استراحتگاه هیولاها

توجه: در این سوال مجاز به استفاده از ابزارهای هوش مصنوعی (ChatGPT، Copilot، Claude و ...) هستید. اما هر تابع (چه خودتان نوشته باشید چه هوش مصنوعی) باید یک docstring داشته باشد که آن را خودتان نوشته باشید و در آن در حداقل ۱۰ کلمه توضیح دهید تابع مربوطه چه کاری را انجام می‌دهد و چگونه انجامش می‌دهد.

آقای آب‌دماغ که به تازگی صاحب یک دفتر مشاوره املاک شده، سالیوان را برای کار در دفترش استخدام کرده است. سالیوان این ایده را داشت که یک وبسایت برای دفتر بسازد و در نتیجه پس از جستجوهای بسیار، به سراغ شما آمده تا به او در ساخت این سایت کمک کنید!

برای این سوال تعدادی لینک (URL) سرویس و منطق و پارامترهای آن‌ها به شما داده و خروجی مورد انتظارشان بیان می‌شود.

آرگومان هر تابعی که سرویس متناظرش پارامتر دارد، باید یک دیکشنری باشد که پارامترها را از آن استخراج می‌کنید. بدیهتا توابعی که سرویسشان پارامتری ندارد، آرگومانی هم ندارند.

خروجی هر تابع شما باید یک tuple باشد که عضو نخست آن یک کد (که به شما گفته می‌شود) و خروجی مورد انتظار سرویس مربوطه باشد.

سرویس‌ها

/api/v1/get-users

پارامترهای درخواست

نام پارامتر	توضیح
ندارد	

منطق

اطلاعات تمامی کاربران گرفته شود.

پاسخ

کد خروجی	داده‌ی خروجی	نوع خروجی	توضیح
200	اطلاعات کاربران	`dictionary`	اطلاعات تمامی کاربران به شکل دیکشنری برگردد.

/api/v1/create-user

پارامترهای درخواست

نام پارامتر	توضیح
username	رشته‌ای حاوی نام کاربری
password	رشته‌ای حاوی رمز کاربر

نام پارامتر	توضیح
age	عدد نشان‌دهنده‌ی سن کاربر
name	رشته‌ای حاوی نام و نام خانوادگی کاربر

منطق

کاربر جدید با اطلاعات داده شده در پارامترها ساخته می‌شود.

پاسخ

کد خروجی	داده‌ی خروجی	نوع خروجی	توضیح
200	User created successfully	`string`	اطلاعات تمامی کاربران به شکل دیکشنری برگردد
400	Username already exists	`string`	در صورتی که نام کاربری‌ای که کاربر در درخواستش ارسال نموده تکراری است این پیغام را برگردانید.
400	Invalid request	`string`	در صورتی که داده‌های موجود در پارامترهای درخواست به هر شکلی نامعتبر بودند این پیام را برگردانید.

/api/v1/get_houses

پارامترهای درخواست

نام پارامتر	توضیح
ندارد.	

منطق

اطلاعات تمامی املاک گرفته شود.

پاسخ

کد خروجی	داده‌ی خروجی	نوع خروجی	توضیح
200	اطلاعات املاک	`dictionary`	اطلاعات تمامی کاربران به شکل دیکشنری برگردد.

/api/v1/create_house

پارامترهای درخواست

نام پارامتر	توضیح
house_id	شناسه ملک که یک عدد صحیح است.
addresss	آدرس ملک که یک رشته است.
price	قیمت ملک که یک عدد صحیح است.

منطق

یک ملک جدید در سامانه با مشخصات داده شده ثبت گردد.

پاسخ

کد خروجی	داده‌ی خروجی	نوع خروجی	توضیح
200	House created successfully	`string`	پیامی که در صورت ثبت موفق ملک باید برگردانید.
400	Invalid Request	`string`	در صورتی که داده‌های پارامتر به هر صورتی غیرمعتبر (یا حتی دارای ID تکراری) بودند، برگردانید.

/api/v1/houses/assign

پارامترهای درخواست

نام پارامتر	توضیح
house_id	شناسه ملک که یک عدد صحیح است.
username	نام کاربری صاحب ملک که رشته است.

