(vbox)

@pyqtSlot()
def button_clicked(self):
    url =
f"https://api.royaleapi.com/player/{self.user_input.text().upper()}/chests"

    headers = {
        'auth': my_key
    }

    response = requests.request("GET", url, headers=headers)

    data = response.json()
    self.chest_label.setText(f"""
Mega Lightning: {data['megaLightning']}\n
Magical: {data['magical']}\n
Legendary: {data['legendary']}\n
Epic: {data['epic']}\n
Giant: {data['giant']}
""")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    main_win = Window()
    main_win.show()
    # loads the .env file and stores the API Key in variable
    load_dotenv()
    my_key = os.getenv("API_KEY")
    sys.exit(app.exec_())

```

Results of task 3:

Clash Royale Chest Cycle

Enter your player tag(without the #)

2GUQQ8JGC

Get Chest Cycle

Mega Lightning: 8

Magical: 42

Legendary: 345

Epic: 229

Giant: 10