


# HackerGame 部分题解

 来自Sxrhhh，得分2350，组内排名第10

## ✧ Hackergame 启动！

在界面内随便读两句后，提交，发现URL内出现了 **similarity** 参数，将其改为任意大于100的数字即可。

## ✧ 猫咪小测

### 图书馆借阅

要查看这本书，来到中科大图书馆官网查询：[中国科学技术大学图书馆 \(ustc.edu.cn\)](http://ustc.edu.cn)

如果查不出来，缩减其中的关键字，如图：



The screenshot shows the '中国科学技术大学图书馆书目检索系统' (USTC Library Catalog System) search results. The search term is 'A classical introduction to modern number theory'. The results are sorted by relevance, showing three items:

- 1. A classical introduction to modern number theory = 现代数论经典引论 / 2nd ed.**  
Ireland, Kenneth F.  
世界图书出版公司, 2003.  
馆藏复本: 6  
可借复本: 6
- 2. A classical introduction to modern number theory /**  
Ireland, Kenneth F.  
Springer-Verlag, 1982.  
馆藏复本: 14  
可借复本: 14
- 3. Introduction to black hole physics /**  
Frolov, V. P.  
Oxford University Press, USA, 2011.  
馆藏复本: 1  
可借复本: 1

The left sidebar shows the search filters: '检索词' (Search Term) with the input 'A Classical Introduction To Modern Number Theory', '出版年' (Publication Year) set to '从 1900 到 2023', and '主题词' (Subject Term) with 'Number theory. (2)' selected.

显然是其中第一本书，点击查看馆藏，找到西区的书，将鼠标悬停即可发现其楼层：

索书号	条码号	年卷期	馆藏地	书刊状态
O156/I 65A(2)	ZW015853	2003.	高新园区综合书库（9层）	可借
O156/I 65A(2)	80014676	2003 -	东区外文图书借阅室	可借
O156/I 65A(2)	80014677	2003 -	西区外文书库	可借
O156/I 65A(2)	80021866	2003 -	西区外文书库	可借
O156/I 65A(2)	80021867	2003 -	西区外文 西区图书馆12楼	可借
O156/I 65A(2)	80021868	2003 -	西区外文书库	可借

答案即为12。

## 鸡的密度

打开arXiv网站，找到天体物理astro-ph板块，搜索chicken关键字，找到目标论文，发现答案，如图：

**Nuggets of Wisdom: Determining an Upper Limit on the Number Density of Chickens in the Universe**

Authors: [Rachel Losacco](#), [Zachary Claytor](#)

**Abstract:** The lower limit on the **chicken** density function (CDF) of the observable Universe was recently determined to be approximately  $10^{-21}$  **chickens**  $\text{pc}^{-3}$ . For over a year, however, the scientific community has struggled to determine the upper limit to the CDF. Here we aim to determine a reasonable upper limit to the CDF using multiple observational constraints. We take a holistic approach to considering the effects of a high CDF in various domains, including the Solar System, interstellar medium, and effects on the cosmic microwave background. We find the most restrictive upper limit from the domains considered to be  $10^{23} \text{ pc}^{-3}$ , which ruffles the feathers of long-standing astrophysics theory. [Less](#)

Submitted 30 March, 2023; originally announced March 2023.

Comments: 5 pages, 1 figure, 1 table, 0 chickens were harmed

答案即为23。

## linux编译选项

先找到linux内核配置详解，如：[史上最全linux内核配置详解-CSDN博客](#)，全文搜索“拥塞”，可找到：

□ TCP: advanced congestion control --->, 高级拥塞控制,如果没有特殊需求(比如无线网络)就别选了,内核会自动将默认的拥塞控制设为"Cubic"并将"Reno"作为候补,可以不选。如果网络用处较多,还是选上为好(如网络视频服务器)。关于拥塞控制,见tcp拥塞控制.doc文档。

锁定关键字：“Cubic”，查询linux下编译选项位置，锁定文件/proc/config.gz，用自己的linux系统打开它，综合搜索"TCP", "BBR", "cubic"，锁定配置位置

```
CONFIG_TCP_CONG_CUBIC=y
CONFIG_DEFAULT_TCP_CONG="cubic"
```

由第一行可知，配置拥塞算法的选项可能为“CONFIG\_TCP\_CONG\_BBR”，填入答案。

另外由官方题解可知，在google中搜索kernal config bbr即可知道答案，因此要求学会使用google并且会用英文提问。

## mypy死循环

翻译" Python 的类型检查和停机问题一样困难"为英文，得到“Python's type checking is just as difficult as the downtime problem”，修改其中专有名词，如：