منطق

یک ملک موجود در سامانه با مشخصات داده شده به یک کاربر با مشخصات داده شده متناظر گردد. برای سادگی کار فرض کنید هر کاربر حداکثر دارای یک ملک و هر ملک متعلق به حداکثر یک کاربر است.

پاسخ

کد خروجی	داده‌ی خروجی	نوع خروجی	توضیح
200	House assigned successfully	`string`	پیامی که در صورت تناظر موفق یک ملک به یک کاربر باید برگردانید.
400	Invalid Request	`string`	در صورتی که داده‌های پارامتر به هر صورتی غیرمعتبر (یا حتی وجود نداشتن هر یک از طرفین در سامانه) بودند، برگردانید.

/api/v1/houses/users

پارامترهای درخواست

نام پارامتر	توضیح
username	رشته‌ای حاوی نام کاربری

منطق

تمامی ملک‌های موجودی که صاحبشان کاربر درخواست شده است را به صورت دیکشنری برگردانید.

پاسخ

کد خروجی	داده‌ی خروجی	نوع خروجی	توضیح
200	اطلاعات املاک	`dictionary`	اطلاعات املاکی که متعلق به کاربر درخواست شده هستند.
404	User not found	`string`	پیامی که باید برگردانید، در صورتی که کاربری با نام کاربری درخواست شده موجود نبود یا کاربر مربوطه ملکی نداشت.

نکات مهم سوال

نکته: برای ذخیره‌ی داده‌ها می‌توانید از ساختاری مشابه قطعه کد زیر استفاده کنید:

```
USERS = {}
HOUSES = {}
HOUSES_ASSIGNED = {}
```

نکته: شما باید لینک‌ها را همراه متدها و توابعشان (که خودتان زدید) به صورت زیر در یک دیکشنری در انتهای فایل پاسختان ذخیره کنید، **هیچ کدام از اسم‌های متغیرها و کلیدها را تغییر ندهید**:

```
URLS = {
    "/api/v1/get_users": # تابع شما
    "/api/v1/create_user": # تابع شما
    "/api/v1/get_houses": # تابع شما
    "/api/v1/create_house": # تابع شما
    "/api/v1/houses/assign": # تابع شما
    "/api/v1/houses/users": # تابع شما
}
```

ورودی و خروجی

این سوال ورودی و خروجی نداشته و داور کوئرا پاسخ‌های شما را از روی توابعی که در متدهای هر URL در متغیر `URLS` قرار داده‌اید بررسی می‌کند.

راهنمایی‌ها

▼ راهنمایی ۱

برای امتحان کردن راه‌حلتان می‌توانید کد زیر را به انتهای فایلتان اضافه کرده و نتایج را بررسی کنید:

```
while True:
    request = input()
    if request == ["EXIT"]:
        break
    try:
        url, params = request.split("?")
        params = params.split("&")
        params = {param.split("=")[0]: param.split("=")[1] for pa
        request = params
        print(URLS[url](request))
    except ValueError:
        url = request
        print(URLS[url]())
```

ورودی نمونه:

```
/api/v1/get_users
/api/v1/create_user?username=john&age=20&password=1234&name=John
/api/v1/create_user?username=jane&age=21&password=1234&name=Jane
/api/v1/create_user?username=doe&age=22&password=1234&name=Doe
/api/v1/create_user?username=doe1212&age=22&password=1234&name=Doe
/api/v1/create_user?username=doe1232&age=22&password=1234&name=Doe
/api/v1/get_users
/api/v1/get_houses
/api/v1/create_house?house_id=1&address=123MainSt&price=100000
```

```
/api/v1/get_houses
/api/v1/create_house?house_id=2&address=456MainSt&price=200000
/api/v1/get_houses
/api/v1/create_house?house_id=3&address=789MainSt&price=300000
/api/v1/create_house?house_id=4&address=101MainSt&price=400000
/api/v1/create_house?house_id=5&address=102MainSt&price=500000
/api/v1/houses/assign?username=john&house_id=1
/api/v1/houses/assign?username=jane&house_id=2
/api/v1/houses/assign?username=doe&house_id=3
/api/v1/houses/assign?username=john&house_id=4
/api/v1/houses/users?username=john
/api/v1/houses/users?username=jane
/api/v1/houses/users?username=doe
/api/v1/get_users
EXIT
```

