Coding Two: Advanced Frameworks

Assignment Element 1: Lab Work

Week 3 Exercise - The Python Challenge!

http://www.pythonchallenge.com

Submit your python challenge solutions to the first 7 challenges

0

The upper left corner shows level 0, and there is also 0 in the webpage code. The prompt is to change the URL address, then 2^38 is the answer, so we use programming to calculate:

```
2**38
```

Get 274877906944, replace this number with the previous 0 position, and enter level 1.

```
<html>
<head>
  <title>ocr</title>
  <link rel="stylesheet" type="text/css" href="../style.css">
</head>
<body>
<center><img src="ocr.jpg">
<br><font color="#c03000">
recognize the characters. maybe they are in the book, <br>but MAYBE they
are in the page source.</center>

<br>
<br>
<br>

<font size="-1" color="gold">
General tips:
<li>Use the hints. They are helpful, most of the times.</li>
<li>Investigate the data given to you.</li>
<li>Avoid looking for spoilers.</li>
<br>
Forums: <a href="http://www.pythonchallenge.com/forums"/>Python Challenge Forums</a>,
read before you post.
<br>
IRC: irc.freenode.net #pythonchallenge
<br><br>
To see the solutions to the previous level, replace pc with pcc, i.e. go
to: http://www.pythonchallenge.com/pcc/def/ocr.html

</body>
</html>

<!--
find rare characters in the mess below:
-->

<!--
%%$@_$^__#)^)&!_+]!*@&^}@[@%]()%+$&[(_@%+%$*^@$^!+]!&_#)_*}{}}!}_]$[%}@[{_@#_^{*
@##&{#&{&)*%(]{{([*}@[@&]+!!*{)!}{%+{))])[!^})+)$]#{*+^((@^@}$[**$&^{$!@#$%)!@(&
+^!{%_$&@^!}$_${)$_#)!({@!)(^}!*^&!$%_&&}&_#&@{)]{+)%*{&*%*&@%$+]!*__(#!*){%&@++
!_)^$&&%#+)}!@!)&^}**#!_$([$!$}#*^}$+&#[{*{}{((#$]{[$[$$()_#}!@}^@_&%^*!){*^^_$^
]@}#%[%!^[^_})+@&}{@*!(@$%$^)}[_!}(*}#}#___}!](@_{{(*#%!%%+*)^+#%}$+_]#}%!**#!^_
)@)$%%^{_%!@(&{!}$_$[)*!^&{}*#{!)@})!*{^&[&$#@)*@#@_@^_#*!@_#})+[^&!@*}^){%%{&#@
@{%(&{+(#^{@{)%_$[+}]$]^{^#(*}%)@$@}(#{_&]#%#]{_*({(])$%[!}#@@&_)([*]}$}&${^}@(%
(%[@%!}%*$}(*@)}){+@(%@*$&]*^*}*]&$[}*]%]+*}^!}*$^^_()#$^]++@__){&&+((#%+(&+){)$
%&&#($[[+##*%${)_!+{_[})%++)$#))]]]$]@]@($+{&%&%+!!!@]_]+])^*@$(@#${}}#}{%}#+{(@
```

1

In the first level, you need to translate a paragraph of unintelligible text. According to the text prompt, you need the last two digits to get the real letter. The rules can be understood by the picture verification.

```
s='''g fmnc wms bgblr rpylqjyrc gr zw fylb. rfyrq ufyr amknsrcpq ypc dmp. bmgle gr gl zw fylb gq glcddgagclr ylb rfyr'q ufw rfgq rcvr gq qm jmle. sqgle qrpgle.kyicrpylq() gq pcamkkclbcb. lmu ynnjw ml rfc spj.'''
for i in range(len(s)):
```

```
    if ord(s[i])>=ord('a') and ord(s[i])<=ord('x'):
        c=chr(ord(s[i])+2)
        print(c,end='')
    elif ord(s[i])>=ord('y') and ord(s[i])<=ord('z'):
        c=chr(ord(s[i])-24)
        print(c,end='')
    else:
        print(s[i],end='')
```

The output result is：

i hope you didn’t translate it by hand. That’s what computers are for. doing it in by hand is inefficient and that's why this text is so long. using string.maketrans() is recommended. now apply on the URL.