停机问题：halting problem

将其放入google搜索，第一条即为所在论文 [《Python Type Hints Are Turing Complete》](#)

在论文头部，找到散落在文章各处的标识，即为所在会议：

2012 ACM Subject Classification Software and its engineering → General programming languages

Keywords and phrases nominal Subtyping with Variance, Python

Digital Object Identifier 10.4230/LIPICs.ECOOP.2023.44

Category Pearl/Brave New Idea

Related Version Previous Version: <https://doi.org/10.48550/arxiv.2208.14755>

Supplementary Material Software [ECOOP 2023 Artifact Evaluation approved artifact](https://doi.org/10.4230/DARTS.9.2.1):  
<https://doi.org/10.4230/DARTS.9.2.1>

Acknowledgements The author would like to thank to Yossi Gil for inspiring them to study the topic of nominal subtyping with variance.

### 1 Introduction

Python enhancement proposal (PEP) 484 introduced optional type hints to the Python programming language, together with a full-blown gradual type system [16]. Tools such as Mypy [9] use type hints to type-check Python programs. Certain programs, however, cause Mypy to enter an infinite loop (we show an example below). We argue that the reason behind these failures is not a Mypy bug, but a deeper issue in the PEP 484 type system. We use Grigore's reduction from Turing machines (TMs) to nominal subtyping with variance [6] to prove that Python type hints are, in fact, Turing complete. In other words, checking whether a Python program is correctly typed is as hard as the halting problem.

#### 1.1 Nominal Subtyping With Variance

Subtyping is a type system decision problem. Given types  $t$  and  $s$ , the type system should decide whether type  $t$  is a subtype of  $s$ ,  $t <: s$ , meaning that every  $t$  object is also a member of  $s$ . For example, every string is an object, `str <: object`, but not every object is a string, `object <: str`. Subtyping is needed, for example, for checking variable assignments:

 © Ori Roth;  
licensed under Creative Commons License CC-BY 4.0  
37th European Conference on Object-Oriented Programming (ECOOP 2023).  
Editors: Karim Ali and Guido Salvaneschi; Article No. 44; 1–12  
 Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