خروجی مورد انتظار:

```
(200, {})
(200, 'User created successfully')
(200, 'User created successfully')
(200, 'User created successfully')
(200, 'User created successfully')
(200, 'User created successfully')
(200, {'john': {'password': '1234', 'name': 'John', 'age': 20}, 'jane
(200, {})
(200, 'House created successfully')
(200, {'1': {'address': '123MainSt', 'price': 100000}})
(200, 'House created successfully')
(200, {'1': {'address': '123MainSt', 'price': 100000}, '2': {'address
(200, 'House created successfully')
(200, 'House created successfully')
(200, 'House created successfully')
(200, 'House assigned successfully')
(200, 'House assigned successfully')
(200, 'House assigned successfully')
(200, 'House assigned successfully')
(200, {'address': '101MainSt', 'price': 400000})
(200, {'address': '456MainSt', 'price': 200000})
```

```
(200, {'address': '789MainSt', 'price': 300000})
(200, {'john': {'password': '1234', 'name': 'John', 'age': 20}, 'jane
```

▼ راهنمایی ۲

در نهایت ساختار بخش توابع کدتان باید شبیه کد زیر شود. فرض کنید اسم توابع را دقیقاً از URLها گرفته باشیم:

```python
def api_v1_get_users():
    """
توضیح کد
    """
کد شما #

def api_v1_create_user(request_body: dict):
    """
توضیح کد
    """
کد شما #

def api_v1_get_houses():
    """
توضیح کد
    """
کد شما #

def api_v1_create_house(request_body: dict):
    """
توضیح کد
    """
کد شما #

def api_v1_assign_house(request_body: dict):
    """
توضیح کد
    """
کد شما #

```

```
def api_v1_get_users_houses(request_body: dict):
    """
توضیح کد
    """
# کد شما
```

▼ راهنمایی ۳

در ساختار بیان شده، کد سرویس `api_v1_create_user` به صورت زیر خواهد بود:

```
def api_v1_create_user(request_body: dict):
    try:
        username = request_body["username"]
        password = request_body["password"]
        name = request_body["name"]
        age = int(request_body["age"])
    except Exception as e:
        return (400, "Invalid request")
    if username in USERS:
        return (400, "Username already exists")
    USERS[username] = {"password": password, "name": name, "age"
    return (200, "User created successfully")
```

که نوشتن docstring این تابع بر عهده شماست.

بیشتر بدانید

▼ مطالعه‌ی آزاد: معماری MVC

در مهندسی نرم‌افزار، **MVC** یا Model-View-Controller یک الگوی طراحی است که برای سازماندهی بهتر کدها و جداسازی مسئولیت‌ها در برنامه‌نویسی استفاده می‌شود. این معماری سه لایه‌ی اصلی دارد:

- **مدل (Model)** که مسئول مدیریت داده‌های برنامه است.
- **نما (View)** که داده‌ها را به کاربر نمایش می‌دهد. (واسطه کاربری)

- **کنترلر (Controller)** که تعاملات کاربر را مدیریت کرده و بین مدل و نما ارتباط برقرار می‌کند. (منطق اجرایی برنامه)

این ساختار باعث ساده‌تر شدن فرآیند توسعه، بهبود خوانایی کد و افزایش قابلیت نگهداری برنامه می‌شود. برای مطالعه بیشتر، این لینک پیشنهاد می‌شود.

▼ مطالعه‌ی آزاد: REST API

در مهندسی نرم‌افزار، **REST API** (Representational State Transfer) یک سبک معماری برای طراحی سرویس‌های وب است که با استفاده از اصول HTTP، ارتباط بین کلاینت و سرور را ساده و استاندارد می‌کند. REST بر اساس چند اصل کلیدی طراحی شده است:

- **منابع (Resources):** هر چیزی در سیستم به‌عنوان یک منبع شناخته می‌شود که از طریق URL منحصربه‌فرد قابل دسترسی است.
- **روش‌های HTTP:** عملیات روی منابع با استفاده از متدهای HTTP مانند `GET`، `POST`، `PUT` و `DELETE` انجام می‌شود.
- **وضعیت‌زدایی (Statelessness):** سرور هیچ اطلاعاتی از وضعیت کلاینت را ذخیره نمی‌کند و هر درخواست باید شامل تمام اطلاعات لازم باشد.