Just replace map with ocr, so the result enters the second level.

2

When we right-click to view the source code, we will find that the comment allows us to find rare characters, and then start programming to search:

```
from collections import Counter
strings='''长长的字符'''
c = Counter(strings)
print(c.most_common())
```

It turned out that some letters appeared once

```
[(')', 6186), ('@', 6157), ('(', 6154), (']', 6152), ('#', 6115), ('_', 6112), ('[', 6108), ('}', 6105), ('%', 6104), ('!', 6079), ('+', 6066), ('$', 6046), ('{', 6046), ('&', 6043), ('*', 6034), ('^', 6030), ('\n', 1219), ('e', 1), ('q', 1), ('u', 1), ('a', 1), ('l', 1), ('i', 1), ('t', 1), ('y', 1)]
```

print it out：

```
from collections import Counter
strings='''长长的字符'''
c = Counter(strings)
print(c.most_common())
print(''.join([i[0] for i in c.items() if i[1]==1]))
```

结果是：equality

于是得到新的 URL:

http://www.pythonchallenge.com/pc/def/equality.html

3

Prompt that you need to find lowercase letters, and you need to have three uppercase letters on both sides. Right-click to view the network code.

Similarly, we can see that there is a comment with amazing length. At this time, we noticed that the title of the webpage is re, so we use regular expressions to find it:

(Explain, because there can only be three uppercase letters on both sides, so we add [a-z] on both sides)

```
import re
strings='''长长的字符串'''
reg=re.compile('[a-z][A-Z]{3}[a-z][A-Z]{3}[a-z]')
print(' '.join(reg.findall(dat)))
```

The results are:

```
qIQNlQSLi eOEKiVEYj aZADnMCZq bZUTkLYNg uCNDeHSBj kOIXdKBFh dXJVlGZVm gZAGiLQZx vCJAsACFl qKWGtIDCj
```

The lowercase letter in the middle is what we are looking for, modify the code:

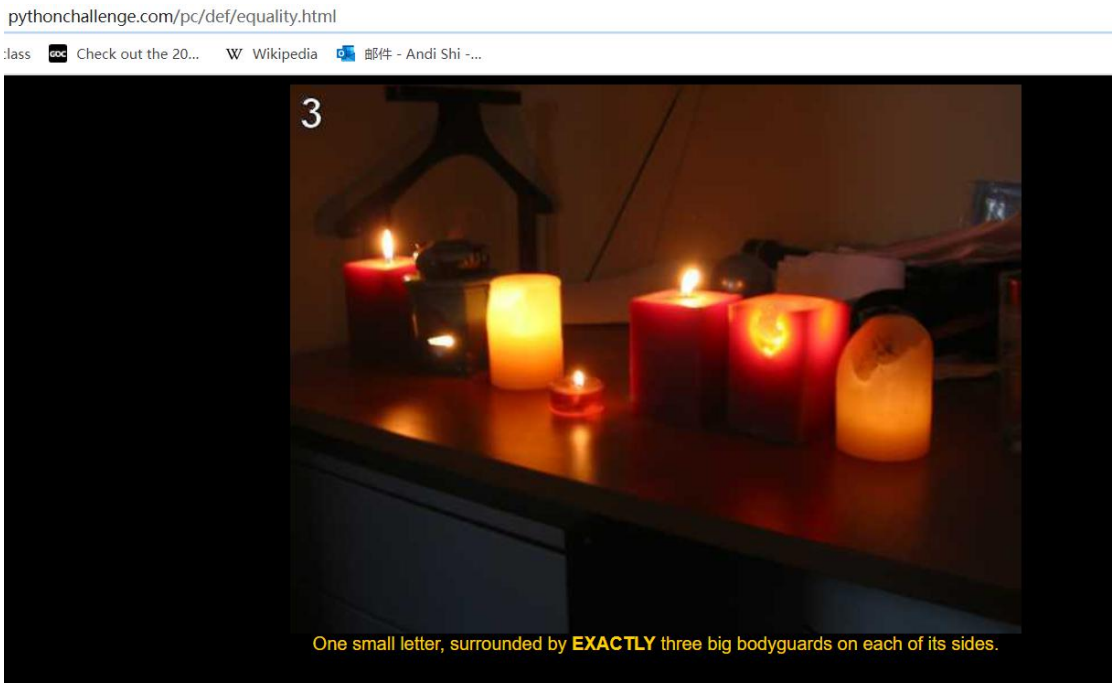
```
result=reg.findall(dat)
url=''
for res in result:
    url+=res[4]
print(url)
```