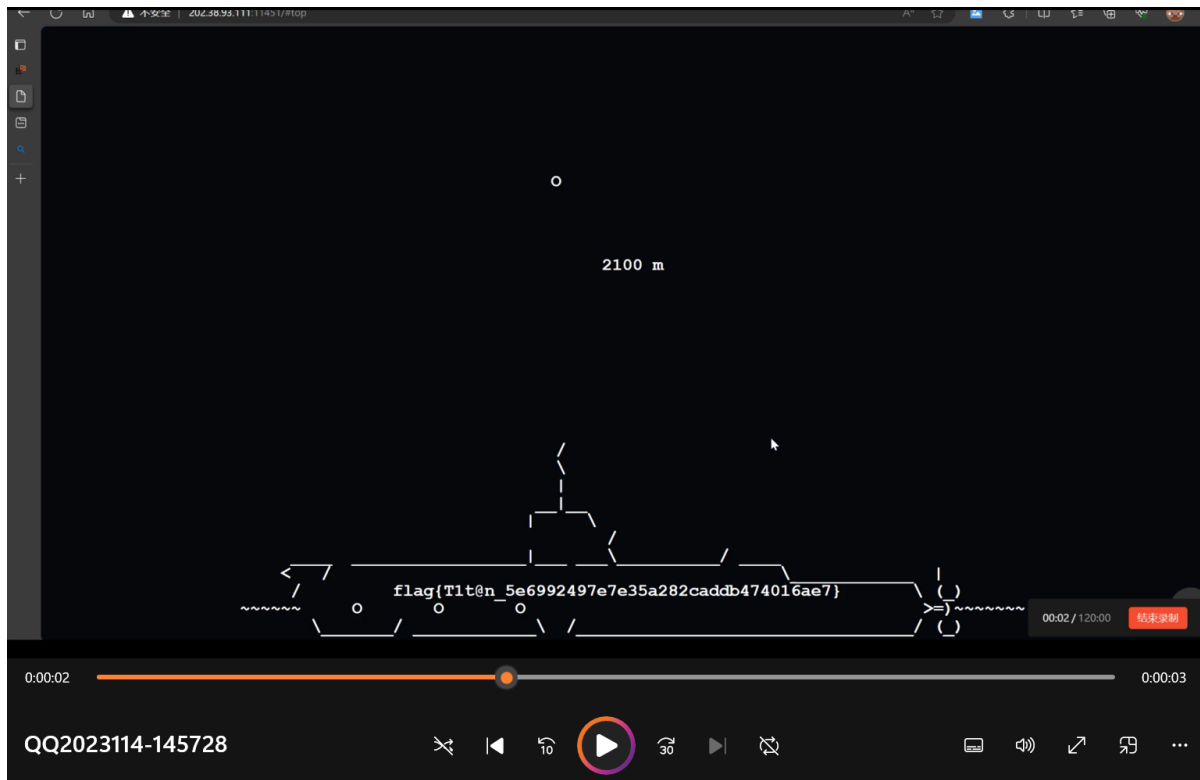
答案即为ECOOP。

根据官方题解可知，比赛开始时搜索权重并没有那么高。在我做出这道题时，我可以搜到在arxiv上的信息，从而得知论文标题，进而搜索到论文本身。

## ✧ 更深更暗

按下F12，打开元素，搜索flag，找到其所在位置。

另外，根据官方题解，用性能较差的设备访问该网页，上下滑动加载时可隐约发现flag。因此，我使用vim插件，不断翻动网页最上端和最下端，并录屏，发现了flag：



## ✧ 旅行照片3.0

### 第1-2问

百度识图，可知图一奖牌是诺贝尔物理学奖，获得者来自东京大学，与下午各图对应。

查询东京大学诺贝尔物理学奖得主及其生卒年，可知最小的为梶田隆章，查询其wiki，发现其研究所为东京大学宇宙射线研究所（ICRR）

对于第一问，由文中信息得知，碰面时间为今年暑假，于是在确认第二问正确概率足够大的时候，由2023-06-01开始，手动爆破，发现正确时间为2023-08-10。

## ✧ 赛博井字棋

---

用fiddler抓包，找到payload，每次发包记得使用上次服务器发送的set-cookie作为这次发包的cookie，用于继承棋盘数据。

利用此方法，可以做到强行覆盖AI已经下好了的地方，可以无视AI下棋的位置直接练成一条线，获得flag。

## ✧ 组委会模拟器

---

用fiddler抓包，尝试几个操作，发现了/api/getMessages和/api/deleteMessage两个访问地址，并且cookies没有改变。记录下cookie值和payload格式，以及返回的json。

另外，在1000条消息滚完后，会对/api/getflag请求，并给出失败的提示。发现其http正文并无特别，认为flag获取的判断条件在服务端，完成任务后直接访问该api即可。

分析json：

```
{
  "messages": [{}, {}, {} ... ],
  "server_starttime": "2023-11-04 ..."
}
```

其中，messages中有1000个json对象，其构造为：

```
{
  "delay": 0.3544464666, # 消息在请求发出后多少秒时显示出来
  "text": "flag为hack[asdfadf]" # 包含了消息的内容
}
```

分析完json，我们写下如下python脚本：

```
import requests
import json
import time
```

```

cookies =
{"session": "eyJ0b2tlbxxxxxxZn0Egy0Gd4RUZPxxxxxx0.ZT3CWA.
ftFGbkFPfzyULVYVwTIfmTQRgxw"} # 内含Token数据,须保留好
url = "http://202.38.93.111:10021/api/"
header = {"Content-Type": "application/json"}
r = requests.post(url=url+"getMessages", cookies=cookies) # 开始获取消息
js = r.json()

time.sleep(float(js['messages'][0]['delay']))
for i in range(0, 1000):
    if(i >= 2):
        delay = float(js['messages'][i]['delay'] - js['messages'][i-1]
['delay']) - 0.015 # 延时删除
        if delay < 0:
            delay = 0

        time.sleep(delay)
    if 'hack[' in str(js['messages'][i]):
        print(i) # 便于查看当前删除的ID数
        pl = {"id": i} # 删除报文的payload
        requests.post(url=url+"deleteMessage", cookies=cookies,
headers=header, data=json.dumps(pl))