این معماری ساده، انعطاف‌پذیر و مقیاس‌پذیر است و آن را به یکی از محبوب‌ترین روش‌ها برای ساخت APIهای مدرن تبدیل کرده است. برای جزئیات بیشتر، اینجا را ببینید.

▼ مطالعه‌ی آزاد: پایگاه‌های داده و سامانه‌های مدیریت آن‌ها

احتمالاً در حین حل این سوال به چالش‌هایی در رابطه با ذخیره‌ی داده‌هایتان خورده‌اید. اگر این سوال برایتان پیش آمده که سرویس‌های بزرگ مانند Google و Netflix و Spotify (که در آن‌ها داده و دسترسی به آن بسیار حائز اهمیت است) چگونه این موضوع را مدیریت می‌کنند، پاسخ سوال شما پایگاه داده (Database) و سامانه‌های مدیریت پایگاه داده (DBMS) است! اگر علاقه‌مندید که بیشتر در مورد آن‌ها بدانید می‌توانید از این لینک و این لینک شروع کنید.

▼ مطالعه‌ی آزاد: مدل‌های بزرگ زبانی

از آنجا که برای حل این تمرین می‌توانید از ابزارهای هوش مصنوعی استفاده کنید، چند ابزار را در ادامه برای شما آورده‌ایم. (اگر تمایل دارید بدانید LLMها چه هستند و چگونه کار می‌کنند، مطالعه‌ی این لینک را پیشنهاد می‌کنیم)

ابزارهای هوش مصنوعی:

- ChatGPT
- Claude
- Gemini
- Microsoft Copilot

▼ مطالعه‌ی آزاد: فرمان‌نویسی (پرامپت‌نویسی)

دقیقاً همانند انسان‌ها، LLMها هم می‌توانند با برخی گفتارها و لحن‌ها و ... بهتر کار کنند! به همین دلیل از مدتی نسبتاً کوتاه بعد از ترویج استفاده از LLMها، مهارتی به نام فرمان‌نویسی و حرفه‌ای به نام مهندسی فرمان یا prompt engineering بوجود آمد. در صورت تمایل می‌توانید از این لینک و این لینک با این مهارت بیشتر آشنا شده و حتی در حل این تمرین از آن استفاده کنید تا نتایج بهتری بگیرید.

▼ مطالعه‌ی (خیلی!) آزاد: سئو (SEO)

مطالعه‌ی آزاد: سئو (SEO)

سئو یا بهینه‌سازی موتور جستجو (Search Engine Optimization که به اختصار SEO نوشته می‌شود) فرآیندی است برای بهبود دیده‌شدن وب‌سایت‌ها در موتورهای جستجو (مانند Google) که باعث افزایش ترافیک ارگانیک می‌شود. این فرآیند ترکیبی از هنر و علم است که بر اساس اصول زیر عمل می‌کند:

- **تحقیق کلمات کلیدی:** شناسایی کلماتی که کاربران در جستجوی اطلاعات موردنظر خود استفاده می‌کنند.
- **محتوای باکیفیت:** تولید محتوای مفید، جذاب و مرتبط که به نیازهای کاربران پاسخ دهد.
- **لینک‌سازی:** ایجاد و دریافت لینک‌های باکیفیت از وب‌سایت‌های دیگر برای افزایش اعتبار سایت.

- **بهینه‌سازی فنی:** بهبود کد، ساختار و سرعت وب‌سایت برای اطمینان از دسترسی آسان و تجربه کاربری بهتر.
- **سازگاری با موبایل:** طراحی وب‌سایتی که روی دستگاه‌های مختلف (به‌ویژه موبایل) به‌خوبی نمایش داده شود.
- **تحلیل و بهبود مستمر:** استفاده از ابزارهایی مانند Google Analytics و Google Search Console برای رصد عملکرد و اعمال بهبودهای لازم.

سئو یک فرآیند طولانی‌مدت و استراتژیک است که به‌طور مداوم باید روی آن کار کرد. برای اطلاعات بیشتر، پیشنهاد می‌شود این لینک و این لینک را مطالعه کنید.