The printed result is: linkedlist, get the new url:

http://www.pythonchallenge.com/pc/def/linkedlist.html

After opening, there is only one sentence linkedlist.php on the page, we replace the url again:

Get: http://www.pythonchallenge.com/pc/def/linkedlist.php This is the final result.



4

Click on the picture to see a sentence and the next nothing is 44827, right click the source code.

```
<html>
<head>
  <title>follow the chain</title>
  <link rel="stylesheet" type="text/css" href="../style.css">
</head>
<body>
<!-- urllib may help. DON'T TRY ALL NOTHINGS, since it will never
end. 400 times is more than enough. -->
<center>
<a href="linkedlist.php?nothing=12345"><img src="chainsaw.jpg" border="0"/></a>
<br><br><font color="gold"></center>
Solutions to previous levels: <a href="http://wiki.pythonchallenge.com/"/>Python Challenge wiki</a>.
<br><br>
IRC: irc.freenode.net #pythonchallenge
</body>
</html>
```

We can see that the following two lines of text, it provides us with a library urllib, and then tells us not to try all nothing, because it is infinite, 400 cycles are enough...

```
<!-- urllib may help. DON'T TRY ALL NOTHINGS, since it will never
end. 400 times is more than enough. -->
```

And the text that appears is and the next nothing is 44827

This is very clear, the usefulness of the crawler came out, prompting us to use the urllib library, but I still use requests to solve the problem

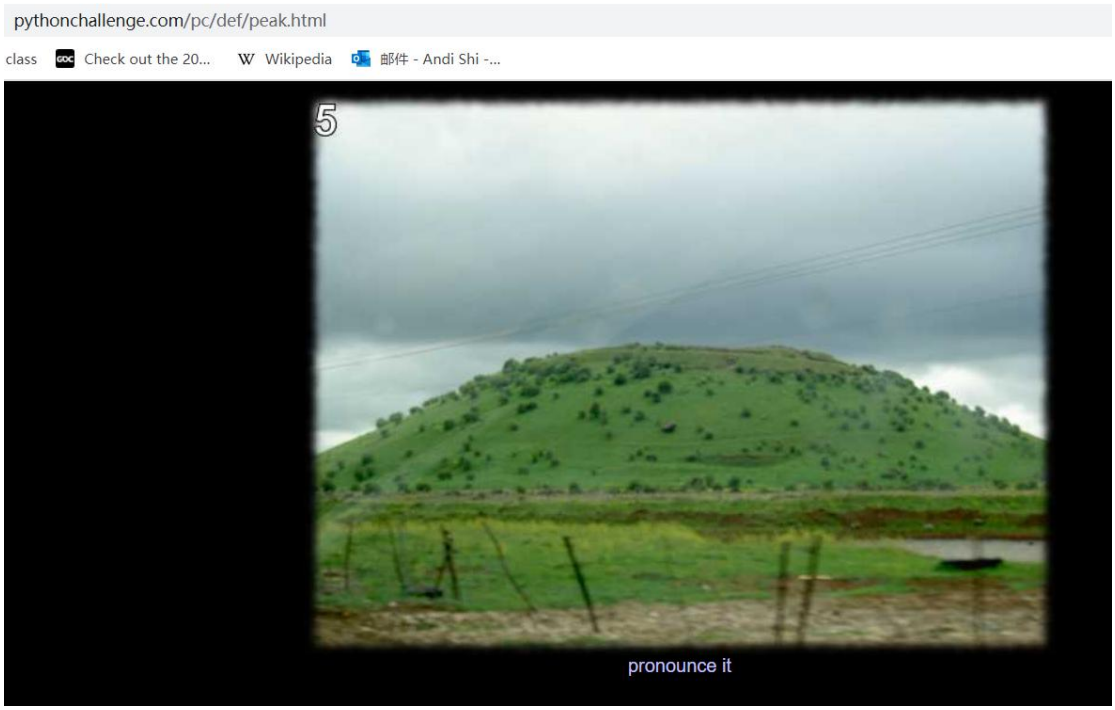
import requests

```
url='http://www.pythonchallenge.com/pc/def/linkedlist.php?nothing=12345'
```

```
for i in range(400):
    url='http://www.pythonchallenge.com/pc/def/linkedlist.php?nothing='+txt.split()[-1]
    r=requests.get(url)
    txt=r.text
    print(txt)
```