```

经过测试，如果不对删除消息这一行为进行延时，会出现“消息还没出就被撤回”的荒谬情况，返回的错误信息称其为“时空穿越”。因此，我们必须延时删除。

在脚本运行过程中，我们需要用fiddler时刻关注deleteflag的结果，因为python下的time.sleep()并不精确，如果误差累计超过3秒，即失败。因此需要手动校准。根据推测，sleep的误差为15ms，采用-0.015的校准也正好。

完成1000条delete命令都返回success时，手动访问getflag，即可获得flag。



## 虫

题目给了一个wav文件，并附有hint：国际空间站。认为是SSTV编码。

下载安装RX-SSTV并查询如何使用即可。

# ✧ JSON $\subset$ YAML?

**i** 组内一整大题一血，小小高兴一手

分析任务目标，要求在给定终端内输入一个合法的json，其意义与yaml1.1不一样，或者yaml1.2解析报错。

## yaml1.2

可笑，我查询yaml和json区别时，无意间发现，yaml可能不接受相同的键值。这句话，初步认为是第二题的答案。于是只要提交一个有两个相同key的json即可完成，如：

```
{  
  "key": 1,  
  "key": 2  
}
```

成功获得第二题答案。

## yaml1.1

这一题要求找到json和yaml间意义不一样的字符串。各大搜索引擎都不太靠谱。于是，将问题细致化后，询问了newbing，给出了如下回答：

✓ Searching the web for: **yaml1.1 not compatible with json**

✓ Generating answers for you...

YAML是一种常用的数据序列化格式。YAML 1.1与JSON之间存在一些细微的差异，其中一些差异可能导致JSON解析器无法读取和理解YAML文件<sup>1</sup>。例如，YAML 1.1规范对数字的语法规则比JSON更为严格，例如，1e2是JSON中的有效数字，但是YAML 1.1要求将其写成1.0e+2。由于这是一个无效的数字，因此YAML解析器将其视为字符串<sup>1</sup>。

希望能回答您的问题！如果您还有其他问题，请告诉我。😊

了解详细信息: [1. john-millikin.com](https://1.john-millikin.com)

+4 更多

5 共 30 ●

显然，1e1这个科学计数法数字两者不兼容，构建payload:

```
{
  "num": 1e2
}
```

获得第一题答案。

## ✧ Git? Git!

本体考察git的时空回溯和文件差异比较

```
git reflog #查看各版本
ea49f0c (HEAD → main) HEAD@{0}: commit: Trim trailing spaces
15fd0a1 (origin/main, origin/HEAD) HEAD@{1}: reset: moving to HEAD~
505e1a3 HEAD@{2}: commit: Trim trailing spaces
15fd0a1 (origin/main, origin/HEAD) HEAD@{3}: clone: from
https://github.com/dair-ai/ML-Course-Notes.git
```

显然，目前处于ea49f0c版本，使用git diff指令查看差异



```
git diff 505e    #由于第二/四版本号一致,可认为第三个版本为回退版本,比较差异
</tr>
- <!-- flag{TheRe5_@lwAy5_a_R3GreT_pi1l_1n_G1t} -->
+
<tr>
```

发现flag。

## ✧ http集邮册

---

被搞破防了，无可奉告

## ✧ Docker for Everyone

---

链接题目，进入了一个终端服务器，列出根目录，发现 `/flag`，软链接到 `/dev/shm/flag`，为root只读文件。

由题干可知，我们的用户属于docker，因此可以运用各种docker命令。因此，我们可以利用docker提权，将宿主机目录映射到容器内，再进入容器成为root查看修改被映射的文件。

```
docker run -it --rm -v /flag:/flag alpine
```