Although it is wasteful to print out the results every time, don't omit it, because maybe some pages are different? ? And when it is printed out, we can stop the running of the program, because this loop is very time-consuming. I started to print the result when the loop was 400th time, but it didn't work. I tried several times before and after 400, so I printed it, and it was printed out once about 250, and the result was like this.

```
and the next nothing is 41643
and the next nothing is 23416
and the next nothing is 54432
and the next nothing is 4448
.....Many lines are omitted here
peak.html
and the next nothing is 72758
and the next nothing is 71301
```



5

View the source code

```
<html>
<head>
  <title>peak hell</title>
  <link rel="stylesheet" type="text/css" href="../style.css">
</head>
```

```
<body>
<center>
<img src="peakhell.jpg"/>
<br><font color="#c0c0ff">
pronounce it
<br>
<peakhell src="banner.p"/>
</body>
</html>
<!-- peak hell sounds familiar ? -->
```

You can see the comment <!-- peak hell sounds familiar? --> Does it sound familiar? ? In fact, it is the pickle library. Get the URL: http://www.pythonchallenge.com/pc/def/pickle.html and find a sentence yes! pickle! The pickle tag of the webpage source code can also get this conclusion.

```
import pickle
from urllib.request import urlretrieve
url = 'http://www.pythonchallenge.com/pc/def/banner.p'
urlretrieve(url, filename = 'five.pkl')
with open('five.pkl', 'rb') as f:
    r = pickle.load(f)
```

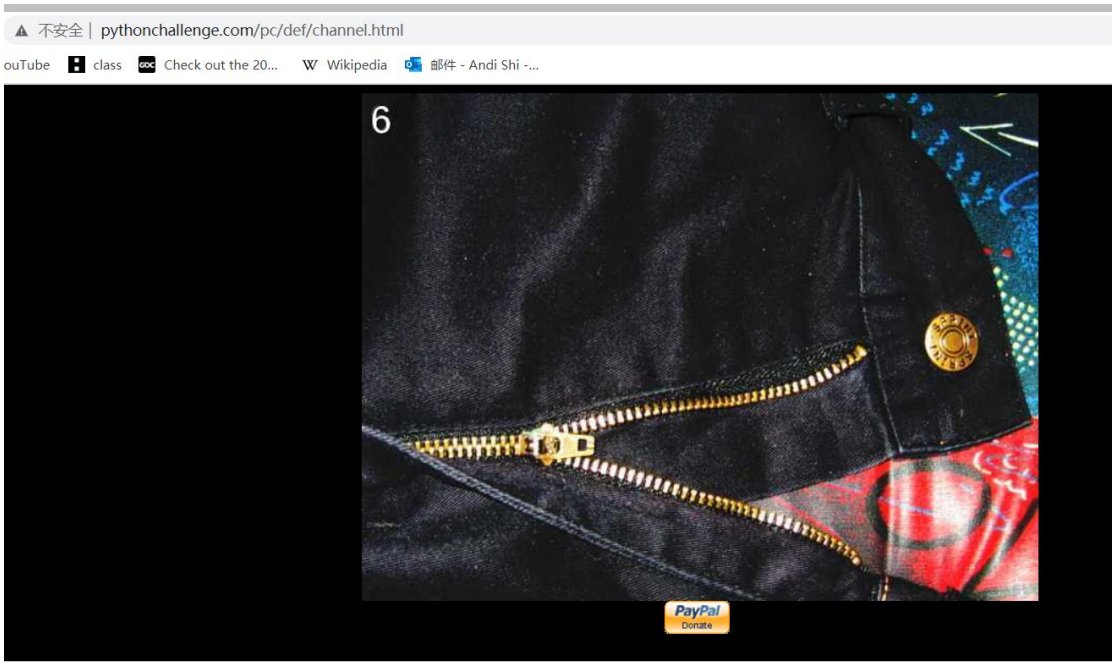
The result is

```
[(' ', 95)], [(' ', 14), ('#', 5), (' ', 70), ('#', 5), (' ', 1)], [(' ', 15), ('#', 4), (' ', 71), ('#', 4), (' ', 1)], [(' ', 15), ('#', 4), (' ', 71), ('#', 4), (' ', 1)], [(' ', 15), ('#', 4), (' ', 71), ('#', 4), (' ', 1)], [(' ', 15), ('#', 4), (' ', 71), ('#', 4), (' ', 1)], [(' ', 15), ('#', 4), (' ', 71), ('#', 4), (' ', 1)], [(' ', 15), ('#', 4), (' ', 71), ('#', 4), (' ', 1)], [(' ', 6), ('#', 3), (' ', 6), ('#', 4), (' ', 3), ('#', 3), (' ', 9), ('#', 3), (' ', 7), ('#', 5), (' ', 3), ('#', 3), (' ', 4), ('#', 5), (' ', 3), ('#', 3), (' ', 10), ('#', 3), (' ', 7), ('#', 4), (' ', 1)], [(' ', 3), ('#', 3), (' ', 3), ('#', 2), (' ', 4), ('#', 4), (' ', 1), ('#', 7), (' ', 5), ('#', 2), (' ', 2), ('#', 3), (' ', 6), ('#', 4), (' ', 1), ('#', 7), (' ', 3), ('#', 4), (' ', 1), ('#', 7), (' ', 5), ('#', 3), (' ', 2), ('#', 3), (' ', 5), ('#', 4), (' ', 1)], [(' ', 2), ('#', 3), (' ', 5), ('#', 3), (' ', 2), ('#', 5), (' ', 4), ('#', 4), (' ', 3), ('#', 3), (' ', 3), ('#', 4), (' ', 4), ('#', 5), (' ', 4), ('#', 4), (' ', 2), ('#', 5), (' ', 4), ('#', 4), (' ', 3), ('#', 3), (' ', 5), ('#', 3), (' ', 3), ('#', 4), (' ', 1)], [(' ', 1), ('#', 3), (' ', 11), ('#', 4), (' ', 5), ('#', 4), (' ', 3), ('#', 3), (' ', 4), ('#', 3), (' ', 4), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 3), (' ', 6), ('#', 4), (' ', 2), ('#', 4), (' ', 1)], [(' ', 1), ('#', 3), (' ', 11), ('#', 4), (' ', 5), ('#', 4), (' ', 10), ('#', 3), (' ', 4), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 3), (' ', 7), ('#', 3), (' ', 2), ('#', 4), (' ', 1)], [('#', 4), (' ', 11), ('#', 4), (' ', 5), ('#', 4), (' ', 5), ('#', 2), (' ', 3), ('#', 3), (' ', 4), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 1), ('#', 4), (' ', 7), ('#', 3), (' ', 2), ('#', 4), (' ', 1)], [('#', 4), (' ', 11), ('#', 4), (' ', 5), ('#', 4), (' ', 3), ('#', 10), (' ', 4), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 1), ('#', 14), (' ', 2), ('#', 4), (' ', 1)], [('#', 4), (' ', 11), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 3), (' ', 4), ('#', 4), (' ', 4), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 1), ('#', 4), (' ', 12), ('#', 4), (' ', 1)], [('#', 4), (' ', 11), ('#', 4), (' ', 5), ('#', 4), (' ', 1), ('#', 4), (' ', 12), ('#', 4), (' ', 1)], [(' ', 1), ('#', 3), (' ', 11), ('#', 4), (' ', 5), ('#', 4), (' ', 1), ('#', 4), (' ', 5), ('#', 3), (' ', 4), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 1), ('#', 4), (' ', 12), ('#', 4), (' ', 1)], [(' ', 2), ('#', 3), (' ', 6), ('#', 2), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 3), (' ', 4), ('#', 4), (' ', 4), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 3), ('#', 3), (' ', 6), ('#', 2), (' ', 3), ('#', 4), (' ', 1)], [(' ', 3), ('#', 3), (' ', 4), ('#', 2), (' ', 3), ('#', 4), (' ', 5), ('#', 4), (' ', 3), ('#', 11), (' ', 3), ('#', 4), (' ', 5), ('#', 4), (' ', 2), ('#', 4), (' ', 5), ('#', 4), (' ', 4), ('#', 3), (' ', 4), ('#', 2), (' ', 4), ('#', 4), (' ', 1), ('#', 6), (' ', 4), ('#', 11), (' ', 4), ('#', 5), (' ', 6), ('#', 3), (' ', 6), ('#', 6)], [(' ', 95)]]
```