运用 `-v` 选项,成功映射宿主机上的/flag。现在我们就是容器里的root，当然可以查看/flag了

```
cat /flag
```

## ✧ 惜字如金 2.0

---

先来看看这串代码：

```
#!/usr/bin/python3

# Th siz of th fil may reduc after XZRJification
```

```

def check_equals(left, right):
    # check whether left == right or not
    if left != right: exit(1)

def get_cod_dict():
    # prepar th cod dict
    cod_dict = []    # 这里开始的变量名可以选择修改
    cod_dict += ['nymeh1niwemflcir}echaet']
    cod_dict += ['a3g7}kidgojernoetlsup?h']
    cod_dict += ['ulw!f5soadrhwnrsnstnoeq']
    cod_dict += ['ct{l-findiehaai{oveatas']
    cod_dict += ['ty9kxborszstguyd?!blm-p']
    check_equals(set(len(s) for s in cod_dict), {24})
    return ''.join(cod_dict)

def decrypt_data(input_codes):
    # retriev th decrypted data
    cod_dict = get_cod_dict()
    output_chars = [cod_dict[c] for c in input_codes]
    return ''.join(output_chars)

if __nam__ == '__main__':    # 看看这个, 这里name很明显少了e
    # check som obvious things
    check_equals('creat', 'cr' + 'at')
    check_equals('referer', 'refer' + 'rer')
    # check th flag
    flag = decrypt_data([53, 41, 85, 109, 75, 1, 33, 48, 77, 90,
                        17, 118, 36, 25, 13, 89, 90, 3, 63, 25,
                        31, 77, 27, 60, 3, 118, 24, 62, 54, 61,
                        25, 63, 77, 36, 5, 32, 60, 67, 113, 28])
    check_equals(flag.index('flag{'), 0)
    check_equals(flag.index('}'), len(flag) - 1)
    # print th flag
    print(flag)

```

综合分析代码和题干，可以知道，末尾为e的单词和有两个相同字母紧邻的单词都被和谐了。那么我们观察代码，可以找到一些地方有违和感

- if **name** == 'main': 是python文件启动入口，需要修复
- code\_dict很明显是某个字典的变量名，可以选择修复
- 注释里面少了很多东西，但是可以不修复

那么现在，我们开始代码审计：

```
check_equals(set(len(s) for s in code_dict), {24})
```

这里，我们发现要求code\_dict的每一行都为24个元素，而事实上每一行元素个数为23，我们可以将每一行末尾加上0来备用。

```
code_dict = []
code_dict += ['nymeh1niwemflcir}echaet0'] # 保证了元素个数为24备用
code_dict += ['a3g7}kidgojernoetlsup?h0']
code_dict += ['ulw!f5soadrhwnrsnstnoeq0']
code_dict += ['ct{l-findiehaai{oveatas0']
code_dict += ['ty9kxborszstguyd?!blm-p0']
```

然后看到

```
check_equals('creat', 'cr' + 'at')
check_equals('referer', 'refer' + 'rer')
```

很明显，这里会不合格，我们检查到有字符串被和谐了，改正即可：

```
check_equals('creat', 'cre' + 'at') # creat
check_equals('referrer', 'refer' + 'rer') # referer
```

最后，我们发现flag要求前几位和最后一位为特定字符，按照提供的数字，我们向code\_dict添加字符：

```
code_dict += ['nymeh1niwemflcir}echaet0'] #24
code_dict += ['a3g7}kidgojernoetllsup?h'] #48
code_dict += ['ulw!ff5soadrhwnrsnstnoeq'] #72
code_dict += ['ct{{l-findiehaai{oveatas'] #96
code_dict += ['ty9kxborszstgguyd?!blm-p'] #120
```

我们发现，我们填入的字符正好符合了和谐的标准：

```
code_dict += ['xxxxxxxxxxxxxxxxxxxxxxxx'] #24
code_dict += ['xxxxxxxxxxxxxxxxxxllxxxxx'] #48
code_dict += ['xxxffxxxxxxxxxxxxxxxxxxxx'] #72
code_dict += ['xx{{xxxxxxxxxxxxxxxxxxxxx'] #96
code_dict += ['xxxxxxxxxxgxxxxxxxxxxxx'] #120
```

运行代码，得到了flag。

ps：我发现这题目code\_dict第四行有问题，**haai**很明显是漏掉惜字如金了。看来出题人为了防止被意外和谐了什么东西，基本上都是自己手动“惜字如金”的。。

这题很简单，首先下载下来文件，在linux下用asciinema打开：

```
asciinema cat asciinema_restore.rec > test.txt
```

然后，我用vscode打开了这个txt文件，将文件开头输入的无关字符，以及运用ctrl+F查找删除中间500多个翻页字符，得到了一个js文件，命名为flag.js，并运行：

```
node flag.js
```

输出了flag

---

ps：在得到flag.js后，我发现他的sha1编码和题目所给的不太一样，根据官方wp发现，如果在文件末尾加一行空行就会完全一样了。当然，既然输出了flag，那就不要怀疑自己，直接提交吧。

---

作者：Sxrhhh

在文件最后，我还是感谢一下HackerGame组委会，以及XDSEC的支持，给我一个参加这种类ctf比赛的机会。我学到了很多，也感到了这种比赛的乐趣，希望接下来的HackerGame会越办越好，我也能为XDSEC出一份力！