It seems that the number should represent the number of characters, change the code:

```
from urllib.request import urlretrieve
import pickle
url = 'http://www.pythonchallenge.com/pc/def/banner.p'
```

```
urlretrieve(url, filename = 'five.pkl')
with open('five.pkl', 'rb') as f:
    r = pickle.load(f)
for i in r:
    for j in i:
        print(j[0] * j[1], end = '')
    print('\n')
```



6

The source code hint has nothing to do with donate, but you can see the word "zip" and the picture is also a zipper.

```
<!-- The following has nothing to do with the riddle itself. I just
thought it would be the right point to offer you to donate to the
Python Challenge project. Any amount will be greatly appreciated.

-thesamet
```

In fact, his other meaning is to prompt the compressed package, http://www.pythonchallenge.com/pc/def/channel.zip visit the link to download the compressed package.

After opening it, there are more than 900 txt files in it. Take a look at it. The last one is read me. After opening it as follows, it seems to be similar to the fourth question.

```
welcome to my zipped list.

hint1: start from 90052
hint2: answer is inside the zip
```

Start programming

```
import zipfile as zi

path = "C:\\Users\\sky\\Desktop\\channel.zip"
```

```
# 使用 zipfile 包解压并读取文件内容到 files
files = {}
fzip = zi.ZipFile(path)
for name in fzip.namelist():
    with fzip.open(name) as fz:
        files[name] = fz.read().decode("utf-8")

# readme.txt 中 nothing 初始值
nothing = "90052"
while True:
    f = nothing + ".txt"
    strs=str(files[f])
    print(strs)
    try:
        nothing = strs.split()[-1]
    except:
        Break
```

The result is

```
Omit n lines here
Next nothing is 67824
Next nothing is 46145
Collect the comments.
```

Make corrections below

```
import re
上面的代码
while True:
    fz = nothing + ".txt"
    # 获取 comment 并输出
    print(fzip.getinfo(fz).comment.decode("utf-8"), end="")
    if fz in files:
        # print(files[fz])
        result = re.search(r"Next nothing is (\d+)", files[fz])
        try:
            nothing = result.group(1)
        except:
```

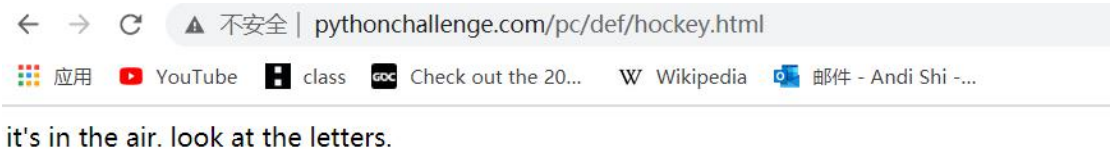
The result is

```
**********************************************************************
**********************************************************************
**                                                                  **
**    OO      OO      XX          YYYY      GG      GG   EEEEEE NN          NN   **
**    OO      OO   XXXXXX     YYYYYY     GG     GG     EEEEEE   NN       NN     **
**    OO      OO XXX   XXX YYY     YY   GG GG        EE           NN  NN       **
**    OOOOOOOO XX       XX YY             GGG          EEEEE        NNNN        **
```

```
**    OOOOOOOO XX      XX YY          GGG         EEEEE        NN        **
**    OO      OO XXX   XXX YYY     YY   GG GG       EE           NN        **
**    OO      OO  XXXXXX    YYYYYY     GG    GG     EEEEEE       NN        **
**    OO      OO     XX        YYYY      GG      GG   EEEEEE       NN        **
**                                                                  **
**********************************************************************
 **********************************************************************
```

Obviously the word hockey, get the link http://www.pythonchallenge.com/pc/def/hockey.html

After opening it, it prompts it's in the air. look at the letters. In the air, look at the letters.



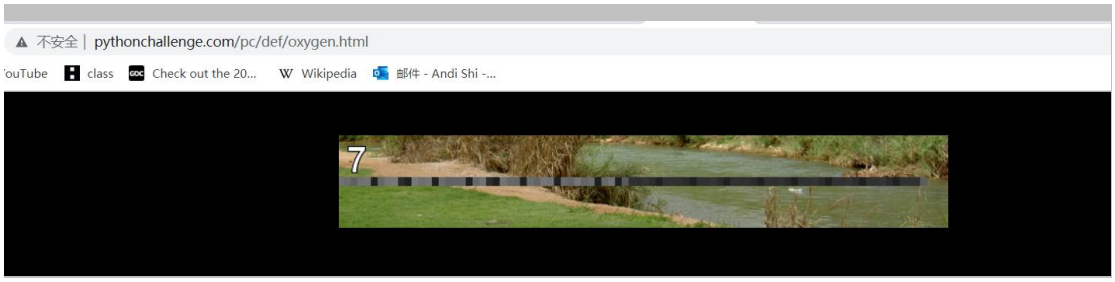
The above result is composed of O X Y G E N. Obviously, this is oxygen, so the answer is:

Obviously the word hockey, get the link http://www.pythonchallenge.com/pc/def/hockey.html

After opening it, it prompts it's in the air. look at the letters. In the air, look at the letters, do you see it?

The above result is composed of O X Y G E N. Obviously, this is oxygen, so the answer is:

http://www.pythonchallenge.com/pc/def/oxygen.html



7

Since there is no hint, the mosaic is in the middle, and PIL is used to process the image.

```
from PIL import Image
im = Image.open("oxygen.png")   #文件路径
width, height = im.size   # 获取尺寸
pic = im.load()
h = height // 2
for x in range(width):
    print(pic[x,h])
```

Get

省略……

(97, 97, 97, 255)

省略……

(100, 100, 100, 255)

省略……

省略……

(116, 116, 116, 255)

省略……

It can be seen that R, G, and B are the same, using ASKII

```
from PIL import Image
im = Image.open("oxygen.png")  #文件路径，如果没有路径就是当前目录下文件
width, height = im.size  # 获取图像尺寸
pic = im.load()
h = height // 2
ss=0  ##ss 是我随便定义一个中间变量，用来存储上一次的值
for x in range(width):
    r, g, b, x = pic[x, h]
    if r != g:
        continue
    if ss!=r:
        print(chr(r), end='')
    ss=r
```

The print result is

```
smart guy, you made it. the next level is [105, 10, 16, 101, 103, 14, 105, 16, 121]
answer = [105, 110, 116, 101, 103, 114, 105, 116, 121]
for l in answer:
    print(chr(l), end='')
```

The output is integrity